

# Design Project

## Wearable Breathing Trainer

University of Twente

19/04/2024

Authors:

Guido Baels (s2676478)

Daan Schram (s2692759)

Simeon Kaishev (s2726815)

Valerijs Farbtuhs (s2794764)

Kipras Klimkevicius (s2735210)

Matthias van der Most (s2852934)

Supervisor:

Randy Klaassen

# Abstract

Respiratory disorders such as asthma and dysfunctional breathing, which are common in childhood, are often treated by pediatric physiotherapists, who assign breathing exercises to the patients and monitor their progress. The main problem with this approach is the need for frequent appointments, which are needed to track patients progress and provide feedback during the training. To address this problem the system for managing remote breathing trainings was introduced by our clients.

Our team has developed a high-level prototype of such a system called the Wearable Breathing Trainer. It consists of a wearable vest for remote trainings, in addition to a mobile application for physiotherapists. The wearable provides haptic feedback to the patients during unsupervised trainings, while also recording training data. Moreover, the app allows physiotherapists to customize haptic feedback settings, retrieve and analyze the data of past trainings. The connection between these two components was implemented through the use of Bluetooth Low Energy (BLE), with delta encoding to quicken the sending of the data. We also used data science techniques for data processing in the backend, to calculate relevant metrics for the trainings.

Throughout the development we worked with our clients to elicit and validate requirements. As a result, we came up with a system that can be used to demonstrate how physiotherapists can treat respiratory disorders more effectively by making the remote trainings viable. After performing user testing, our team and the clients are satisfied with the developed product, and we are confident that it can be used as a strong basis for future research and development in that domain.

# Table Of Contents

<b>Abstract.....</b>	<b>2</b>
<b>Table Of Contents.....</b>	<b>3</b>
<b>1. Introduction.....</b>	<b>6</b>
1.1 Context.....	6
1.1.1 Condition.....	6
1.1.2 Treatment.....	6
1.2 Goals.....	6
1.3 Outline.....	6
<b>2. System proposal.....</b>	<b>8</b>
2.1 Application.....	8
2.2 Wearable.....	8
2.3 Communication.....	9
2.4 Conclusion.....	9
<b>3. Requirement Specification.....</b>	<b>10</b>
3.1 Must.....	10
3.2 Should.....	11
3.3 Could.....	12
3.4 Won't.....	13
3.5 Conclusion.....	13
<b>4. System Design.....</b>	<b>14</b>
4.1 Platform.....	14
4.2 Frontend.....	15
4.2.1 Home page.....	15
4.2.2 Add Patient Page.....	17
4.2.3 Profile Page.....	18
4.2.4 Patient App.....	19
4.2.5 Extra Features.....	20
4.3 Backend.....	21
4.3.1 Application Programming Interface (API).....	21
4.3.1.1 Account.....	21
4.3.1.2 Patient.....	22
4.3.1.3 Physiotherapist.....	22
4.3.1.4 Admin.....	22
4.3.1.5 Training.....	23
4.3.1.6 Device Settings.....	23
4.3.1.7 Treatment Plan.....	23
4.3.2 Email.....	24
4.3.3 Calculating breathing metrics.....	25

4.3.3.1 Peak and through detection.....	26
4.3.3.2 Metrics calculations.....	26
4.4 Wearable.....	27
4.4.1 Microcontroller.....	27
4.4.2 Breathpal Sensor.....	28
4.4.3 Processing of Sensor readings.....	28
4.4.4 Vibration motors.....	29
4.4.5 Storage.....	29
4.4.6 Power supply.....	29
4.4.7 Clock module.....	30
4.5 Connection.....	30
4.5.1 Wi-Fi.....	30
4.5.2 BLE.....	30
4.6 Conclusion.....	33
5. Testing Plan.....	34
5.1 Frontend.....	34
5.1.1 Frontend Tests.....	34
5.1.2 Frontend Testing Results.....	34
5.2 Backend.....	34
5.2.1 Backend Tests.....	34
5.2.2 Backend Testing Results.....	35
5.3 Wearable.....	36
5.3.1 Testing the Wearable.....	36
5.3.2 Wearable Testing Results.....	36
5.4 User testing with clients.....	37
5.5 Conclusion.....	37
<b>6. General Discussion.....</b>	<b>38</b>
6.1 Team Evaluation.....	38
6.1.1 Planning.....	38
6.1.2 Responsibilities.....	39
6.2 Did we meet the goal?.....	40
6.2.1 Finished demonstrator.....	40
6.3 Limitations.....	40
6.3.1 BLE.....	41
6.3.2 Patient App.....	41
6.3.3 Security and Privacy.....	41
6.3.4 Other limitations.....	41
<b>6.4 Future planning.....</b>	<b>42</b>
6.4.1 Wearable improvements.....	42
6.4.2 Patient app improvements.....	42
6.4.3 General app improvements.....	42

6.4.4 ESP Casing.....	43
6.4.5 Different Microcontroller.....	43
<b>7. Conclusion.....</b>	<b>44</b>
<b>8. Acknowledgements.....</b>	<b>45</b>
<b>9. References.....</b>	<b>46</b>
<b>Appendix A: Additional Diagrams.....</b>	<b>47</b>
<b>Appendix B: Wearable testing.....</b>	<b>51</b>
Testing SD card reader.....	51
Testing RTC module.....	51
Testing Breathpal.....	51
<b>Appendix C: User testing results.....</b>	<b>52</b>

# 1. Introduction

## 1.1 Context

### 1.1.1 Condition

Dysfunctional breathing is a condition that adults can suffer from, but lately is increasingly being recognized as a significant health issue among children. M.E. van Schaik & S.A.S. Pichon (2023, under review) says that “Dysfunctional breathing occurs when the normal breathing pattern is disrupted, leading to irregular, shallow, or rapid breathing, which can cause various symptoms such as fatigue, chest pain, shortness of breath and difficulty concentrating. It can also exacerbate underlying conditions such as asthma, anxiety, and sleep disorders.” It is not known exactly how many children suffer from dysfunctional breathing, but it is estimated to be over 10% of all children (M.E. van Schaik & S.A.S. Pichon, 2023, in preparation). In order to prevent long-term health consequences and improve quality of life, it is essential that these patients get treated as soon as possible at a young age.

### 1.1.2 Treatment

To treat this condition, the pediatric physiotherapist (PP) has to guide the patient in training sessions to help them breathe through their abdomen, as opposed to the thorax (M.E. van Schaik & S.A.S. Pichon, 2023, in preparation). After the physiotherapist has shown the patient how the breathing exercises work, the patient is supposed to be able to work on these exercises at home so they can improve over the week. Then at the next meeting with the physiotherapist, progress can be monitored and new goals can be set. However, the PPs have a hard time monitoring how and if the children are doing them, which in turn hinders the progress of the treatment.

## 1.2 Goals

The goal of this project is a demo solution to the problem introduced in the previous section. We designed a system consisting of a smart wearable, which uses haptic feedback to support children doing the exercises on their own; and an app for PPs to track the treatment of their patients. While the target audience of the treatment are 8-18 year olds, the main stakeholder for this project are the pediatric physiotherapists, as it aims to assist them in treating patients. The purpose of this system is to aid physiotherapists in treating children with dysfunctional breathing, by creating and monitoring the training programs for patients.

## 1.3 Outline

Since dysfunctional breathing is such a problem for many children, it is of the utmost importance that there are sufficient treatment capabilities to try and reduce the amount of affected people. This is where the Wearable Breathing Trainer (WBT) comes in. In chapter 2, the proposal is

written in more detail and it will become clear how the wearable and the application work. After that, chapter 3 will elaborate on the requirement specification process. Following the requirements is the system design in chapter 4, where the implementation of the requirements is described and the choices we made to realize our goals are discussed. Furthermore, chapter 5 consists of the testing plan and how it is made sure that the delivered product works correctly. Finally, in chapter 6, the design project is reviewed in a general discussion. There will also be a look into how the WBT can be improved for future purposes. Final conclusions will be made in chapter 7 with a thank you to the people that helped us in chapter 8 and our references in chapter 9.

## 2. System proposal

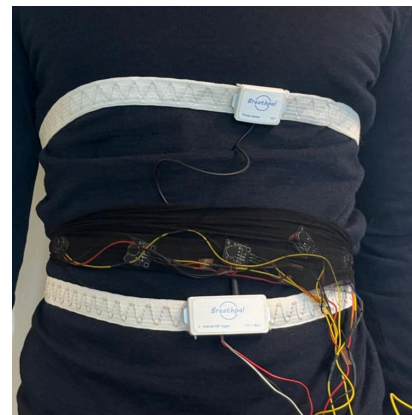
As mentioned in the previous chapter, the system will consist of 2 main components: the wearable, which will help patients do their breathing exercises, and the app, which will let PPs track patients' progress. In this section we will go into detail about how each of the individual components will function and the communication between them.

