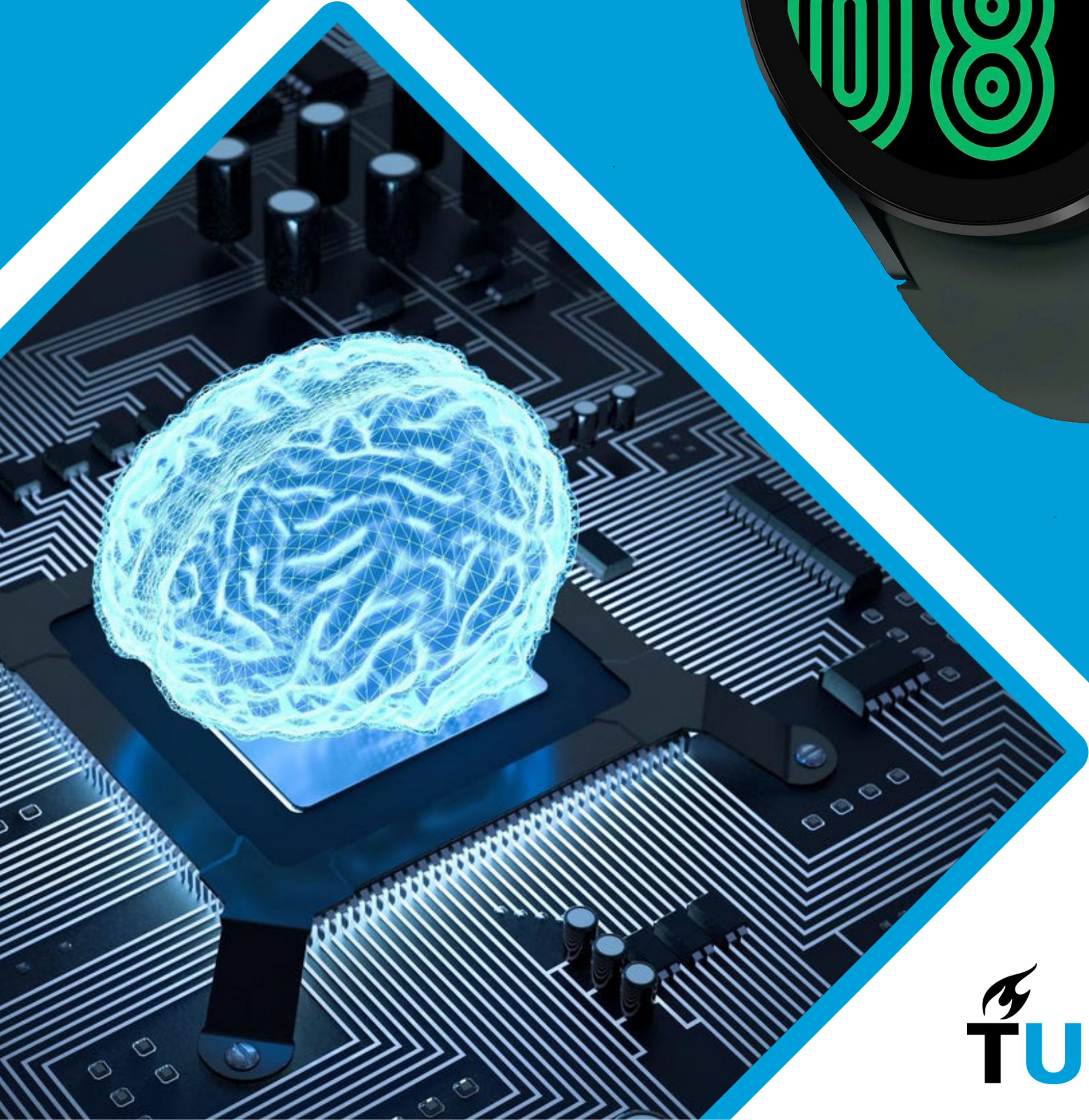
Facilitating healthcare using smartwatches

Smartwatch data acquisition platform

Shijie Liao
Robbin Poll
Marijn Sluijs

June, 2022



Facilitating healthcare using smartwatches

Smartwatch data acquisition platform

by

Shijie Liao
Robbin Poll
Marijn Sluijs

to obtain the degree of Bachelor of Science

Student numbers:	4705939 (Shijie Liao)
	4732324 (Robbin Poll)
	5071364 (Marijn Sluijs)
Supervisors:	Dr. B. Abdikivanani
	Prof. dr. ir. A.J. van der Veen
Project Duration:	April, 2022 - June, 2022
Faculty:	Faculty of Electrical Engineering,
	Mathematics and Computer Science, Delft

TUDelft

Abstract

This report details the design and implementation of a subsystem providing a web-based platform for smartwatch data acquisition. A smartwatch application is developed to read out the accelerometer, gyroscope and heart rate sensor on the smartwatch and transmit the sensor data to the platform. A platform is developed to receive the data and store it in a database. Recordings can be downloaded from the platform to use in research of human activity. The platform presents the type of activity the smartwatch user is doing using machine learning models, by integrating the other subsystem. Multiple tests have been performed to analyse and improve the performance of the system. The battery life of the smartwatch has been tested using various settings in the smartwatch application to determine the most power-efficient settings.

Preface

This thesis is written in context of the Bachelor Graduation Project. In the span of two months, we explored the world of smartwatches and we present the results in this document. We created a smartwatch application and web-based platform for recording human activity. Together with our three other group members interested in artificial intelligence, we integrated classification methods into the platform to recognize the type of human activity. The GitHub repository containing the smartwatch application, web-based platform, and classification methods can be found here: `https://github.com/MarijnSluijs/GetSmart`.

We would like to express our gratitude to dr. B. Abdikivanani and prof. dr. ir. A.J. van der Veen for their supervision and support during the project. In addition, we would like to thank our colleagues Ricardo Cavalini, Ibrahim Hassan and Thomas Pouwels for their collaboration with us on the project.

Shijie Liao, Robbin Poll & Marijn Sluijs,
Delft, June 2022

Contents

1 Introduction

The presence of smartwatches is expanding rapidly with 40 million smartwatches sold in Q4 of 2021 and a year-over-year shipments increase of 21% in the same year [1]. The newest smartwatches contain more types of sensors, which provide lots of new opportunities for engineers and researchers. Smartwatches become an accessible base for developing suitable applications without requiring extensive resources. Smartwatch sensors have become increasingly more reliable over time. This has improved their recognition and acknowledgement in healthcare services and research.

Healthcare

Smartwatches started with simple functions like heart rate detection and setting alerts for medication reminders. By now they have progressed to more advanced applications like early detection of diseases such as arrhythmia, abnormalities in the heart rhythm. The sensors that make those advanced applications in healthcare possible are the PPG and ECG sensors.

Newer smartwatch models (such as the Apple Watch 7 [2]) use these sensors to detect signs of arrhythmia in their health app. Multiple studies have been conducted to test the detection accuracy of atrial fibrillation (a type of arrhythmia) using ECG data and conclude that smartwatches are already quite accurate [3, 4, 5]. The comparison in these studies is made with traditional 12-lead ECG's, which are more extensive and time-consuming for the patient and healthcare worker. Recently, Mishra et al. [6] even used smartwatches for pre-symptomatic detection of COVID-19.

Human Activity Recognition

Smartwatches also contain other sensors like the accelerometer and the gyroscope which open the door to research in human activity recognition (HAR). HAR is an active field of research due to the many possible applications it has in domains such as healthcare monitoring [7], surveillance, and exercise tracking. For instance, HAR can recognise early mobility for Intensive Care Unit patients. Early mobility is essential for ICU patients who suffer from long-time immobilisation [8].

Improvements

More research is still needed to approve or improve the performance of diagnosis based on smartwatch sensor data. For example some research questions that still need to be answered are: how good is the accuracy of the smartwatch sensors compared to traditional methods? How does their reliability hold up under different conditions such as swimming and jogging?

The first step to investigate these questions, is to collect data using these smartwatches and build the required database for them. A platform which enables easy and fast data collection from smartwatches is necessary and invaluable for researchers. Therefore the goal of this project is to provide a stable base for research on smartwatch data. To make this possible, a smartwatch app and a web based platform are designed and implemented.

Our app ”GetSmart” reads out the sensor data from the smartwatch and sends it to the web-based platform, where it can be downloaded. This allows for fast and easy collection and generation of data sets from smartwatches sensors. The design of the app and platform is the main focus of this thesis.

In the second part of the project, the focus is on human activity recognition. Accelerometer and gyroscope data is collected using the platform and used to generate a data set. This data set is used to train and test machine learning algorithms for human activity recognition. It is important to note that initially the algorithms are designed using an existing data set, as the app and platform were still in development.

Human Activity Recognition

As mentioned before, HAR has many applications in domains such as healthcare monitoring, surveillance, and exercise tracking. There are, however, some challenges that HAR faces such as data collection, privacy and power consumption.

Multiple measuring methods can be applied for HAR. Cameras are one of the possibilities to collect data for HAR. While cameras provide lots of useful data, the privacy invasion that comes with it causes big societal problems. Although there are many solutions to reduce the collection of sensitive data [9] [10], they can't ensure privacy-preserving video's. Besides privacy, another main problem with collecting data using a camera is that the participants are bound to the view of the camera.

To overcome these issues, smartphones [11] [12] and wearable sensors are valid solutions. Smartphones are less viable if the position change of the smartphone gets taken into account. Changing the phone's position, e.g. from pocket to purse, can influence the incoming data of the HAR. In doing so, the accuracy of the prediction model gets influenced. For wearable sensors, this is not the case. The privacy infringement in these scenarios is also limited to the user of the device. In contrast to cameras, an accelerometer does not record its surroundings.

Wearable sensors can be divided into two categories: sensors on multiple parts of the body [13], or smartwatches [14] [15]. The first category is an accurate way to measure human activity, especially with suitable measuring points. However, depending on the position and the connections, users may find the sensors uncomfortable and can thereby influence the data by moving differently than normal. Smartwatches however are the most viable solution as they are relatively comfortable and do not interfere with movement.

Acquisition

Commercially available smartwatches from manufacturers such as Samsung, Apple, and Garmin already provide users insight on recordings such as heart rate, ECG, and blood pressure. They are presented in a graphical user interface on an app but they do not provide or save continuous recordings of raw data that can be used for research and development. Several approaches have been proposed in literature to gather required data for researchers like the Real-time Online Activity and Mobility Monitor (ROAMM) framework [16], where several characteristics of such a framework are listed. The ROAMM framework consists of an application for data collection, a server for data storage, and an online monitoring tool. This framework focuses on patient well-being, patients can indicate their mood, pain and fatigue level on the smartwatch. The framework developed during this project focuses on creating recordings of human activities using smartwatches.