### 2.1 Application

After interviewing our client and performing requirement elicitation we have discovered the main requirements for the application (which are discussed in chapter 3). One of the crucial ones is that the application should be mobile, because that would be convenient for the physiotherapists. Our clients had already done research on this by asking various physiotherapists what devices they would use for the app. Most of them had an IOS device, so it is of importance that the mobile application works for IOS. The setting of training parameters for the wearable should take place through an application installed on physiotherapists' phone. Next to that, It should allow the PP to manage their patients, examine their general progress, specific past trainings from the wearable. Furthermore, the app should have basic features such as a working authorization and login system, there should be an information page and there should be a way to view accounts of users that display their contact details, which can also be altered by the user.

### 2.2 Wearable

First of all, the Wearable Breathing Trainer consists of a wearable, which incorporates the Breathpal (Ben Bulsink, 2020), a respiration sensor using Respiratory Inductive Plethysmography (RIP). It allows us to track the expansion of the thorax and abdomen of the user as they breathe. The data from it is processed by a microcontroller, and based on it, real-time haptic feedback is applied using up to 5 vibration motors, placed on a horizontal line along the patient's abdomen. These sensors will provide data that can be used to view the breathing activity in the thorax area and the abdominal area, which are of importance to the physiotherapist. The aforementioned vibration are there to guide the patient during training. The wearable will also include wireless connectivity, in the form of Bluetooth, to allow the PP to modify training parameters based on a patient's needs.



**Figure 1** System prototype

Secondly, It is essential that the wearable is able to assist patients with their exercises without needing to connect to a mobile device, as they might not have access to one. This request by our clients meant that we had to change our proposal for the wearable in the following manner: The system will be controlled by a microcontroller. The microcontroller should interface with the



BreathPal sensor, by reading its abdomen and thorax expansion values through UART communication. It should then be able to process it and apply haptic feedback using the 5 vibration motors in real time. Since the system will function autonomously, it needs to be able to receive settings selected by the physiotherapist from the app, store them, and keep logs of all trainings, so the PP is able to see information on how the trainings went. It should also be able to track when trainings took place to further help PPs with their analysis.

## 2.3 Communication

Of course it is of the essence that the wearable and the app can communicate with each other, which our clients preferred to be done wirelessly. After considering advantages and disadvantages of the two wireless communication methods available on the microcontroller - Bluetooth Low Energy (BLE) and Wi-Fi, we have chosen BLE. We thought that it is more appropriate, since the use of wifi would require configuring network connection, which can be problematic for the patients. This way, the app can be connected to the wearable when both devices are in close proximity. Then, there should be the possibility to send over the settings of the patients that are set in the app and will then be stored on the wearable after it is sent over. Also, the wearable device should be able to send over data from the breathing exercises, so that these can be displayed in the app.

## 2.4 Conclusion

In conclusion, our system proposal consists of an application, a wearable, and the communication between the two.

## 3. Requirement Specification

During the first two weeks of the project, we allocated two meetings where the following requirements were elicited and prioritized (using MoSCoW prioritization), with what we thought would be the client's needs. After we finished these requirements, we sent them over to the clients, so we could get some feedback and adjust them. Then we had another meeting so we could discuss some more of the requirements and show some diagrams and the proposed design for the frontend of the application (Appendix A). That meeting eventually led to the list of requirements. Throughout the project, these requirements have slightly changed, adjusted and new ones were added after further meetings with the clients. The final list of requirements can be found below.

As for the MoSCoW prioritization, "must" requirements are the most important and the first requirements that will be focussed on. These will definitely be finished by the end of the project and preferably even sooner. After this come the "should" requirements. These are still quite important for delivering a nice product and so we also try to implement all of these before the end of the project. The "could" requirements are less important than "must" and "should", but are still valuable for the product and thus will be implemented if time allows it. Then there are "won't" requirements that have been suggested either by us or the client, but after discussion fell outside the scope of the project and are thus not going to be implemented.

### 3.1 Must

Functional requirements

- [mf1] As a Pediatric Physiotherapist(PP), I want to be able to wirelessly connect to the wearable device.
  - *This is needed for sending information over between the application and the wearable.*
- [mf2] As a PP, I want to change basic settings on the wearable.
  - [mf2.1] Set the duration of the exercise.
  - [mf2.2] Set general vibration strength (upper and lower bound).
  - [mf2.3] Set the type of vibration (during inhale, exhale, both).
  - [mf2.4] Select the number of motors used for haptic feedback (1, 3, or 5 motors).
    - *These settings will be saved on the wearable and used for each training session of the patient, until the settings are changed.*
- [mf3] As a PP, I want to be able to create an account for myself and log into it.
  - *This makes sure a PP is able to create an account for themselves, and start using the app.*
- [mf4] As a PP, I want to see the number and duration of performed trainings when I am connected to the wearable.

- *A PP wants to see how often their patients did trainings, and for how long.*
- [mf5] As a patient, I want to be able to use the wearable and perform training without the need to connect the wearable to an application.
  - *This was important, because the patients are often young children, so they will not always have access to a phone, which would mean they can't use an application to start the training and thus the wearable itself needs to have an option to do this.*
  - *Because of said requirement, we had to add more components like the RTC module, for tracking time, and the SD card module, for storing the data. It also affected our original idea of having BLE for live data tracking, to having BLE for sending over the entire training data packages.*
- [mf6] As a patient, I want to have haptic feedback while performing the exercise, of which the type is set by the PP.
  - *The haptic feedback is important for guiding the patient through the exercises.*
- [mf7] As a PP, I want to see the amount of chest vs abdominal breathing in a training.
  - *This is an important metric, since the goal is to have the patient breathe more through their abdomen instead of their chest.*

#### Non-functional requirements

- [mn1] It should be possible to establish a wireless connection between the application and the wearable within 1 minute, so the PP does not spend a lot of time on it.
  - *Connecting to the wearable is one of the primary functions, so it should not take too long*
- [mn2] The wearable should be self-contained, meaning that it shouldn't be connected to any other devices or peripherals, apart from sensors, vibrating motors and microcontroller.
  - *The wearable only has the microcontroller, two breath sensors and the vibration motors, it is important that no other peripherals can be connected*
- [mn3] The application should be available for the demonstration, but it does not have to be easily deployable to production.
  - *This is because the product is still in its early stages of development. That is why our clients explained how it will only be used for a demonstration for now.*
- [mn4] The application should be secured by using JWT tokens and role-based authentication.
  - *Security is important, and we want to be able to login as a patient or physiotherapist. We want to accomplish this by using JWT tokens alongside role-based authentication.*

## 3.2 Should

#### Functional requirements

- [sf1] As a PP or patient, I want to be able to request a new password.
  - *So that you can regain access after losing credentials.*

- [sf2] As a PP, I want to be able to create inactivated patient accounts.
  - *This is used to store data as belonging to patients. Patients can activate accounts and access their data themselves as well.*
- [sf3] As a PP, I want to see the thorax breathing in a simple graph when inspecting the data from past training.
- [sf4] As a PP, I want to see the abdomen breathing in a simple graph when inspecting the data from past training.
  - *These two requirements allow viewing of the most basic data without preprocessing.*
- [sf5] As a PP, I want to see the frequency of breathing when inspecting the data from the past training.
  - *This means the amount of breaths per minute, for a training in the past.*
- [sf6] As a PP, I want to see the duration of exhalation vs inhalation of a training.
- [sf7] As a PP, I want to see the frequency of breathing during training.
- [sf8] As a PP, I want to see the depth of breathing during training.
- [sf9] As a PP, I want to see the amount of movement during training.
  - *These requirements allow for more extensive analysis of the training, which allow for easier tracking of progress.*
- [sf10] As a PP, I want to have a tutorial explaining how to set up and use the wearable and app.
  - *This should make it easier for PPs to use the app.*

### 3.3 Could

#### Functional requirements

- [cf1] As a patient, I want to be able to activate my account and use it.
- [cf2] As a patient, I want to be able to wirelessly connect to the wearable device.
- [cf3] As a patient, I want to be able to start the training from the app.
  - *Extra functionalities for the patient app. These are not necessary, but could be useful.*
- [cf4] As a PP, I want to have a list of my patients.
- [cf5] As a PP, I want to be able to add patients to my patient list.
- [cf6] As a PP, I want to be able to remove patients from my patient list.
  - *These requirements give the PP control over their patient list, by showing them and giving the option to add or remove patients.*
- [cf7] As a PP, I want to be able to link settings to the patient, so I can update them remotely and new settings can be applied to the wearable when the patient connects to it through the app.
  - *These are used by the wearable during training.*
- [cf8] As a PP, I want to be able to see training data without connecting to the wearable itself.
  - *This would mean a PP can analyze data, even when the patient is not around.*
- [cf9] As a PP, I want to be able to set up a treatment plan for a patient.
  - [cf9.1] Set up a general description of the treatment plan.