Recognition

After acquiring the data from the smartwatch, the next step is to process and use it for activity recognition. This can be done using machine learning. Many classification models are proposed in literature that use sensor data to perform human activity classification. They range from traditional machine learning to deep learning algorithms. It is also possible to combine different classifiers in an ensemble. Using an ensemble can increase the accuracy of human activity recognition.

Problem description

As described in the previous sections, there are many applications for data recorded by smartwatches. However, to the best of our knowledge, there is no easy and user friendly platform to collect raw smartwatch sensor data. In this project we have developed an easy-to-use smartwatch application and a web-based platform for researchers, to record smartwatch data, send the data to the platform and retrieve the data from the platform for their applications. The recordings can be downloaded as a Comma Separated Value (CSV) file from the server. The recordings made during this project will be used to train a HAR machine learning model, in order to demonstrate the usefulness of raw smartwatch data collected in this way.

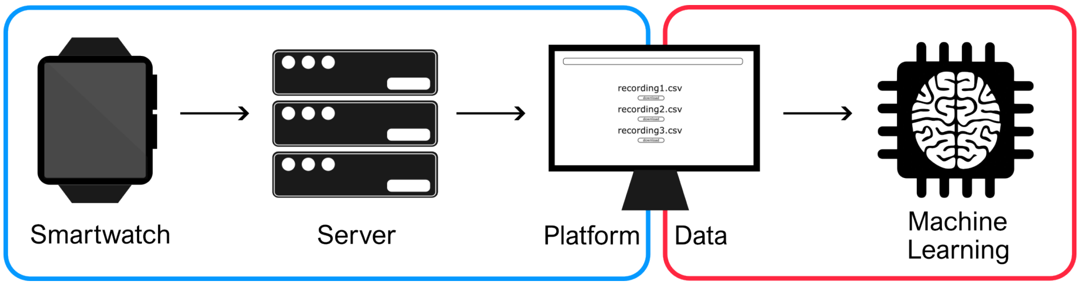


Figure 1.1: Project overview.

Subdivision

To address these problems, the six group members are divided into two subgroups. One of the subgroups focuses on the data acquisition and storing the data on a web-based platform, and the other subgroup focuses on the data classification. Figure 1.1 shows the proposed workflow. This thesis will focus on the data acquisition subgroup, the thesis of the other subgroup can be found in [17].

Thesis Outline

This thesis is structured as follows. Chapter 2 will describe the program of requirements for the complete system and the requirements of the subsystem discussed in this thesis. Chapter 3 will discuss the design of the smartwatch application and Chapter 4 discusses the design of the server. Chapter 5 will list the procedures used to test the application and server and the results are shown and discussed. In Chapter 6 conclusions are drawn and recommendations for future work are given. In Appendix A the user manual of the software can be found.

2 Program of Requirements

The program of requirements is split into two parts. First, the general requirements of the complete system are given. Next, the requirements of the subsystem discussed in this thesis are given. The mandatory requirements must be met for the project to be successful. The trade-off requirements were listed and selectively implemented, as they are drawn up to improve the system but are not essential to creating a functional prototype.

2.1. General Requirements

The goal of the project is to facilitate research on applications that use smartwatch data. First, we allow for the easy recording of smartwatch data using an app and web-based platform. The app on the smartwatch collects and sends raw sensor data to a server. The server receives the data and makes recordings. These recordings can then be used in various settings. The general requirements of the project are stated below.

1. The smartwatch application must acquire sensor data from a commercially available smartwatch.
2. The smartwatch application must transmit sensor data to the server.
3. The server must be able to record the sensor data.
4. The website of the server must present the active users and recordings.
5. The server must integrate a machine learning algorithm to recognize the wearers' current activity.
6. The activity recognition must be done within 5 seconds

2.2. Data Acquisition Requirements

The goal of this subgroup is to develop a smartwatch application to acquire sensor data and set up a server to make and store recordings of smartwatch sensors. In the next sections, the requirements of the smartwatch application and the server are listed.

Smartwatch Application Mandatory Requirements

1. The application must acquire data from the accelerometer, gyroscope and heart rate sensor on the smartwatch.
2. The application must transmit data to the server.
3. The application must transmit a user ID and session token together with the data, used to distinguish different users and different recording sessions on the server.
4. The application must be able to create training sets used to train and test the machine learning algorithms. The smartwatch wearer will indicate in the app what activity he/she is performing.

Smartwatch Application Trade-off Requirements

1. The power consumption of the smartwatch should be optimized to make continuous monitoring over a period of 8 hours (a working day) possible.
2. The application should continue running in the background.
3. The application should acquire data from the ECG and PPG sensor.

Server Mandatory Requirements

1. The server must receive the data sent by the smartwatch application.
2. The server must store the data in a database.
3. The server must be accessible via a website.
4. The user must be able to download the recordings from the server using the website.
5. The server must distinguish recordings from different users and recording sessions.
6. The server must show the users actively sending data.
7. The server must have a SSL certificate to enable an encrypted connection.

Server Trade-off Requirements

1. The server should integrate a machine learning algorithm to recognize the wearers' current activity. The algorithm is created by the other subgroup of this project, the Data Classification group.

3 Smartwatch Application Design

The design choices of the smartwatch application created during this project are discussed in this chapter. The purpose of the application is to acquire data from the sensors on the smartwatch and transmit it to the server. At first, the smartwatch model chosen for this project will be discussed. Afterwards, the working of the sensors and the design of the application for the smartwatch is discussed.

3.1. Smartwatch Choice

The goal of the project is to make recordings of human activity using smartwatches, so the first choice made is which smartwatch to use for the prototype. The specifications influencing the choice of the smartwatch are stated below.

- Type of sensors
- Price
- Market share
- Battery life
- Operating system

These specifications will be discussed in upcoming sections and a comparison is made between a variety of commercially available smartwatches.

3.1.1. Type of Sensors

To record human activity, the smartwatch should have an accelerometer and gyroscope sensor. The heart rate of the person wearing the smartwatch can also be helpful when classifying certain types of human activity, so a heart rate sensor should be included. The smartwatch should also have an ECG and PPG sensor, since acquiring ECG and PPG data is one of the trade-off requirements. In Table B.1 in the Appendix, various commercially available smartwatches are compared with their sensor availability. Three smartwatches, the Apple Watch 7, Samsung Galaxy Watch4 and Fitbit Charge 5, have all sensors required for this project.

3.1.2. Price

The price is an important factor in the choice for the smartwatch used in this project. Choosing a more affordable smartwatch will make the platform financially more interesting for researchers. Table B.1 in the Appendix shows the price of various commercially available smartwatches. Looking at the prices of the smartwatches of interest discussed in the section above, the Fitbit Charge 5 is the most interesting option. The Samsung Galaxy Watch4 is a bit more expensive, while the Apple Watch 7 is more than double the price.

3.1.3. Market Share

The platform should be able to record sensor data from most of the smartwatches available on the market, so the smartwatch with the highest market share should be chosen. Figure 3.1 shows the market share of different operating systems used on smartwatches. The operating system WatchOS created for Apple Watches has the highest market share with 28.0% in Q3 2020 and 21.8% in Q3 2021. The market share of the operating system Wear OS developed by Google increased from 3.2% in Q3 2020 to 17.3% in Q3 2021 due to the addition of Wear OS to the smartwatches made by Samsung. Wear OS is already available on multiple smartwatch brands and with the possible addition of Fitbit and Tizen watches to Wear OS, there is a market share opportunity of 23.7%, making it the operating system with the highest market share.

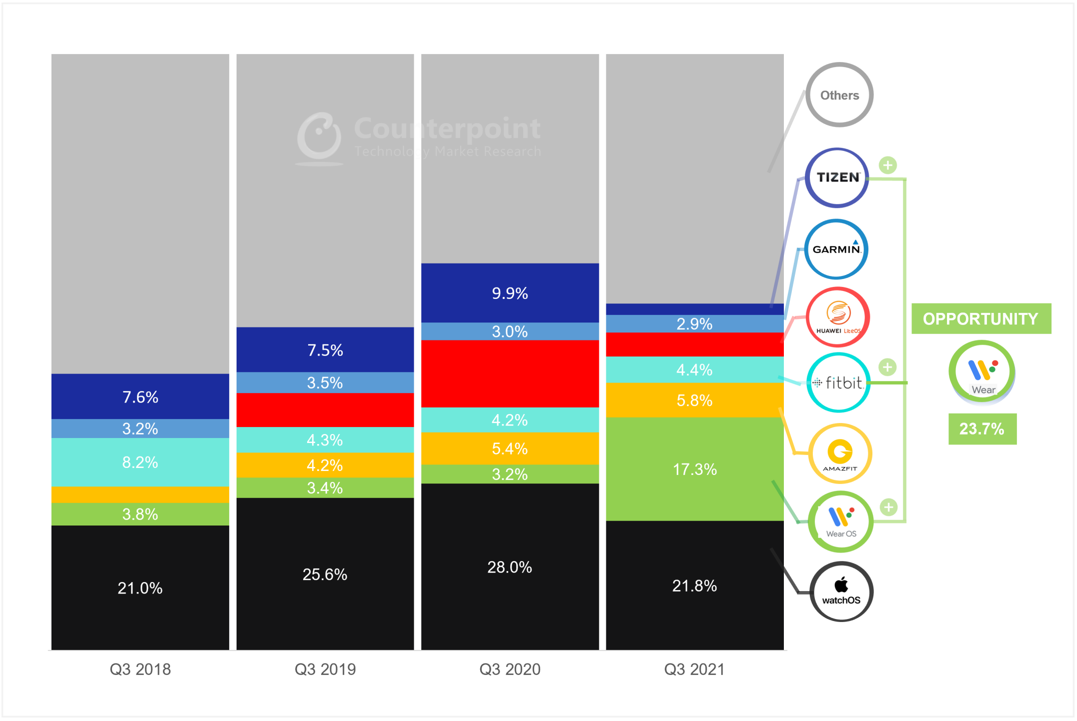


Figure 3.1: Smartwatch Shipment Share by OS, Q3 2018-Q3 2021 [18].

3.1.4. Battery Life

The battery life of the smartwatch is an important factor for this project. With longer battery life, the smartwatch will be able to send sensor data for a longer period of time before needing a recharge. In Table 3.1 the battery capacity and expected battery life of the three relevant smartwatches are shown. The Apple Watch 7 has the lowest battery life with 18 hours, while the Fitbit Charge 5 can last up to 5 days. The reason for the Fitbit to last so long while having a low battery capacity of 70 mAh is because the smartwatch has few features. Therefore the Fitbit Charge 5 is often called an activity tracker instead of a smartwatch.

Table 3.1: Battery life on three commercially available smartwatches.

	Battery capacity	Expected battery life
Apple Watch 7	309 mAh	18 hours [19]
Samsung Galaxy Watch4 Classic	361 mAh	40 hours [20]
Fitbit Charge 5	70 mAh	5 days [21]

3.1.5. Final Choice

Based on the required sensors for this project, there are three smartwatch options: Apple Watch 7, Samsung Galaxy Watch4 and Fitbit Charge 5. Looking at the price, the Samsung Galaxy Watch4 and Fitbit Charge 5 are more interesting optons than the Apple Watch 7. The Fitbit has the longest battery life, but the functionality is limited, making it sub-optimal. The expected battery life of the Samsung Watch 4 is more than twice the battery life of the Apple Watch 7. As explained in Section 3.1.3, the operating system Wear OS, running on the Samsung Galaxy Watch4, is of the most interest for this project. The operating system Wear OS gives developers access to sensors on the smartwatch, so it can be used in this project. Based on these conclusions, the Samsung Galaxy Watch4 is the chosen smartwatch for this project.

3.2. Sensor operation

The first requirement of the smartwatch application states that it must acquire data from the accelerometer, gyroscope and heart rate sensor (Section 2.2). Before the design of the application is discussed, the functioning of these three sensors will be explained.

3.2.1. Accelerometer

The accelerometer is an electromechanical device that measures acceleration force. The Samsung Galaxy Watch4 equips a 3-axis accelerometer, measuring acceleration in both x,y and z direction. For each axis, the accelerometer has the mechanism shown in Figure 3.2. The mechanism consists of fixed plates and a moving plate connected to a spring. There exists a capacitance between the fixed and moving plate. When acceleration is applied, the moving plate moves with respect to the fixed plate, creating a difference in capacitance (shown on the right of Figure 3.2). This difference in capacitance results in a sensor output voltage, proportional to the acceleration in m/s^2 [22]. The accelerometer takes Earth's gravity into account, meaning that the total magnitude of the acceleration in x,y and z direction at rest is equal to 9.8 m/s^2.

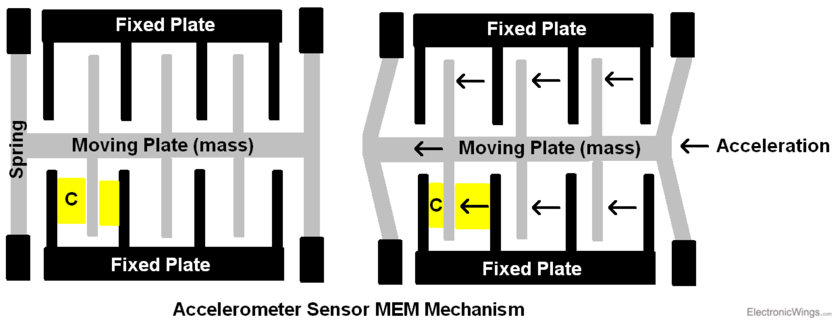


Figure 3.2: Accelerometer sensor working mechanism [22].

3.2.2. Gyroscope

The gyroscope sensor is a device capable of measuring the angular velocity of an object by means of Coriolis acceleration. The gyroscope's structure consists of an inner frame with a mass connected to springs, and an outer frame with so-called Coriolis sense fingers to sense displacement of the inner frame (Figure 3.3). The inner and outer frame are connected using springs. The structure is mounted to a disk that can rotate around an axis, shown in Figure 3.4.

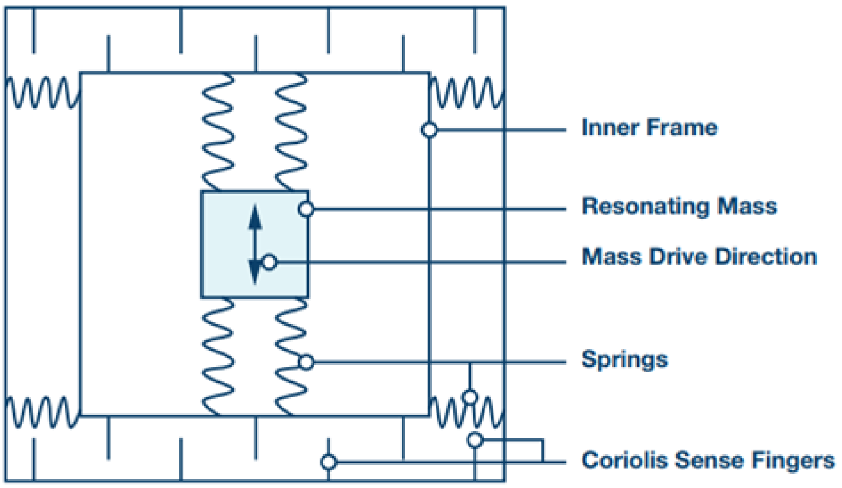


Figure 3.3: Schematic of the gyroscope's mechanical structure [23].

As the disk rotates and the mass moves, the mass and its frame experience Coriolis acceleration, sensed by the Coriolis sense fingers. The change in capacitance measured by the sense fingers determines the angular velocity. On the left of Figure 3.4, the mass moves from the center to the edge of the disk, compressing the springs on the left of the inner frame. This corresponds to an increase of angular velocity. The disk on the right of Figure 3.4 corresponds to a decrease in angular velocity [23]. To determine angular velocity in three directions, the gyroscope sensor has a disk with a resonating mass on each axis.

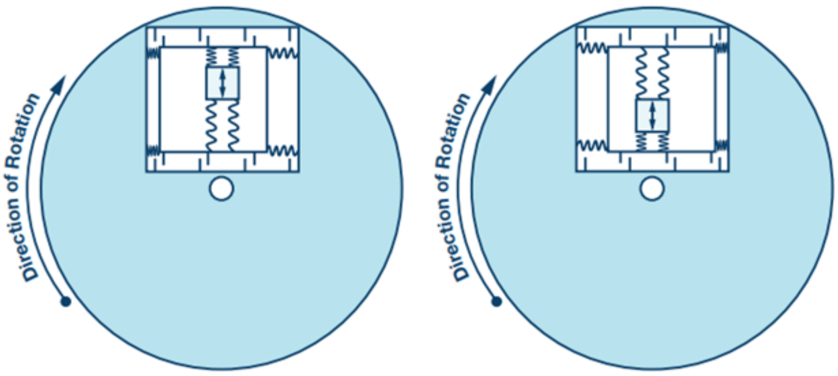


Figure 3.4: The disk and resonating mass are displaced laterally in response to the Coriolis effect [23].

3.2.3. Heart Rate Sensor

The Samsung Galaxy Watch4 determines the heart rate of the smartwatch wearer using the PPG (photoplethysmogram) sensor. The sensor makes use of low-intensity infrared light traveling through the tissues of the skin. Blood absorbs light better than the surrounding tissues, making it possible to detect a change in blood flow. The voltage signal transmitted by the sensor is proportional to the quantity of blood flowing through the blood vessels. Every heartbeat temporarily increases the blood flow, so the heart rate can be determined from the voltage signal transmitted by the sensor [24]. The heart rate sensor on the smartwatch outputs the heart rate in beats per minute.

3.3. Smartwatch Application Design

The purpose of the smartwatch application is to read out the sensors on the smartwatch and send the data to a server where it will be stored. The users will then be able to download the recording and use it for their research. The application is written in the programming language Kotlin [25], this was chosen because of its easy integration with Android apps. The integrated design environment (IDE) Android Studio [26] is used for programming the app, this IDE is compatible with Android watches. The application is split into multiple parts, in Android Studio such different parts are called activities and you can easily switch between activities with, for example, a button. The smartwatch application will be split in a main activity for the user setup and home screen and a transmit activity for recording the data.

3.3.1. Main Activity

The main activity is what a user will see when starting up the app. During the first startup it will show a prompt asking the user to fill in a personal ID, as seen in Figure 3.6a. This ID is used to find the user data on the server. After filling your ID, the app will check if it has the permission required to access the heart-rate sensor data, if not the app will ask permission, shown in Figure 3.6b. After allowing access to the sensor data the main activity will look like Figure 3.6c, from here you can start a recording. The app is also able to add a label that indicates what type activity you are doing, which can be used as a true label to train and test a machine learning algorithm. It is also possible to change your user ID in the main activity. A diagram of the code in shown on the left of Figure 3.5.

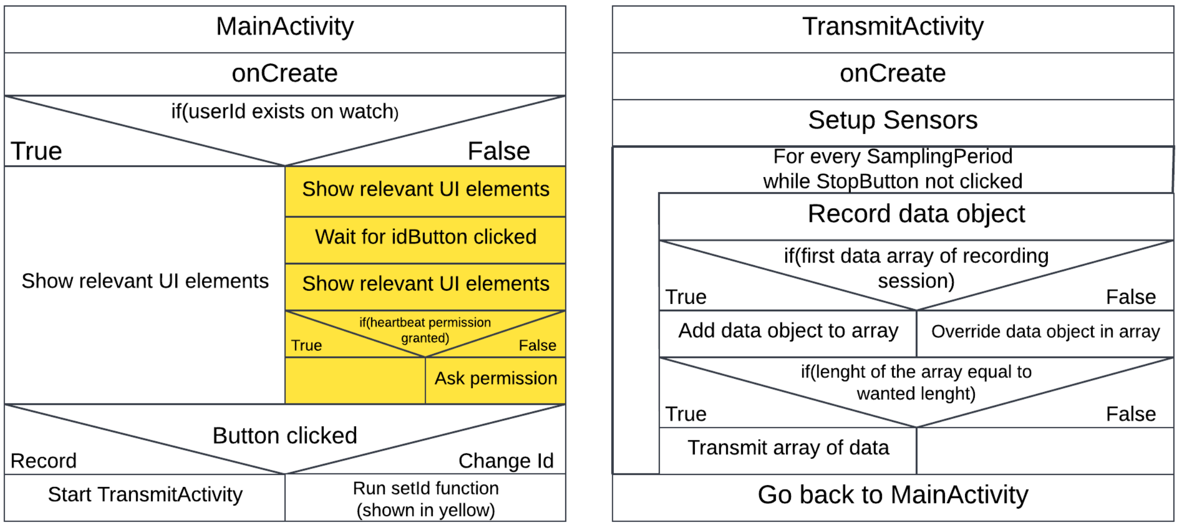
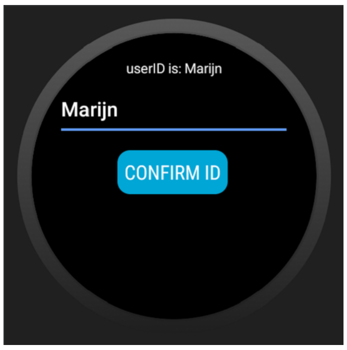

MainActivity
onCreate
if(userId exists on watch)
True
False
Show relevant UI elements
Show relevant UI elements
Wait for idButton clicked
Show relevant UI elements
if(heartbeat permission granted)
True
False
Ask permission
Button clicked
Record
Change Id
Start TransmitActivity
Run setId function (shown in yellow)
TransmitActivity
onCreate
Setup Sensors
For every SamplingPeriod while StopButton not clicked
Record data object
if(first data array of recording session)
True
False
Add data object to array
Override data object in array
if(lenght of the array equal to wanted lenght)
True
False
Transmit array of data
Go back to MainActivity

Figure 3.5: Nassi–Shneiderman diagram of the MainActivity and the TranmitActivity.