- [cf9.2] Set up physical meetings.
- [cf9.3] Set up goals and their deadlines.
- *This gives the PP the ability to enter textual details of the treatment of a patient, so they do not have to register this anywhere else.*
- [cf9] As a PP, I want a QR code functionality to easily add patients.
  - *Scanning a QR code goes faster than entering long account codes.*
- [cf10] As a PP I want to be able to see if the patient was “faking” breathing through the abdomen during their exercise.
- [cf11] As a PP, I want the training to be reset if too much movement is detected, to ensure correct measurements.
  - *These requirements are related to the amount of movement to a patient, as this might influence training results.*
- [cf12] As a PP, I want to see the number of trainings that are missed.
  - *Useful information to track for a PP, so that they can talk to a patient if they need to train more.*
- [cf13] As a patient, I want the data on the SD card to be encrypted.
  - *Providing extra security.*

#### Non-functional requirements

- [cn1] The option of using the wearable without the app should be preserved even if the app is available.
  - *This means that patients can use the wearable, even without the app.*

### 3.4 Won't

#### Functional requirements

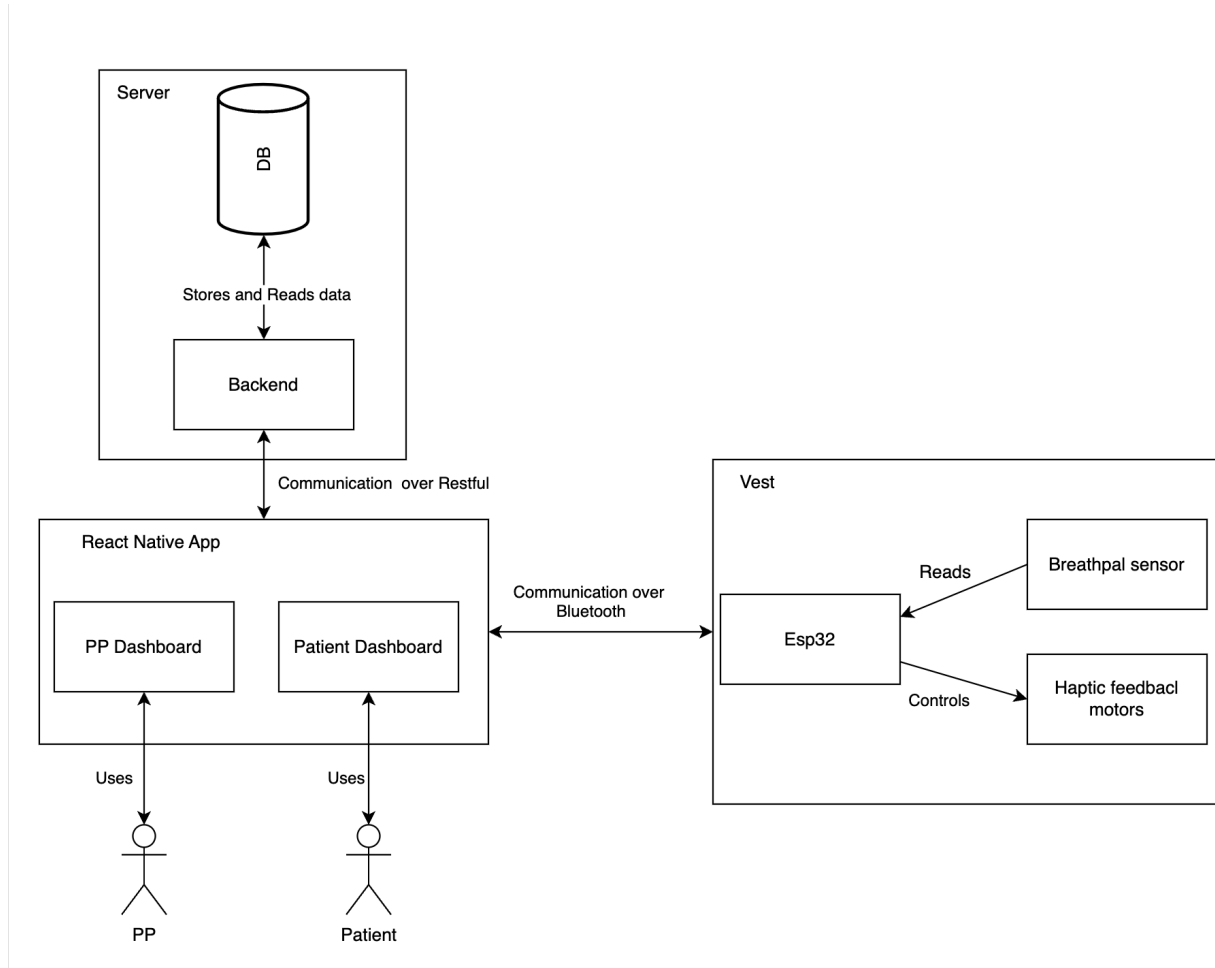
- As a patient, I want to chat with my PP.
- As a PP, I want to chat with my patients.
  - *A chat functionality is something which can be added, but does not have high priority.*
- As a patient or PP, I want to add pictures to my account.
  - *Profile pictures do not have high priority but can be included.*
- As a patient, I want a gamification element to be present in the training.
  - *This could make it more fun for a patient, but is a bit out of scope for this project.*
- As a product owner, I want to have a dedicated admin profile in order to monitor the app.
  - *An admin would be able to change all accounts, but this is not necessary.*

### 3.5 Conclusion

These requirements help us in setting clear goals for our project from the perspective of the user. In interviews with our clients, we developed a shared understanding of the scope of the project. The prioritization also allowed us to reflect on the feasibility of the project within the limited timeframe we had. In chapter 4 we discuss how we designed and implemented the system to fulfill these requirements.

## 4. System Design

In Figure 2 you can see a general overview of what we imagined the system would look like. In the sections that follow we will go into more detail on what the system looks like exactly and how we realized it.



**Figure 2** System overview

### 4.1 Platform

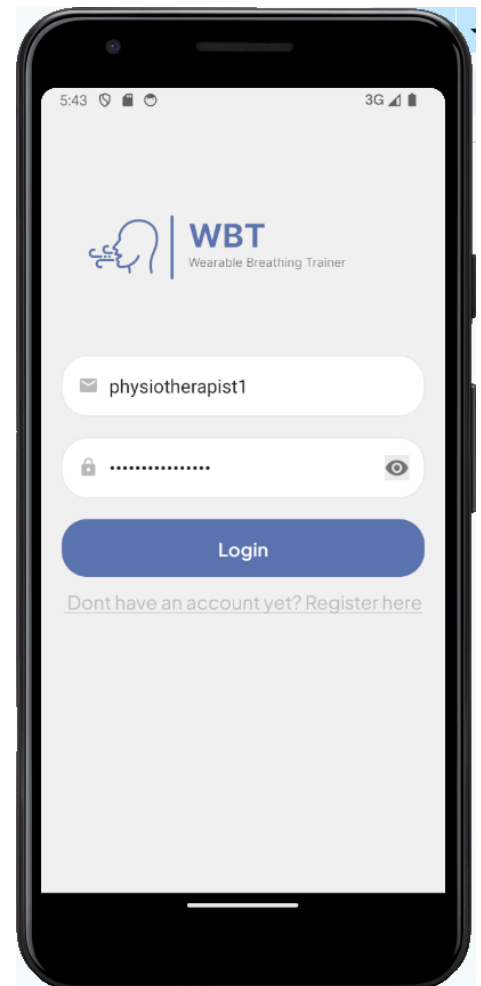
After forming all necessary requirements based on the needs of our clients, we have made the following design choices. First of all, for the design and implementation of the application, we use ReactNative. This is because our clients wanted the application to be available on mobile for physiotherapists. They especially wanted it to be available on IOS, since most of the physiotherapists they interviewed use this. ReactNative is useful for applications that are available on both Android and IOS, so after suggesting this to our clients they agreed this would be a good framework for the application. Next to that, one of our group members already had a lot of experience with React and also, we found an extensive course that could teach us the features needed for the application.

## 4.2 Frontend

Furthermore, we decided to split the application for the physiotherapist up into three pages that all have a separate function. We came to this decision, because there are three main functionalities that are important for this app. Namely, a way to display patient information and connect to a wearable, a way to add new patients and a way to see and alter personal details. Of course, before the physiotherapist is able to access any of the applications, there is a login page that asks for the credentials of the user.