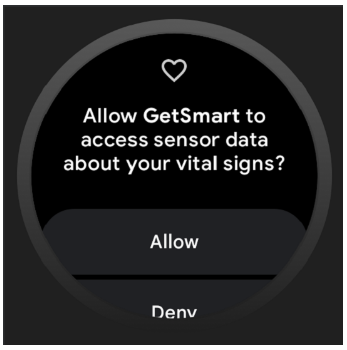
3.3.2. Transmit Activity

When clicking the record button, the app launches the transmit activity. This starts the recording and the app will send data to the database on the server. In this activity, the app only shows a simple GUI indicating it is recording and a stop recording button, see Figure 3.6d. The stop recording button, as the name suggests, stops the recording when clicked and makes the app return to the main activity. A diagram of the code in shown on the right of Figure 3.5.

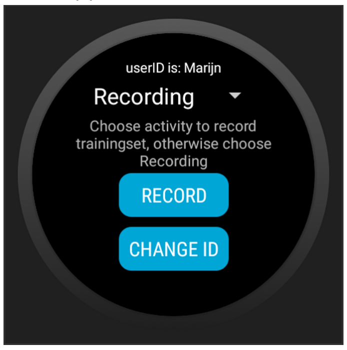

userID is: Marijn

Marijn

(a) User ID fill-in screen.


Allow GetSmart to access sensor data about your vital signs?

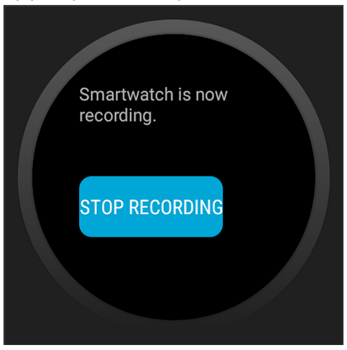
(b) Request sensor permission screen.


userID is: Marijn

Recording

Choose activity to record trainingset, otherwise choose Recording

(c) Home screen.


Smartwatch is now recording.

(d) Recording screen.

Figure 3.6: Smartwatch application screens.

3.3.3. Data

During a recording the app sends the following data to the server:

- Timestamp: time when data is recorded, accurate to the millisecond.
- User: the ID filled in by the user.
- AcceX, AcceY, AcceZ: x, y and z data of the accelerometer in m/s^2.
- GyroX, GyroY, GyroZ: x, y and z data of the gyroscope in rad/s.
- Heart rate: heart rate of the user in beats per minute.
- Token: the token is the timestamp of the start of the recording, this is used to distinguish between different recording sessions.
- Label: this is the activity the user is doing. If no activity is chosen, the label is 'Recording'.

The data is collected at a sampling frequency that can be easily changed depending on the needs of the researcher or user. For example at a sampling frequency of 10 Hz every 100 ms the app records all the data explained above in a data class. The app can then send that data directly or it can store multiple samples in an array and then send the array to the server. The data is sent as a JSON post request, which will be explained in Section 4.1.

3.3.4. Power Consumption

When realtime data is not needed, the sensor data does not have to be sent after every sample. Therefore the application should store multiple samples locally and send a package of samples after a fixed time interval. This reduces the number of requests that the smartwatch needs to make, which may lead to a lower power consumption over time, Chapter 5 will test whether this method has an influence on the power consumption.

Another possible factor influencing the power consumption of the smartwatch is the sampling frequency of the sensors. A lower sampling frequency might lead to lower power consumption, this hypothesis will also be tested in Chapter 5.

4 Web Server Design

The design choices made for implementing the web server will be discussed in this chapter. First the communication protocol used will be discussed. Afterwards, the different parts of the web server will be listed and the way of implementing each components will be discussed. Finally, a diagram of the web server will be shown.

The initial design of the web sever is done on an online integrated development environment (IDE) and web hosting service called Pythonanywhere [27]. Pythonanywhere was chosen because it offers the ability to program Python, web hosting and database management, all in one, through its website. It also offer analytical tools such as access logs and website statistics to the user. Later the web server is moved to a private PC, as the private PC server has a shorter response time, it provides more privacy and total control over the server is possible.

4.1. JSON

JSON is an abbreviation of JavaScript Object Notation. It is a format for writing HTTP requests and it is the most used format for communication between client and server. JSON has several request functions such as: **GET**, **HEAD**, **POST** and **CONNECT**. In this project only the **POST** request will be used. It is easily readable for humans and it supports sending data in arrays, which makes it a lot easier to debug during this project. To secure the user data that is sent in the POST request, a SSL certificate should be added to the server.

When the data is send to the specific page on web server through a JSON post request, it will be parsed into a Python dictionary and different values in the requests are accessed through the corresponding key. The web server accommodates for requests in the form of a JSON array of variable length where multiple samples of the data are being bundled together into a single request. This is done to reduce the sending frequency of the smartwatch, which might reduce its power consumption. The influences on the power consumption will be tested in Chapter 5 and the result of the tests will be discussed in Chapter 6.

Appendix D shows an example of a JSON object with the keys on the left of the colon and the corresponding values on the right of the colon, the values are stored in different attributes in the database. As seen in the example, a JSON request consists of two strings of characters for every key value pair, this means that every character is send as a byte. In the example the timestamp key and value consists of 32 characters, so that is already 32 bytes of data.

The example in Appendix D is a JSON request that was actually transmitted from the smartwatch to the web server, for this example the sampling frequency was set to 10 Hz and the sending frequency to 2 Hz. So every 0.5 seconds an array containing 5 samples, as in the example, was transmitted to the server. According to the log this 5 sample POST request has a size of 1165 bytes. From this, it is clear that the app will send a lot of data if it is recording for a longer period or at a higher sampling rate. Section 6.3 will talk about options to reduce the amount of data send to the server.

4.2. Components

The web server in this project has three components: front-end, back-end and database. The front-end is the interface that the users can see and interact with on the website, it is programmed using HTML. The back-end organizes and redirects the users between different pages on the website, it handles incoming data, connects to the database and fetches data for the user when requested. The micro web framework called Flask is chosen for this purpose. The database is where the data is stored and retrieved. The database management system MySQL is used for the implementation and management of the database.

4.2.1. Front-end

As mentioned in the introduction, the website server should allow researchers to access their own data easily, they should be able to see which watches are recording data and able to download specific sessions of data. The front-end of the website has several functionalities:

- The user can use a search bar to find their own recording sessions. If the database contains a large number of users, a search bar will make it easy to find recording sessions.
- The user can download a session data in a comma separated value (CSV) format, Figure 4.1 shows a diagram of the function.
- A scroll down menu is also present that lists all sessions present in the database. This menu gives quick access to recording sessions.
- The website can show a list of users that are actively sending data, this is done by comparing the timestamp of all data with the current time, the website will select data of less than 1 minute old and list their users.
- The website can show all of the unique users in the database, so the users can find their own sessions more easily.

Figure 4.2 shows a screen shot of the Homepage of the website.

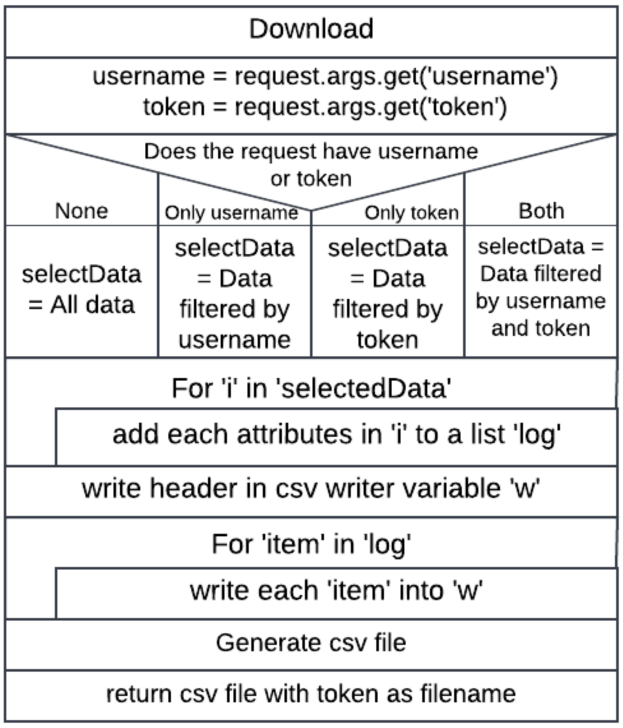
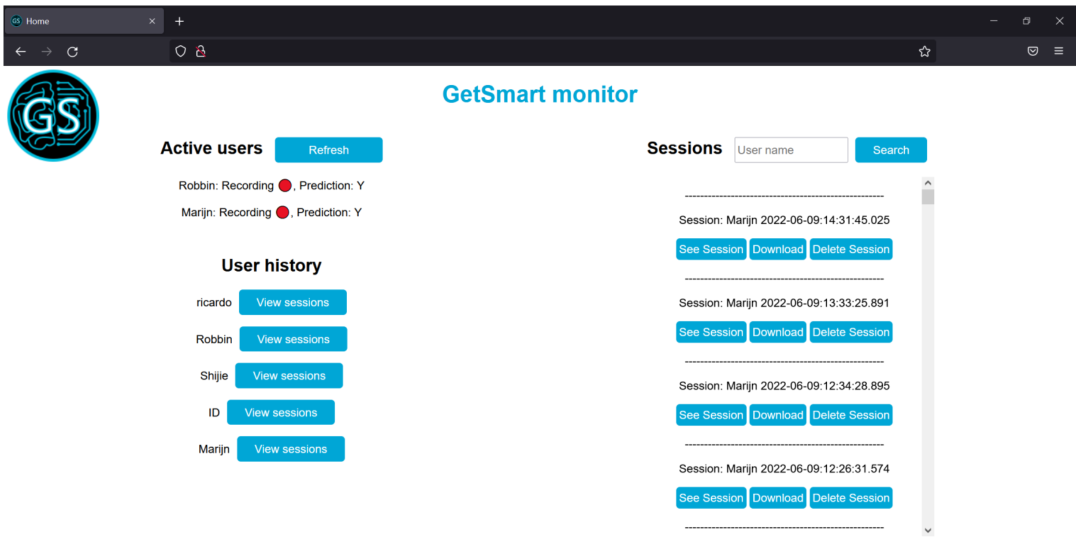

Download
username = request.args.get('username')
token = request.args.get('token')
Does the request have username or token
None
Only username
Only token
Both
selectData = All data
selectData = Data filtered by username
selectData = Data filtered by token
selectData = Data filtered by username and token
For 'i' in 'selectedData'
add each attributes in 'i' to a list 'log'
write header in csv writer variable 'w'
For 'item' in 'log'
write each 'item' into 'w'
Generate csv file
return csv file with token as filename

Figure 4.1: The Nassi-Shneiderman diagram of the download function.



GetSmart monitor

Active users

Robbin: Recording ●, Prediction: Y

Marijn: Recording ●, Prediction: Y

User history

ricardo	View sessions
Robbin	View sessions
Shijie	View sessions
ID	View sessions
Marijn	View sessions

Sessions

Session: Marijn 2022-06-09:14:31:45.025

Session: Marijn 2022-06-09:13:33:25.891

Session: Marijn 2022-06-09:12:34:28.895

Session: Marijn 2022-06-09:12:26:31.574

Figure 4.2: Home page of the website, with two active users sending data to the web server.

4.2.2. Back-end

Flask is a micro-framework written in Python that is used for designing websites. It has many advantages such as being simple to learn and use, structured to make even a large website with many pages to be easily manageable, it is also highly flexible when incorporating other Python scripts and functions.

When building a back-end of a website with Flask, the different pages are created by defining a function associated with the page and mapping the function to a URL extension of the main page. The functions return a front-end to the user by calling templates and passing arguments to it. The template can be written in various different programming or markup languages, HTML is used in this project.

4.2.3. Database

MySQL is a relational database management system (RDBMS). As with most of other existing RDBMSs it is programmed using Structured Query Language (SQL) which is the standard programming language for RDBMSs. The main advantages of MySQL are it is [28]:

- Easy to learn and use.
- Free and open source.
- Very fast, reliable, and scalable.
- Widely popular and has been proven to be able to handle massive volumes of user data, as it is used by some of the biggest tech companies such as Twitter and Wikipedia.

Data stored in a RDBMS is organized into rows (tuples) and columns (attributes), making samples stored in this way easily accessible by filtering it using its attributes, the samples can then be updated by a few simple lines of commands. Figure 4.3 gives a visual description of the database. The attributes in the database in this project are the same as the ones listed in Section 3.3.1.

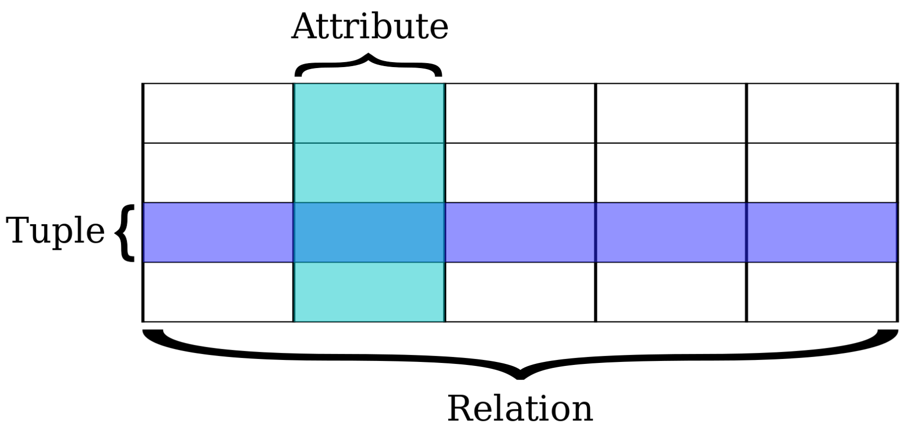


Figure 4.3: Visualization of the database structure [29], with each row being a sample in the recording, different values within a sample such as user id, timestamp and sensor data points are stored in the attributes within the row.

4.3. Diagram

A diagram of the web server is shown in Figure 4.4. The user starts at the Homepage, where a number of functions are embedded. They can go to the User Page to see a list of their sessions, the user can also examine the data points in the Session Page to verify the recording is done properly.

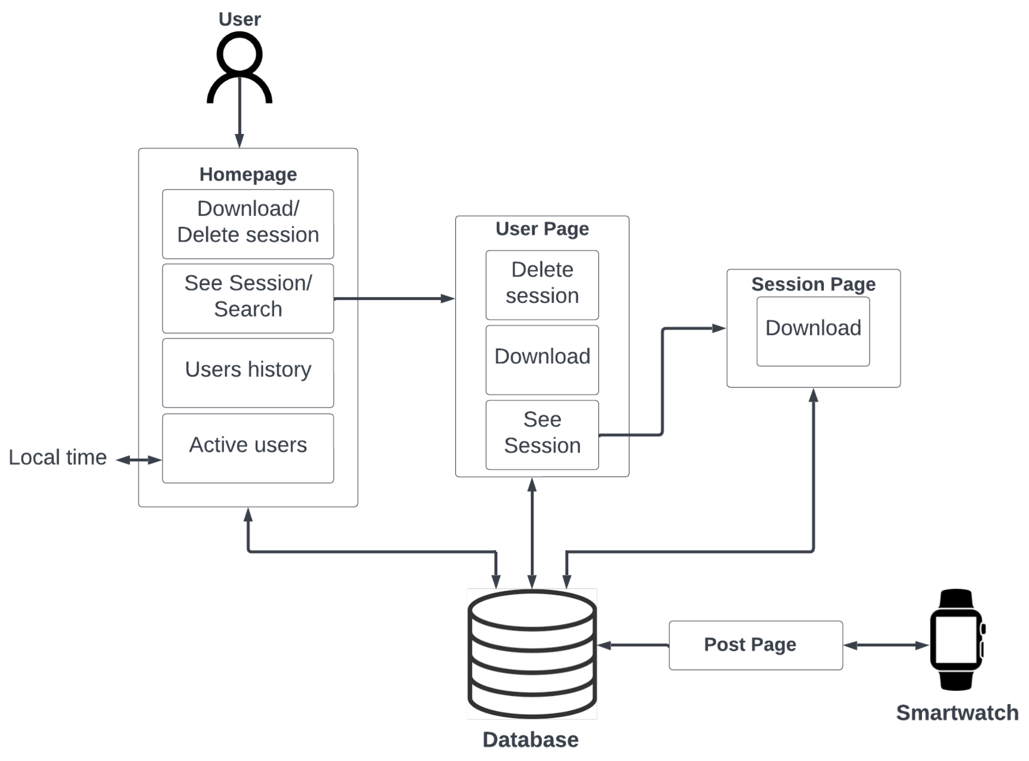


Figure 4.4: A diagram of the web server.

5 Test and Results

To train the machine learning model integrated in the platform, a data set with recordings of all activities has to be created. Furthermore, the time delay of the sensors on the smartwatch has to be tested to determine the maximum possible sampling frequency of the sensors. At last, the battery performance of the smartwatch is tested under different system settings.

5.1. Test Setup

5.1.1. Creating Data Set of Activities

The goal of this test is to create data sets for the other subsystem. This subsystem, created by the data classification subgroup, uses machine learning models to recognize activities performed by smartwatch users. Up to now, the machine learning models are trained using data sets publicly available. To obtain a high classification accuracy of activities on the platform, the models should be trained using data sets created on the platform.

The machine learning models should be able to record the following five activities, as stated in the requirements of the other subsystem [17].

1. Sitting
2. Standing
3. Walking
4. Jogging
5. Walking on stairs

Recordings in sessions of 5 minutes will be made for each of the activities, with as many participants as possible, in order to ascertain the robustness of the machine learning model. The sampling rate of the recordings will be 50 Hz, the sensors have different sampling frequencies which will be explained by section 5.2.1. All recordings are made with the smartwatch being worn on the left wrist.

5.1.2. Smartwatch Sensor Sampling Frequency

Wang et al. in [30] examined the frequency components of three activities: walking, falling and sitting down. They found a high-energy band around 35-40 Hz in the spectrogram for the walking activity. For the machine learning model integrated in the platform to recognize the walking activity, the sampling frequency of the sensors should thus be above 40 Hz. This way, the frequency band of interest is included in the data. The frequency bands of interest at the activities sitting down and falling found in the research by [30] are ignored as sitting down has a lower frequency band and falling is not included in this project.

To determine the highest possible sampling frequency of the sensors on the smartwatch, the delay between sensor updates has to be measured. The value of a sensor is requested using the function $OnSensorChanged()$. This function is called using two possible methods: after a fixed time period or whenever a sensor value changes. The second method is used to determine the sensor delay, the delay is measured by timing the period between two function calls. The delay is measured over a few minutes and the average delay is determined. The test results are discussed in Section 5.2.1.

5.1.3. Smartwatch Power Consumption

Several variations of tests need to be done to ascertain the different factors influencing the battery drain on the smartwatch during operation. Two smartwatches are available for testing during the project, the Samsung Galaxy Watch4 Classic and Samsung Galaxy Watch4, with a battery capacity of 361 mAh and 247 mAh respectively. The goal of the tests will be the following:

1. To test and establish a possible correlation between the different recording frequencies and the battery drain.
2. To test whether sending data less frequently and in an array is more power-efficient compared to sending every data samples individually.
3. To test whether the smartwatch can continuously record for at least 8 hours.

To test the first goal, the fully charged smartwatches will make recordings of 2 hour long using sampling frequencies of 5 Hz, 10 Hz, 20 Hz and 50 Hz. The samples will be send every second and the array size will be kept constant by padding the array to contain 50 samples to ensure that the test result will not be influenced by the array size of each transmission.

For the second test, the smartwatch will first send the data per individual sample to the server. Then the samples will be send in arrays of 10 samples, and also sending arrays of 100 and 10.000 samples is tested. For all of these test the sampling rate is set to 10 Hz.