### 4.2.1 Home page

After logging in, the physiotherapist will be directed to the home page (Figure 4 Home page). This page displays the most important features of the application, which is why we decided to make it the home page and the page that the therapist is automatically shown after logging in. It will either display a list of patients, or it will allow you to connect to a wearable, depending on which tab on the homepage the physiotherapist has selected. The list of patients shows the names of the patients, with a pop-up menu. It will display four options related to the patient. First of which is the profile of the patient. Here, the physiotherapist is able to see the contact details of a patient (Figure 5 Patient Profile Page). Next to that the device settings of the patient will be displayed here (Figure 5 Device settings), together with their account code that can also be seen as a qr code. The next option in the menu is the progress page for the patient (Figure 5 Training List). Here, the physiotherapist will be able to see a weekly overview of results on the top of the page. The physiotherapist can select different overviews for either the abdomen or thorax breathing, or the time the patient has spent on the exercises. This decision was made after feedback from our clients, who said these were the most important features that a physiotherapist would want to see as overall progress. Then, below the general overview there will be a list of individual trainings. Here, the physiotherapist is able to view more specific information of a training that is visualized in an easily understandable way (Figure 5 Training Data). For the design of the training details, it was decided to keep it very simple, because our clients recommended this, since a lot of the physiotherapists they interviewed struggled with the more complicated graphs. Now, the page consists only of the percentage of training completion, the averages for time per breath, breathing frequency, inhalation versus exhalation time and abdomen versus thorax breathing. If



**Figure 3** Log in screen

the physiotherapist scrolls down they can still see the more advanced graphs if they are interested in this, because we decided this should still be an option in the application. The third option for the dropdown is the device settings page for a patient. As the name implies, the physiotherapist can view, change and save the device settings for a certain patient. This includes the duration of the exercise, the amount of motors used, the minimum and maximum strength of the motors and the type of exercise, which can differ between inhalation, exhalation, or both. The fourth option will allow the physiotherapist to unlink the patient. Which can also be accessed by sliding the patient to the right.

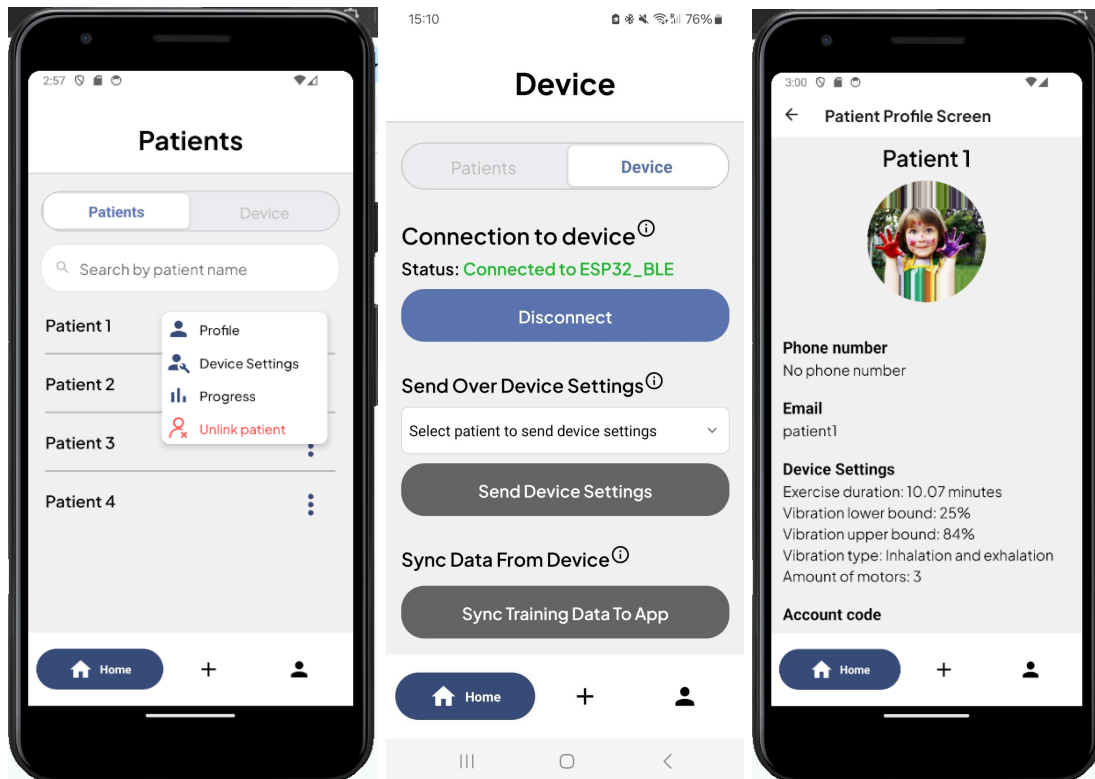
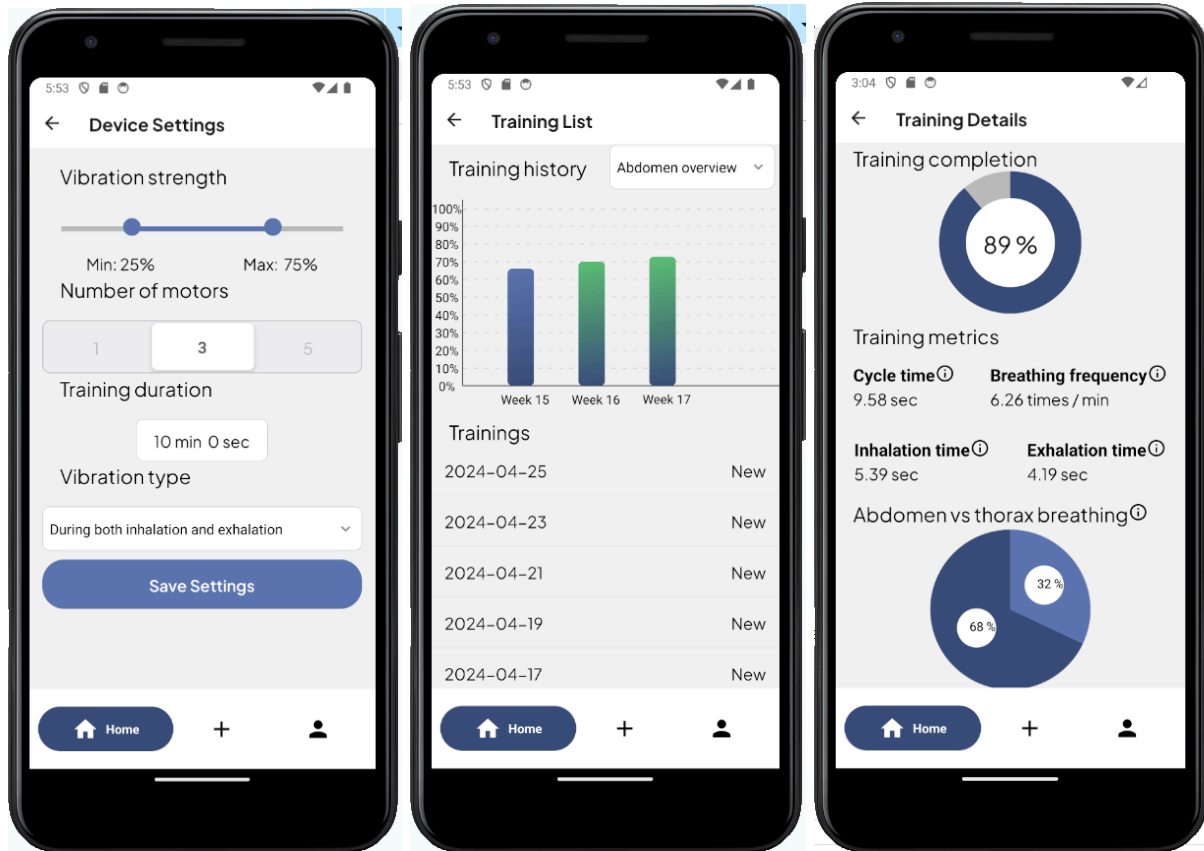