The third test will be done by having the fully charged smartwatch record and send data continuously until the battery is completely empty, at a sampling frequency of 50 Hz, and transmitting every 10 seconds.

To verify the results all test will be done on two smartwatches, it should be noted that the smartwatches are not the exact same model, the second smartwatch has a lower battery capacity. Also a reference test will be done where the smartwatch application is set to idle mode, where only the screen will be on and no data will be recorded or sent to the server, in order to ascertain the base power consumption of the smartwatch.

5.2. Test Results

This section will discuss the results of the tests described in last sections. First, the measured delay of the sensors on the smartwatch is discussed and the highest sampling frequency is determined. Afterwards the results of the battery performance tests are discussed.

5.2.1. Sensor Delay Results

Using the testing method discussed in Section 5.1.2, the time delay of the accelerometer, gyroscope and heart rate sensor have been measured. The results are shown in Table 5.1. Based on this results, the highest sampling frequency without creating duplicate values can be determined. The gyroscope sensor has the highest update frequency with 107.87 Hz, more than twice the frequency of the accelerometer. Choosing this frequency as sampling frequency will lead to duplicate values read from the accelerometer. By choosing the sampling frequency equal to or lower than the update frequency of the accelerometer, the chance of duplicates is minimized. The update frequency of the heart rate sensor is much lower than the update frequency of the other sensors, but setting the sampling frequency equal to this frequency makes it impossible to detect human activity. Therefore a sampling frequency of 49.85 Hz, rounded to 50 Hz, is the maximum available sampling rate of this smartwatch application.

Table 5.1: Time delay of sensors on smartwatch.

	Accelerometer	**Gyroscope**	**Heart rate sensor**
Time delay	20.06 ms	9.27 ms	1003.82 ms
Corresponding frequency	49.85 Hz	107.87 Hz	0.996 Hz

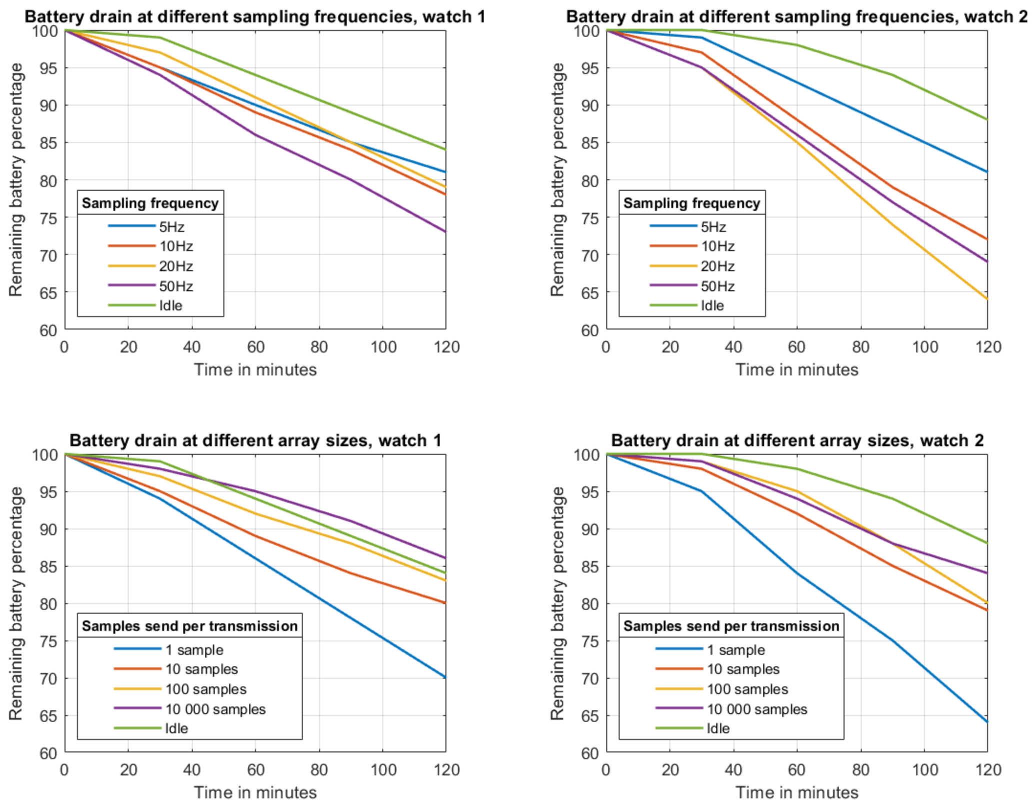


Figure 5.1: Battery drain over time at different sampling frequencies and different amounts of samples send, for two smartwatches. Watch 1 = Samsung Galaxy Watch4 Classic, watch 2 = Samsung Galaxy Watch4.

5.2.2. Battery Performances

Because every battery test took at least two hours, it was not possible to do all the tests on the same day and in the same environment. So the results might vary if the tests would be done again. Another source of variance in the results is the fact that the battery percentage reported by the smartwatch might not be completely accurate. There are different factors making it difficult to measure the remaining battery percentage of a lithium-ion battery, for example [31] names battery age and battery temperature. To improve the results the tests should be repeated, however the acquired results do show some significant trends.

As can be seen in Figure 5.1, when the smartwatch is idle, so not reading sensor values and not transmitting data, the battery is draining slower. So acquiring and sending sensor data to the server has an impact on the power consumption of the smartwatch. The magnitude of the impact on the power consumption depends on the factors tested, a higher sampling frequency results in a higher battery drain. This is what was expected, because at a higher sampling frequency the smartwatch has to read the sensors and process the acquired data more often. Furthermore, the number of samples sent per transmission significantly impacts the power consumption, especially transmitting every sample individually seems to drain the battery very fast. However the difference between sending every 100 samples or every 10.000 samples is not very significant.

Figure 5.2 shows the results of the long duration battery test. From the figure it is clear that the battery of watch 2 drains faster than the battery of watch 1. This is expected, as watch 2 has a smaller battery capacity. The test was conducted to see if it is possible to record for at least 8 hours continuously. The results show that depending on the smartwatch model it is possible, when transmitting every 10 seconds and sampling at 50 Hz.

5.2.3. Test Recommendations

From the tests results it can be concluded that a lower sampling rate is preferable, so a potential user of the product should consider what minimum sampling rate is sufficient for their application. For example for human activity recognition a sampling rate of 40 Hz should be enough [30], sampling at a higher frequency will result in a faster battery drain, but not a better activity recognition.

The sending frequency test shows that sending every sample individually causes a higher power consumption compared to sending data less frequently and in an array, it also does not have much benefit, since real-time data acquisition is not required for making recordings. Furthermore, sending the data once every 10 seconds is very near real-time, so for most applications transmitting in an arrays is better.

The battery of the first watch died after just under 10 hours. The battery of watch 2 was drained after 7 hours and 39 minutes. With some optimizations, such as sampling at a lower frequency, it maybe possible to extend the battery life to 8 hours. Section 6.3 also discusses some other modifications to the smartwatch application that should reduce the power consumption of the application.

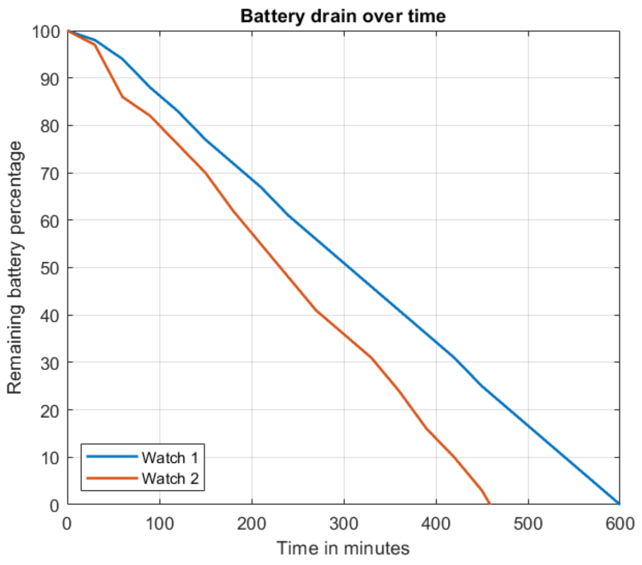


Figure 5.2: Battery drain over time while transmitting data, for two smartwatches. Watch 1 = Samsung Galaxy Watch4 Classic, watch 2 = Samsung Galaxy Watch4.

6 Conclusion and Discussion

6.1. Conclusion

In this thesis the design and implementation of a smartwatch application and web server are discussed. The purpose of the smartwatch application is to record and transmit sensor data to the server, the purpose of the server is to store the data and to make it available for the user on a website. The data is recorded from the accelerometer, gyroscope and heart rate sensor on the smartwatch.

Reflecting back on the group-specific program of requirements for the data acquisition group as listed in Section 2.2, all of the mandatory requirements are met. The smartwatch application and server are both fully functional. Smartwatch users can record activities and indicate the activity performed in the app. The server can receive the data and store them in recordings and the active users and recordings are shown on the website of the server.

The only unmet requirements were the trade-off requirements stating that the smartwatch application should acquire data from the ECG and PPG sensors and the application should continue running in the background.

The reason for not implementing these trade-off requirements is discussed in Section 6.2. All in all, the project is considered a success, because the prototype platform and the smartwatch application, together with the machine learning model developed by the Data Classification group, are able to meet all of the general requirements listed in Section 2.1.

6.2. Technical Challenges

Many challenges were encountered during this project, some of them are discussed in this section:

- One of the problems was that the server response time exceeded the sending delay of the smartwatch application. A slow response was initially not considered as an obstacle as the platform does not need to operate in real-time, but during some of the testing sessions, the backlog of the server became so severe that the server stopped responding to new requests. The culprit was later found to be a filter function that was called during each request in order to prevent duplicate data from being stored in the database, which occupied too much of the CPU resources. By removing the function, the server response time issue was solved.

- Another issue is that only a selected number of sensors are used during this project, namely the accelerometer, gyroscope, and heart rate sensor in beats per minutes. Other raw sensors data such as PPG and ECG data were not made available by the smartwatch manufacturers due to the General Data Protection Regulation. In order to gain access to these sensors, a lengthy review process must be done. Thus it was decided that these sensors would not be included for this project.

- The smartwatch application would crash if internet connection was temporarily lost during a recording session. In order to fix this issue, the smartwatch application needs to store some of the data locally on the smartwatch. But in the test setting of the prototype, it is assumed that the connection will be stable, thus this issue was not considered urgent enough to be fixed.

- If the user exits the application to the main menu of the smartwatch, or switches to another app, the smartwatch will not continue to send data while the 'GetSmart' application is in the background. But in a controlled testing environment the smartwatch does not do anything other than running the 'GetSmart' app, so this issue was ignored as it is not critical to the prototype.

6.3. Recommendations

Several other features need to be considered if the platform is aimed to be made commercially available. However, for the purpose of making a functioning prototype in this project, these features are shelved for the time being. Some of the features that have been considered but ultimately not implemented are listed below.

Login

One beneficial features is to include a type of authentication mechanism in order to secure the user data. A login screen can be implemented where the user needs to first register for an account, then a unique and randomized code can be generated on the website. The test subjects who will wear the smartwatches will have to input the code on the smartwatch application, in order to have their data stored in the database. Moreover, downloading and altering the data can only be done by first logging in with the associated account. This way different users will not have access to data belonging to another user, and it prevents unauthorized smartwatches from saving their data on the database and occupying the server resources.

Realtime Graph

Another shelved, but useful, feature was using a live updating graph to represent the data. A functional real-time graph was made by using an API called Highcharts, it can update the chart based on incoming sensor data. But this feature was later scrapped when the real time aspect of data collection was dropped, as it was considered unnecessary from the perspective of a researcher, who mostly needed the platform as a way to download their recorded data so they can process the data themselves. A function showing the currently active users was made instead.

Data Transmission

It would be beneficial if the amount of data needed to be sent could be reduced, this may also help with the battery drain. The most straight forward way of reducing the amount of data is by reducing the sampling rate, this directly reduces the amount of data that is send to the server, it also reduces battery drain (discussed in Section 5.2.2).

Every sample sent to the server contains contains 11 key value pairs, every character in the key as well as in the value is send as a byte. So the amount of data sent could be reduced by reducing the length of the keys or the values. Currently the keys are easily readable, but quite long. It would be possible to make the keys one letter of the alphabet, and translate this letter on the server side to a column in the database. This would reduce every key to one byte, reducing the size of the JSON by 45 bytes per sample. The size of the values can also be reduced by rounding them, the gyroscope data for example is a float with 6 decimal digits, where less decimal digits could be enough depending on the application of the data.

Another way to possibly make the data transmission better is by compressing the data, [32] discusses some possible compression techniques. Data compression would reduce the amount of data needed to be sent, however it would also make the app and server more complex, which might lead to more power consumption on the smartwatch. It should also be possible to compress the data in the database, reducing the overall size of the database.

Other Smartwatch Application Features

Some other features for the smartwatch applications were considered, such as keeping the application running while the screen is turned off in order to reduce the power consumption. A timer showing the recording time integrated with the smartwatch application would also make the application more convenient. But these functions were neglected in favour of other functions, as they are deemed not essential for a working prototype.

Bibliography

[1] B. Mohan, "Global smartwatch shipments hit a record high in q4 2021." `https://www.androidcentral.com/wearables/global-smartwatch-shipments-hit-a-record-high-in-q4-2021`, 2022. Accessed: 20-04-2022.

[2] Apple, "Specifications of the apple watch models." `https://www.apple.com/uk/watch/compare/`. Accessed: 21-04-2022.

[3] S. Abu-Alrub, M. Strik, F. D. Ramirez, N. Moussaoui, H. P. Racine, H. Marchand, S. Buliard, M. Haïssaguerre, S. Ploux, and P. Bordachar, "Smartwatch Electrocardiograms for Automated and Manual Diagnosis of Atrial Fibrillation: A Comparative Analysis of Three Models," *Frontiers in Cardiovascular Medicine*, vol. 9, 2022.

[4] T. Caillol, M. Strik, F. D. Ramirez, S. Abu-Alrub, H. Marchand, S. Buliard, N. Welte, S. Ploux, M. Haïssaguerre, and P. Bordachar, "Accuracy of a Smartwatch-Derived ECG for Diagnosing Bradyarrhythmias, Tachyarrhythmias, and Cardiac Ischemia," *Circulation: Arrhythmia and Electrophysiology*, vol. 14, no. 1, 2021.

[5] M. Nasarre, M. Strik, F. D. Ramirez, S. Buliard, H. Marchand, S. Abu-Alrub, S. Ploux, M. Haïssaguerre, and P. Bordachar, "Using a smartwatch electrocardiogram to detect abnormalities associated with sudden cardiac arrest in young adults," *EP Europace*, vol. 24, no. 3, pp. 406–412, 2021.

[6] T. Mishra, M. Wang, and A. e. a. Metwally, "Pre-symptomatic detection of covid-19 from smartwatch data," *Nature Biomedical Engineering*, vol. 4, pp. 1208–1220, Nov 2020.

[7] R. Liu, A. A. Ramli, H. Zhang, E. Henricson, and X. Liu, "An overview of human activity recognition using wearable sensors: Healthcare and artificial intelligence," *arXiv preprint arXiv:2103.15990*, 2021.

[8] R. Liu, S. A. Fazio, H. Zhang, A. A. Ramli, X. Liu, and J. Y. Adams, "Early mobility recognition for intensive care unit patients using accelerometers," *arXiv preprint arXiv:2106.15017*, 2021.

[9] M. S. Ryoo, B. Rothrock, C. Fleming, and H. J. Yang, "Privacy-Preserving Human Activity Recognition from Extreme Low Resolution," *Privacy-Preserving Human Activity Recognition from Extreme Low Resolution*, vol. 8, no. 1, pp. 1–8, 2017.

[10] I. Y. Jung, "A review of privacy-preserving human and human activity recognition," *International Journal on Smart Sensing and Intelligent Systems*, vol. 13, no. 1, pp. 1–13, 2020.

[11] M. Straczkiewicz, P. James, and J.-P. Onnela, "A systematic review of smartphone-based human activity recognition methods for health research," *npj Digital Medicine*, vol. 4, no. 1, 2021.

[12] R.-A. Voicu, "Human Physical Activity Recognition Using Smartphone Sensors," 01 2019.

[13] M. Z. Uddin, "Human activity recognition using wearable sensors, discriminant analysis, and long short-term memory-based neural structured learning," 08 2021.

[14] S. Mekruksavanich and A. Jitpattanakul, "Smartwatch-based human activity recognition using hybrid lstm network," in *2020 IEEE SENSORS*, pp. 1–4, 2020.

[15] S. Mekruksavanich and A. Jitpattanakul, "Sensor-based complex human activity recognition from smartwatch data using hybrid deep learning network," in *2021 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC)*, pp. 1–4, 2021.

[16] S. Nair, M. Kheirkhahan, A. Davoudi, P. Rashidi, A. A. Wanigatunga, D. B. Corbett, T. M. Manini, and S. Ranka, “Roamm: A software infrastructure for real-time monitoring of personal health,” in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–6, 2016.

[17] R. Cavalini, I. Hassan, T. Pouwels, “Facilitating healthcare using smartwatches.” June 2022.

[18] S. Lim, “Wear OS Share Surges on Samsung’s Highest Quarterly Smartwatch Shipments in Q3 2021.” `https://www.counterpointresearch.com/wear-os-share-surges-samsungs-highest-quarterly-smartwatch-shipments-q3-2021/`, 2021. Accessed: 4-06-2022.

[19] Apple, “Apple Watch-batterij.” `https://www.apple.com/nl/watch/battery/`. Accessed: 4-06-2022.

[20] T. Guide, “Samsung Galaxy Watch 4 review.” `https://www.tomsguide.com/reviews/samsung-galaxy-watch-4`, 2022. Accessed: 4-06-2022.

[21] Fitbit, “Fitbit Charge 5.” `https://www.fitbit.com/global/us/products/trackers/charge5`. Accessed: 4-06-2022.

[22] ElectricWings, “Adxl335 accelerometer module.” `https://www.electronicwings.com/sensors-modules/adxl335-accelerometer-module`. Accessed: 4-06-2022.

[23] J. Watson, “Mems gyroscope provides precision inertial sensing in harsh, high temperature environments, year = .” `https://www.analog.com/en/technical-articles/mems-gyroscope-provides-precision-inertial-sensing.html`. Accessed: 4-06-2022.

[24] Ambrose Soehn, “Measuring the heart – how do ecg and ppg work?.” `https://imotions.com/blog/measuring-the-heart-how-does-ecg-and-ppg-work/`. Accessed: 13-06-2022.

[25] JetBrains, “Kotlin programming language.” `https://kotlinlang.org/`. Accessed: 6-06-2022.

[26] Google Developers, “Android studio.” `https://developer.android.com/studio`. Accessed: 6-06-2022.

[27] PythonAnywhere LLP, “Pythonanywhere.” `https://www.pythonanywhere.com/`. Accessed: 6-06-2022.

[28] W3schools, “Introduction to mysql.” `https://www.w3schools.com/mysql/mysql_intro.asp`, 2022. Accessed: 30-05-2022.

[29] Wikicommons, “Relational database terms.” `https://commons.wikimedia.org/wiki/File:Relational_database_terms.svg`. Accessed: 30-05-2022.

[30] Wei Wang, Alex X. Liu, Muhammad Shahzad, Kang Ling, Sanglu Lu, “Understanding and Modeling of WiFi Signal Based Human Activity Recognition,” *ACM Digital Library*, p. 5, 2015.

[31] V. Pop, H. Bergveld, P. Notten, J. Op het Veld, and P. Regtien, “Accuracy analysis of the State-of-Charge and remaining run-time determination for lithium-ion batteries,” *Measurement*, vol. 42, no. 8, pp. 1131–1138, 2009.

[32] R. Shurtz, “JSON Compression: Alternative Binary Formats and Compression Methods.” `https://www.lucidchart.com/techblog/2019/12/06/json-compression-alternative-binary-formats-and-compression-methods/`, 2019. Accessed: 1-06-2022.

Appendices

A User Manual

This user manual explains the setup of the smartwatch application and server to record sensor data. In section A.2 the setup of the server is explained, in section A.1 the setup of the application is explained.

A.1. Server setup

The server is built using the Flask web-framework written in Python. The server can be hosted on a local machine or using a third-party Python web hosting service. The roadmap for both options are explained below.

A.1.1. Server on local machine

1. Download the server files from GitHub:
   `https://github.com/MarijnSluijs/GetSmart/tree/main/Server`.
2. Create MySQL database to store data. Documentation:
   `https://www.w3schools.com/python/python_mysql_create_db.asp`.
3. Add MySQL database to the flask file flask_app.py on line 20.
4. Run flask_app.py to start server. The server is now accessible on the local network.
5. To access the server from outside the local network, port 80 has to be forwarded on the router.