Figure 4 Home Screen, Device Page and Patient Profile Page

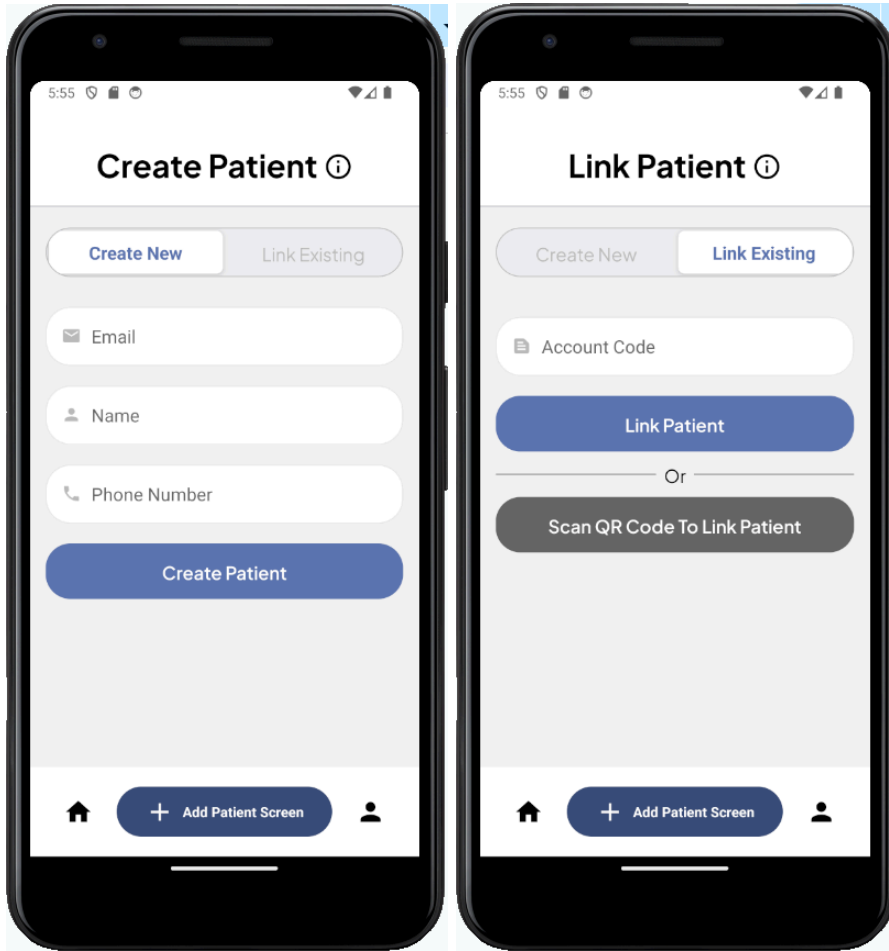




**Figure 5** Edit Device settings page, Training List, Training Data

#### 4.2.2 Add Patient Page

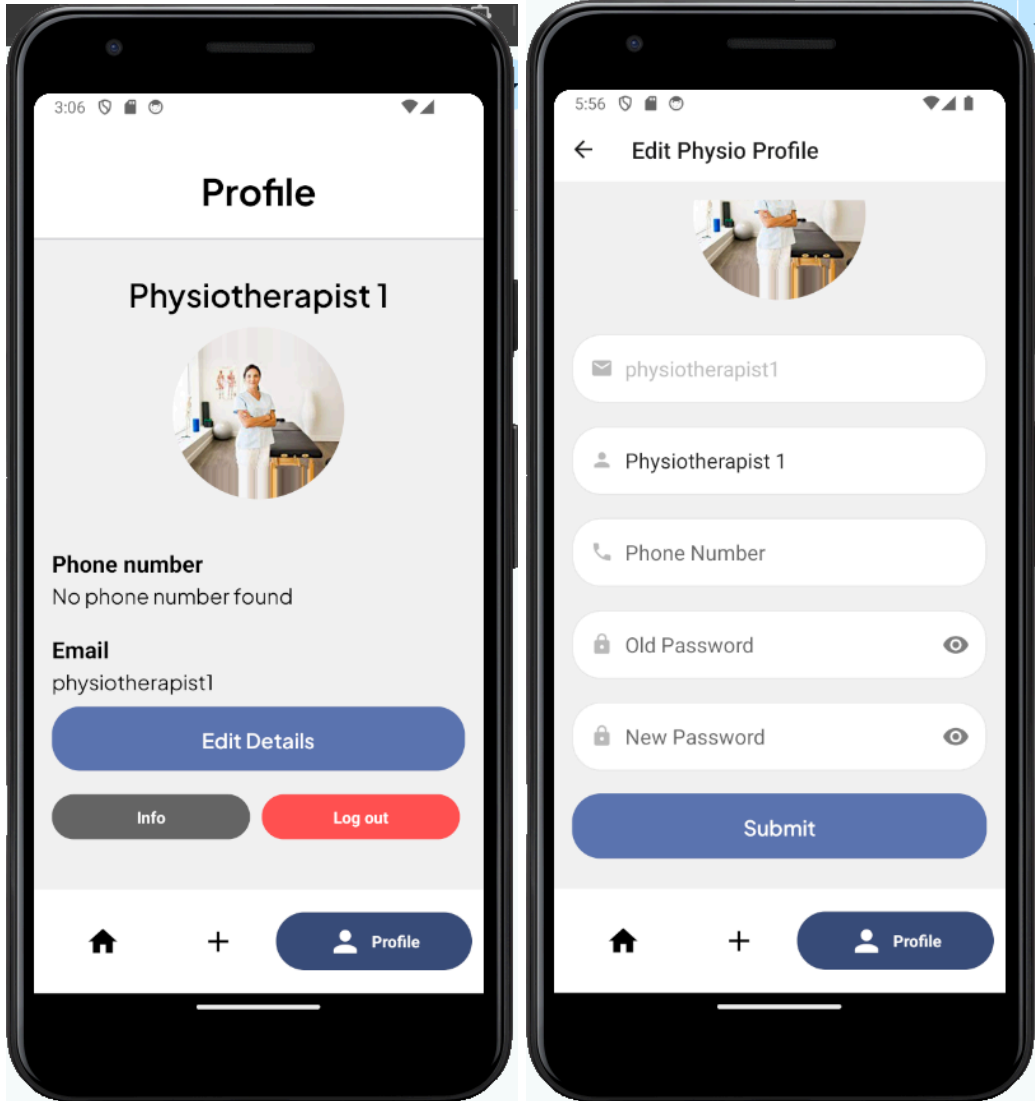
The second page of the physiotherapist application, is the page where they can add or link a patient to their account. On the add page (Figure 6 Create Patient), the physiotherapist fills in the name of the patient and optionally some other information, like their phone number. Then after they click on a button for adding the patient, the account is created and data regarding the progress of the patient can be stored. Also, the patient will receive an email with their account code, so they can activate their account for themselves, which we thought would be a more convenient way for the patient and physiotherapist, instead of the physiotherapist having to send the code to the patient themselves. If a patient switches physiotherapists, a physiotherapist can also link a patient to themselves using the patient's account code (Figure 6 Link Patient).



**Figure 6** Create Patient, Link Patient

### 4.2.3 Profile Page

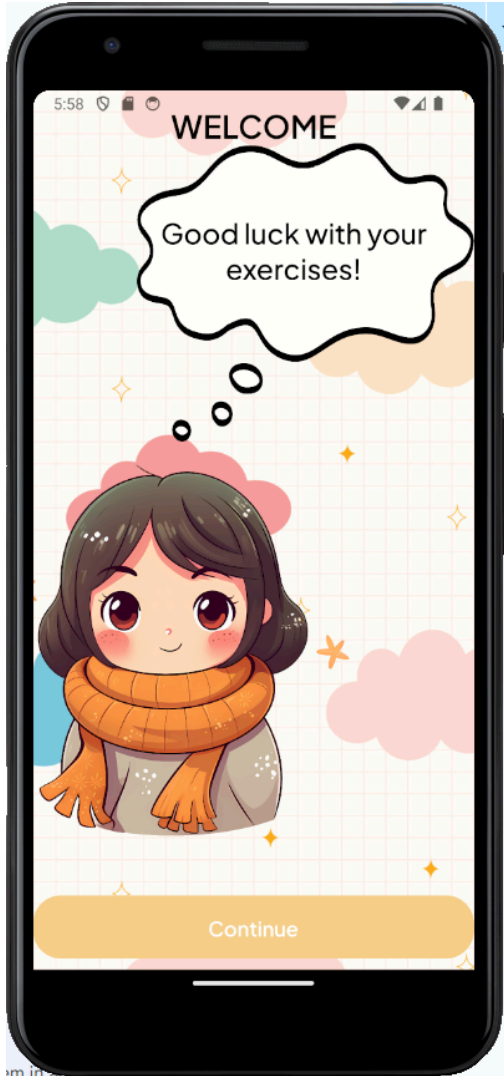
The third page of the physiotherapist application will show the profile of the physiotherapist themselves. Here they will be able to view and alter their own contact information by clicking on a button that will bring them to a form (Figure 7). Next to that, there is an info button that will bring the user to a general information page, that explains how the other pages in the application work. Also, this is where the physiotherapist will be able to log out of the app.



*Figure 7 Physiotherapist Profile Page, Edit profile page*

#### 4.2.4 Patient App

If a patient logs in on the login page, they will be directed to the patient screen on the app. Since the app is designed for children and it was not a high priority it is an application that only has simple basic functionality (Figure 8). First of all, they will be met with a nice and cheerful welcome screen. This page was added to the patient side with the idea that it would make them feel more calm before doing the exercises. After that, there is a page where the patient can view their own profile and their contact details that can also be edited.