The server can now be accessed on the local network using the local IP address of the machine. To access the server from outside the network, the IP address of the router has to be used.

A.1.2. Server on third-party web hosting service

1. Download the server files from GitHub:
   `https://github.com/MarijnSluijs/GetSmart/tree/main/Server`.
2. Import server files to web hosting service.
3. Create MySQL database on web hosting service.
4. Add MySQL database to the flask file flask_app.py. on line 20.
5. Start hosting the server.

The server can now be accessed using the website URL given by the web hosting service. An image of the website is shown in Figure A.2.

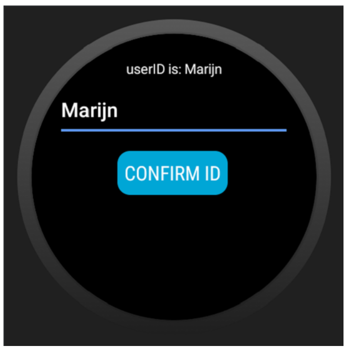
A.2. Smartwatch application setup

The smartwatch application is developed for the smartwatch Samsung Watch 4. To download the application onto the smartwatch, the following steps need to be done:

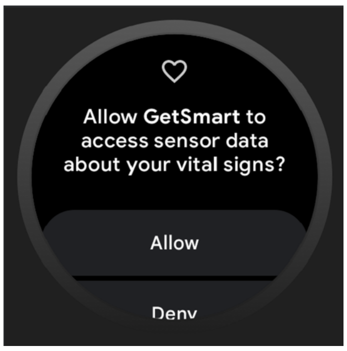
1. Download the application from Github:
   `https://github.com/MarijnSluijs/GetSmart/tree/main/Smartwatch%20app`.

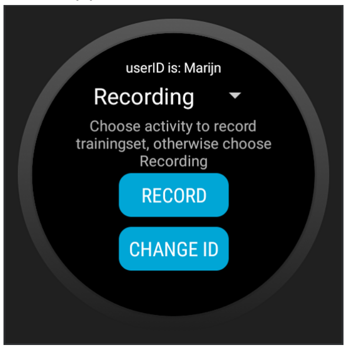
2. Unzip folder and open in Android Studio.
3. Link Samsung Watch 4 to Android Studio using Wifi or Bluetooth. Documentation: `https://developer.android.com/training/wearables/get-started/creating`.
4. Add server IP address/URL in the transmitData.kt file on line 28.
5. Click ”Run ’app’” in Android Studio to upload the application to the smartwatch.

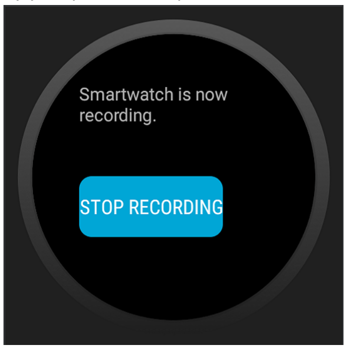
The smartwatch application is now ready to be used. After opening the application, a screen is shown to fill in the user ID (Figure A.1a). With the user ID, the recording sessions can be distinguished from other smartwatch users on the server. After filling in the user ID, a screen is shown asking for permission to access the sensors (Figure A.1b). For the application to function correctly, permission to the sensor has to be granted. After granting permission, the home screen is shown (Figure A.1c). On this screen the user can specify the type of activity performed in case the researcher wants to create a training data. Otherwise the user can leave it on ”Recording”, meaning that a data set will be recorded without the type of activity specified.



(a) User ID fill-in screen.


Allow GetSmart to access sensor data about your vital signs?
Allow
Deny
(b) Request sensor permission screen.


userID is: Marijn
Recording
Choose activity to record trainingset, otherwise choose Recording
RECORD
CHANGE ID
(c) Home screen.



(d) Recording screen.

Figure A.1: Smartwatch application screens.

A.3. Recording data

To send data, the smartwatch user has to click on the button ”RECORD” on the home screen. The data will be sent to the server where it is stored. On the website of the server the active users, user history and recording sessions are shown. From here the smartwatch wearers can be monitored and recordings can be downloaded.

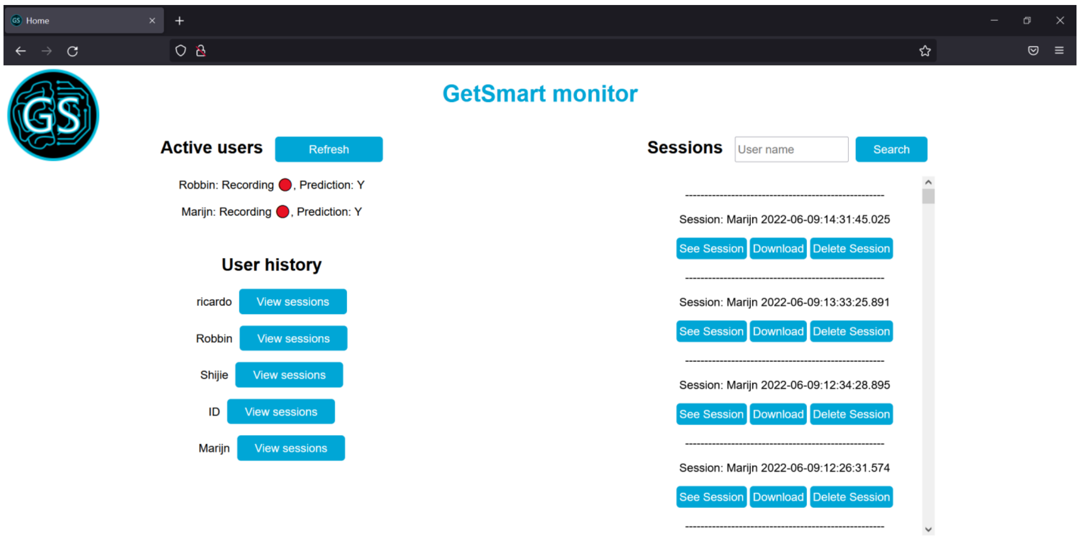

Home
GetSmart monitor
Active users
Refresh
Robbin: Recording ●, Prediction: Y
Marijn: Recording ●, Prediction: Y
User history
ricardo
View sessions
Robbin
View sessions
Shijie
View sessions
ID
View sessions
Marijn
View sessions
Sessions
User name
Search
Session: Marijn 2022-06-09:14:31:45.025
See Session
Download
Delete Session
Session: Marijn 2022-06-09:13:33:25.891
Session: Marijn 2022-06-09:12:34:28.895
Session: Marijn 2022-06-09:12:26:31.574

Figure A.2: Website of the server.

B Smartwatch Choices

Table B.1: Sensor availability and price of various smartwatches.

	Accelerometer	Gyroscope	PPG sensor	ECG sensor	Price[1]
Apple Watch 7	Yes	Yes	Yes	Yes	€379
Samsung Watch 4 Classic	Yes	Yes	Yes	Yes	€180
Fitbit Charge 5	Yes	Yes	Yes	Yes	€150
Noise ColorFit Pulse	Yes	Yes	Yes	No	€60
Fossil Gen 6	Yes	Yes	Yes	No	€230
Amazfit GTR3	Yes	Yes	Yes	No	€160
Garmin Vivoactive 4	Yes	Yes	Yes	No	€260
Xiaomi Mi Watch	Yes	Yes	Yes	No	€100
Imoo Watch Phone Z6	Yes	Yes	No	No	€260

[1]Prices are subject to change

C
Battery tests results

Table C.1: Battery percentage of smartwatch 1 after 120 minutes during multiple tests

Sampling Frequency (Hz)	**30 minutes**	**60 minutes**	**90 minutes**	**120 minutes**
5	95%	90%	85%	81%
10	95%	89%	84%	78%
20	97%	91%	85%	79%
50	94%	86%	80%	73%
Samples send per transmission				
1	94%	86%	78%	70%
100	97%	92%	88%	83%
10000	98%	95%	91%	86%
Idle	99%	94%	89%	84%

Table C.2: Battery percentage of smartwatch 2 after 120 minutes during multiple tests

Sampling Frequency (Hz)	**30 minutes**	**60 minutes**	**90 minutes**	**120 minutes**
5	99%	93%	87%	81%
10	97%	88%	79%	72%
20	95%	85%	74%	64%
50	95%	86%	77%	69%
Samples send per transmission (10 Hz)				
1	95%	84%	75%	64%
100	99%	95%	88%	80%
10000	99%	94%	88%	84%
Idle	100%	98%	94%	88%

D
JSON

D.1. JSON example

```
[{
    "acceX":-1.1540052,
    "acceY":1.7621324,
    "acceZ":9.600749,
    "bpm":"86.0",
    "gyroX":-0.0061086523,
    "gyroY":0.0,
    "gyroZ":0.010995574,
    "label":"Recording",
    "timestamp":"2022-06-01:11:20:13.743",
    "token":"2022-06-01:11:19:42.631",
    "user":"ID"
},
{
    "acceX":-1.0223241,
    "acceY":1.6448165,
    "acceZ":9.548077,
    "bpm":"86.0",
    "gyroX":0.02565634,
    "gyroY":-0.0036651916,
    "gyroZ":0.018325957,
    "label":"Recording",
    "timestamp":"2022-06-01:11:20:13.844",
    "token":"2022-06-01:11:19:42.631",
    "user":"ID"
},
{
    "acceX":-0.9792285,
    "acceY":1.6591818,
    "acceZ":9.495404,
    "bpm":"86.0",
    "gyroX":-0.002443461,
    "gyroY":0.04886922,
    "gyroZ":-0.03176499,
    "label":"Recording",
    "timestamp":"2022-06-01:11:20:13.942",
    "token":"2022-06-01:11:19:42.631",
    "user":"ID"
},
{
    "acceX":-1.1803415,
    "acceY":1.6208745,
    "acceZ":9.476251,
    "bpm":"86.0",
    "gyroX":-0.010995574,
    "gyroY":0.018325957,
    "gyroZ":0.052534413,
    "label":"Recording",
    "timestamp":"2022-06-01:11:20:14.043",
    "token":"2022-06-01:11:19:42.631",
    "user":"ID"
},
{
    "acceX":-1.0773908,
    "acceY":1.6591818,
    "acceZ":9.586384,
    "bpm":"86.0",
    "gyroX":-0.008552114,
    "gyroY":0.05497787,
    "gyroZ":-0.02443461,
    "label":"Recording",
    "timestamp":"2022-06-01:11:20:14.143",
    "token":"2022-06-01:11:19:42.631",
    "user":"ID"
}]
```