**Figure 8** Patient Welcome Screen

#### 4.2.5 Extra Features

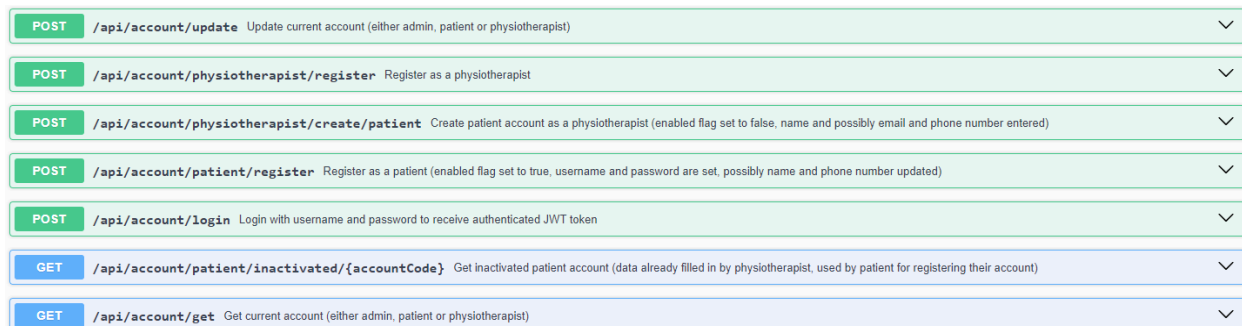
At first, there was also the idea of creating a chat function between the patient and the physiotherapist. However, after discussing with our clients we found out that this feature was not really of importance for the demonstration of the Wearable Breathing Trainer and it thus fell out of the scope of this project.

## 4.3 Backend

In order to make the frontend functional, we need a backend that can handle all necessary requests. We built a REST backend with Spring Boot. This means we created several APIs, through which data could be queried, added, updated, or deleted. For now, this backend runs on our own laptops, but it can be deployed to any virtual machine. We use a PostgreSQL database for the backend to communicate with. This database also runs locally, on a postgres server on our laptops. This can be changed to any location, as long as it is a PostgreSQL database. For the convenience of our clients and future developers, we have created detailed instructions on setting up the project. Moreover, we have implemented Swagger UI into our backend, meaning that documentation is automatically provided under localhost:8080/docs (when running the backend). Our annotations for every endpoint are included as description next to each endpoint. Now, let us go over the several APIs to see how the backend works, as well as go over some other functionalities of the backend.

### 4.3.1 Application Programming Interface (API)

#### 4.3.1.1 Account



POST	/api/account/update	Update current account (either admin, patient or physiotherapist)	▼
POST	/api/account/physiotherapist/register	Register as a physiotherapist	▼
POST	/api/account/physiotherapist/create/patient	Create patient account as a physiotherapist (enabled flag set to false, name and possibly email and phone number entered)	▼
POST	/api/account/patient/register	Register as a patient (enabled flag set to true, username and password are set, possibly name and phone number updated)	▼
POST	/api/account/login	Login with username and password to receive authenticated JWT token	▼
GET	/api/account/patient/inactivated/{accountCode}	Get inactivated patient account (data already filled in by physiotherapist, used by patient for registering their account)	▼
GET	/api/account/get	Get current account (either admin, patient or physiotherapist)	▼

**Figure 9** Account API endpoints

The account API (Figure 9) contains endpoints to get an account, update an account, register accounts, and to login. The system includes three types of accounts, namely a patient, a physiotherapist, and an admin. All three are able to use the get account endpoint, the login endpoint, and the update endpoint. A physiotherapist is able to create an inactivated account with preset data for a patient. This account can be normally used by a physiotherapist, but is inactivated as long as a patient does not activate it. This does not hinder any functionality for the physiotherapist. A patient is then able to query this preset data, change or add data to this, and register their account.

### 4.3.1.2 Patient

GET	/api/patient/treatmentplan/get	Get current treatment plan of patient account	▼
GET	/api/patient/physiotherapist/get	Get current physiotherapist of patient account	▼
GET	/api/patient/get	Get current patient account	▼
GET	/api/patient/devicesettings/get	Get current device settings of patient account	▼

**Figure 10 Patient API endpoints**

The patient API (Figure 10) includes some basic functionalities, which can only be requested by patient accounts. They are able to get their own treatment plan, their physiotherapist, their entire account data, or the device settings set for their account.

### 4.3.1.3 Physiotherapist

POST	/api/physiotherapist/patient/remove/{accountCode}	Unlink patient by account code from current physiotherapist account	▼
POST	/api/physiotherapist/patient/add/{accountCode}	Link patient by account code to current physiotherapist account	▼
GET	/api/physiotherapist/patient/{id}	Get patient by id if treated by current physiotherapist account	▼
GET	/api/physiotherapist/patient/list	List all patients treated by current physiotherapist account	▼
GET	/api/physiotherapist/get	Get current physiotherapist account	▼

**Figure 11 Physiotherapist API endpoints**

The physiotherapist API (Figure 11) contains endpoints, which can only be requested by physiotherapist accounts. They are able to add or remove a patient from their treatment list, based on their account code. This account code is also shown as a QR code in the patient app, meaning physiotherapists can scan this QR code without having to enter the long account code. Next to this, a physiotherapist is able to get the data of a certain patient (if that patient is being treated by the physiotherapist), a list of all patients being treated by this physiotherapist, and their own account.

### 4.3.1.4 Admin

GET	/api/admin/physiotherapist/basic/all	Find all physiotherapist basic information (account code, username, name, phone number, email address)	▼
GET	/api/admin/physiotherapist/all	Find all physiotherapist information (account code, username, name, phone number, email address, list of patients)	▼
GET	/api/admin/patient/basic/all	Find all patient basic information (account code, username, name, phone number, email address)	▼
GET	/api/admin/patient/all	Find all patient information (account code, username, name, phone number, email address, physiotherapist, treatment plan, device settings)	▼

**Figure 12 Admin API endpoints**

The admin API (Figure 12) has some simple endpoints to retrieve all physiotherapists and patients, including simple overviews and some more data.

### 4.3.1.5 Training

POST	/api/training/upload	Upload training data	▼
GET	/api/training/{id}	Get training by id as physiotherapist	▼
GET	/api/training/patient/{patientId}	Get training overview for patient by patient id as physiotherapist, including list of trainings, and lists of weeknumbers/thorax averages/abdomen averages/duration totals	▼

**Figure 13 Training API endpoints**

The training API is crucial to the platform (Figure 13). It contains the endpoint to upload training data to. This endpoint receives and decodes the incoming data from the wearable device, computes peaks and throughs from them, and computes metrics which are saved in the database. It contains an endpoint to retrieve a specific training, including all points and all metrics. The last endpoint is to get an overview of trainings for a specific patient. This includes a list of week numbers, with thorax and abdomen averages, and total durations. Next to this, it contains a boolean seen, meaning a physiotherapist can see in the app whether they already looked at a training or not.

### 4.3.1.6 Device Settings

POST	/api/devicesettings/update	Update device settings as physiotherapist for patient, creates them if they do not yet exist	▼
GET	/api/devicesettings/{patientId}	Get device settings as physiotherapist for patient	▼

**Figure 14 Device settings API endpoints**

The device settings API (Figure 14) makes sure that a physiotherapist can get or update device settings for a patient. This includes an exercise duration, vibration lower bound, vibration upper bound, vibration type (during inhalation, during exhalation or during both) and amount of motors. This data needs to be set for a patient, and can be transferred to a wearable when connected to it.

### 4.3.1.7 Treatment Plan

POST	/api/treatmentplan/update	Update treatment plan as physiotherapist for patient, creates them if they do not yet exist	▼
GET	/api/treatmentplan/{patientId}	Get treatment plan as physiotherapist for patient	▼

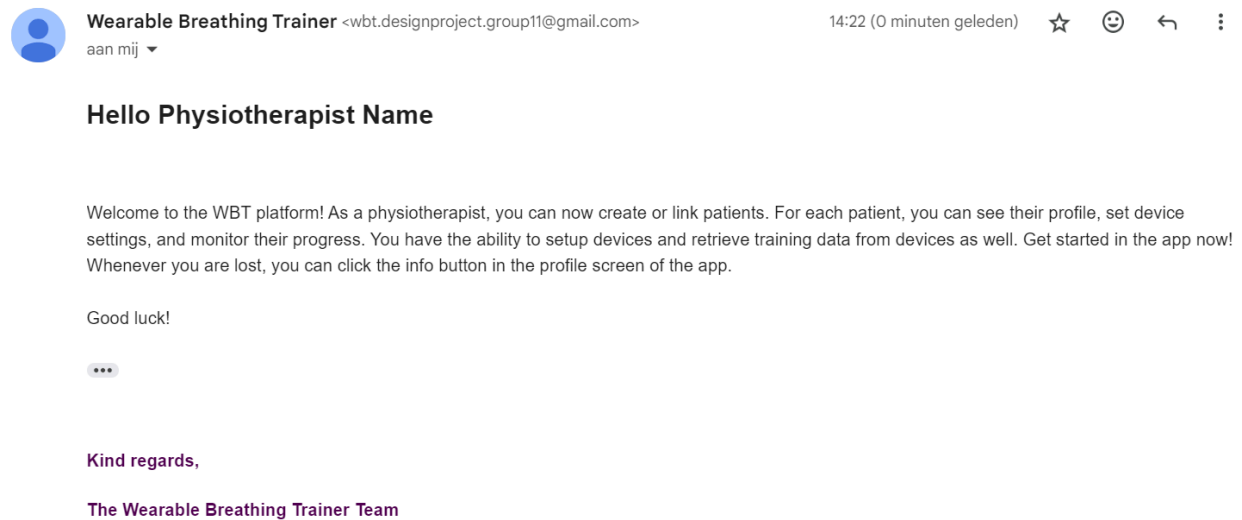
**Figure 15 Treatment plan API endpoints**

The treatment plan API (Figure 15) contains endpoints to get or update a treatment plan for a patient. These can only be used by a physiotherapist. The treatment plan includes several information about the treatment, including created at, starts at and ends at timestamps, a description, and a list of meetings and goals. Although the backend functionality exists, we did not have the time to incorporate this into the frontend of the app.

### 4.3.2 Email

For this project, we created a google email, named [wbt.designproject.group11@gmail.com](mailto:wbt.designproject.group11@gmail.com). This email is used by the platform to send mail notifications to users. Currently, this is being done in three places.

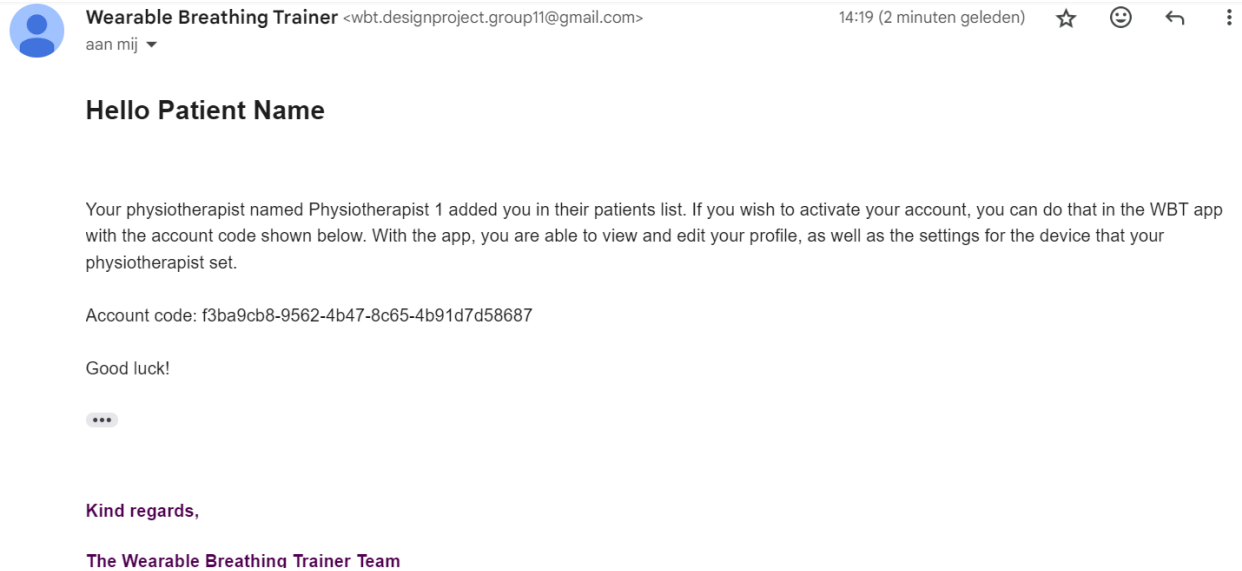
The first place where an email is sent is where a physiotherapist registers an account. It explains the physiotherapist their basic functionalities within the app, and gives information where to click if they are lost. An example of such an email can be seen in Figure 16.



**Figure 16** Welcome email for physiotherapist

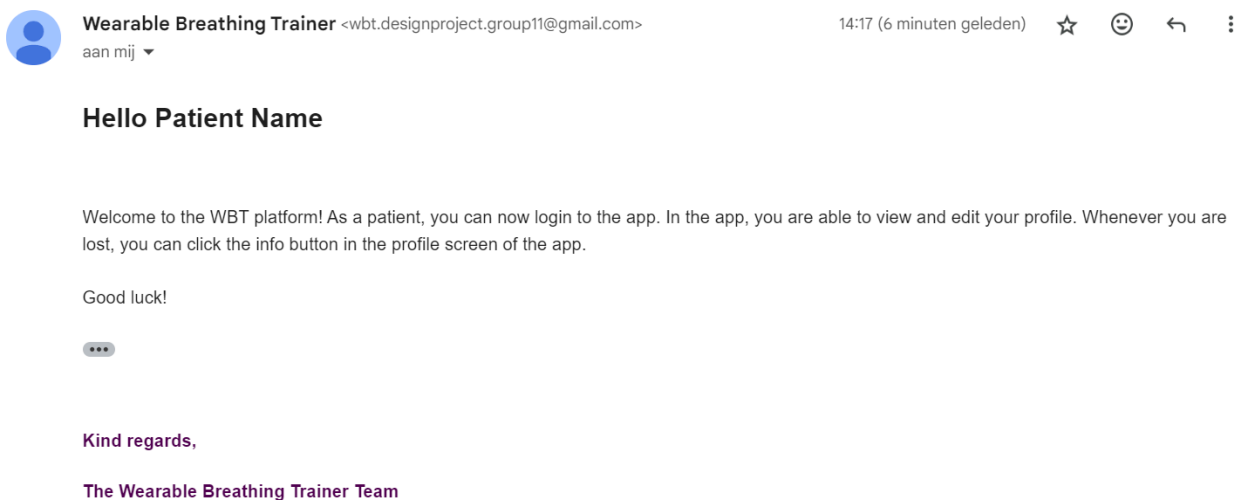
The second place in the backend where an email is sent, is when a physiotherapist adds a patient to their patient list. This means an inactivated patient account is created, and a patient can activate this themselves. This is explained in the email they receive. An example of such an email can be seen in Figure 17.





**Figure 17** Welcome email for patient before registering

The last place where an email is sent, is when a patient actually registers their account. The message they receive is simple, and states they can now login. Besides this, it directs the patient to information pages in the app as well. An example of such an email can be seen in Figure 18.



**Figure 18** Welcome email for patient after registering

### 4.3.3 Calculating breathing metrics

In order to get useful insights from the training data, we needed to calculate certain metrics. All metrics are calculated using the peaks and troughs of the breathing cycles. We will go over how we compute peaks and troughs, as well as how our metrics are calculated.

#### 4.3.3.1 Peak and through detection

In order to identify all peaks and troughs, we first smooth out the data by computing averages. For each point, we look at all points one second before and one second after, and take the average. This means we end up with a curve that is less pointy, but does contain all peaks and troughs. After this, we use a sliding window. The sliding window has a size of 5, meaning that at each point, the 2 previous and the 2 next points are looked at. If we find a maximum, or a minimum, we put these in a list. In the end, it might occur that two or more maximums/ minimums exist directly after each other. We included logic to make sure that only the lowest minimum or the highest maximum would stay. This means that in the end, after every minimum, there is a maximum, and vice versa. We now end up with two lists of peaks and troughs.

#### 4.3.3.2 Metrics calculations

The following metrics were calculated in accordance with the clients' request, which are based on their prior research (Juffermans, 2023).

##### **Thorax and abdomen percentage**

This metric is calculated by taking the average fluctuations in abdomen and thorax readings from sensors and comparing them to each other to see if abdomen or thorax breathing was used more during the training.

##### **Time per breath and frequency**

The time per breath, or breath cycle time, is the amount of seconds it takes to complete one full breath cycle. The frequency is the amount of breaths per minute. It is computed as an average over all breath cycles in the training.

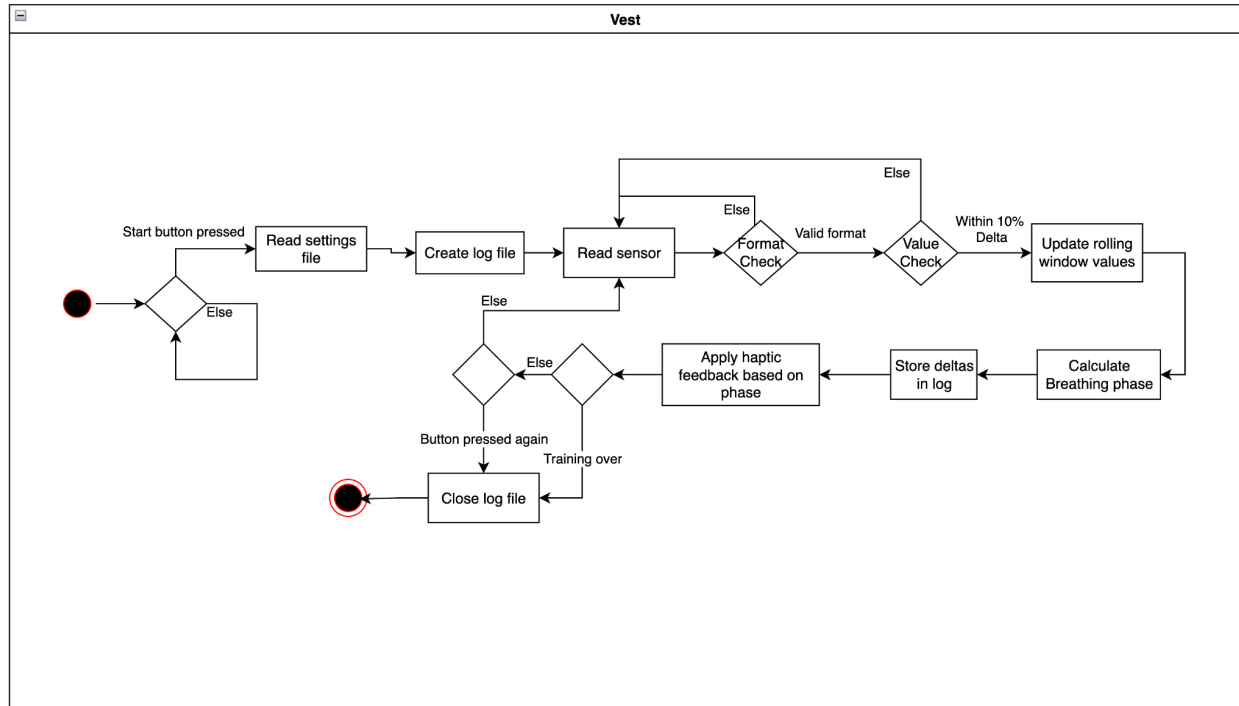
##### **Inhalation time and exhalation time**

The inhalation metric shows the average inhalation time of the patient during training. It is computed as an average over the entire training. The exhale time is calculated by subtracting the inhalation time from the breathing cycle time.

## 4.4 Wearable

We have divided the functionality of the wearable into two main parts, communication and handling training. This section discusses the wearable's configuration and its functionality during training, while chapter 4.4 discusses communication with the app.

The following activity diagram shows how a training is handled by the wearable.



**Figure 19** Activity diagram of test

Throughout this process the wearable makes use of a number of hardware components. The following subchapters will delve into each of the components, how they work and the design decisions behind them. An exact wiring diagram can be found in Figure 31, located in Appendix A.

### 4.4.1 Microcontroller

For the backbone of our system we selected an ESP32 microcontroller. Many factors went into this decision, including its low cost, availability and convenience. Our clients already provided one board at the start of the project and its low cost meant that we could easily purchase more to accelerate the development process. The main reason why we opted for the ESP32, however, was its convenience. Its compatibility with the Arduino IDE means that it is easily programmable and that we have access to a wide array of software libraries. Additionally it offers both Bluetooth and Wi-Fi connectivity options, which we found attractive at the start of the project as we wanted to explore both venues for communicating with the app.

## 4.4.2 Breathpal Sensor

To measure the breaths we use the BreathPal (Ben Bultink, 2020) sensor, which communicates with the microcontroller through UART format serial communication, which is connected to pin G16. As mentioned in section 1.1.2 the sensor uses 2 RIP straps, one to measure Abdominal and one to measure Thorax expansion. The sensor takes 10 measurements per second and publishes the data in ASCII format in the following manner:

```
31236,34667,-1000,121,700  
31237,34667,-1006,130,710  
31239,34664,-1004,131,727
```

Each new line depicts a new sensor reading, with the first number representing the reading of the Abdominal strap, and the second number the reading of the Thorax strap. The following 3 values are the x,y,z readings of the integrated accelerometer. To read this output on the ESP we start a Serial port with rx pin 17 and SERIAL\_8N1 configuration, and read each byte, until we encounter a CRLF, at which point we know we've reached the end of the current reading. The bytes are then put into a string and the abdominal and thorax readings are then ready to be further processed by the microcontroller. Due to time constraints we don't currently utilize or store the accelerometer values, however they could be used to determine whether the patient moved during the exercise, which could skew the Breathpal readings.

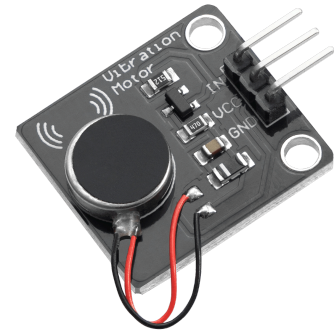
## 4.4.3 Processing of Sensor readings

To get useful data from the sensor and give live feedback, we need to determine what phase of breathing we are in, i.e. inhaling, exhaling or pausing. To do this, our clients have provided us with a code base from Angelika Mader (May 2023) which works by using a sliding window. Then, in the sliding window we use a certain number of consecutive readings to determine the change in phase. For example, if we see a decrease in 3 consecutive readings, we can safely determine we are exhaling. Meanwhile, if there are consecutive readings that only increase, we must be inhaling. How this information is used to give haptic feedback is described in the next section.

We did have to modify the code in question (discussed in section 4.4.2), since we had access to a newer version of the Breathpal sensor, which doesn't use bit flags in its communication.

#### 4.4.4 Vibration motors

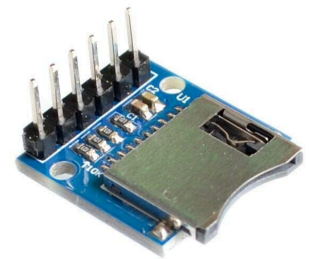
To give the haptic feedback to the user while the training is going on, we use 5 vibration motors. The choice in the type motor was simple, since they were provided to us by our clients. The motors are set up to be used in multiple patterns, with options for 5, 3 or 1 motor. There are also options to only vibrate on inhalation or exhalation only, or both. These motors work in a way that the vibration bleeds over towards the belly on inhalations, and towards the back while exhaling. Because the outermost motors are supposed to vibrate in the exact same way, they share their signal wire. This goes for the middle two motors as well. All the motors share power and grounding, to reduce the amount of wires that have to go into the ESP. The motors are put on the body by using a pantyhose, in which the motors are placed. We chose this crude way because we simply needed to develop the feedback, since a higher fidelity version of the wearable already exists. As for how the feedback is determined, we need a numerical value on how far we are into a breath. This is determined by taking the maximum and minimum values noted during the whole training, since we expect the user to get at least somewhat close to these values on every breath. Then using this information, we determine which exact motors should vibrate and how strongly.



**Figure 20** Vibration Motor

#### 4.4.5 Storage

There was also a necessity to store certain data beyond rebooting, which is possible normally for the ESP. However, we did not want to be limited by the 4MB of flash memory of our ESP32 boards, so we added a Youmile Micro SD card module. During development we used a 32GB sd card, however the final system can use any size of FAT32 encoded micro SD card. To interface with the SD card we used v1.2.4 of the SD library for Arduino.



**Figure 21** SD Reader

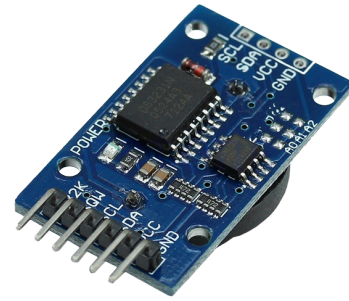
The settings are stored as strings on the SD card in a simple file called settings.txt. The trainings are separated in a folder. When the device is connected to the app, all training sessions are sent over and deleted on the SD card. The trainings get unique filenames by using the clock module, which is explained in section 4.4.7.

#### 4.4.6 Power supply

The system is powered by a 5v power supply, as 3.5v is the minimum voltage needed to power the breathpal sensor and some of the components(rtc module, vibration motors) required the 5v power. As it is we can guarantee that the system can be powered by any source capable of providing a 5V output with a current rating of 1 amp.

#### 4.4.7 Clock module

Finally we needed the arduino to be able to keep track of the time even after it is powered off, so that we can keep track of when patients do their training. Since we decided against connecting the ESP to Wi-Fi, the reasons for which we discuss in section 4.4, we needed to find a different way of keeping its clock correct. We found the solution in a DS1307 RTC Clock module, with which we interfaced using the RTCLib library for arduino by Adafruit. The clock is set whenever the program is compiled, and we have found that to be accurate enough, however there is also the possibility to set the clock every time the wearable device connects to the app, should more accurate timekeeping be required in future development. The clock is used to generate unique file names for the log of each training in the following format: **TRAINING\_YYYY-MM-DD\_(WEEKDAY)\_HH-mm.txt**.



**Figure 22** RTC module

### 4.5 Connection

For connecting the wearable with the mobile application we had 2 options, either Wi-Fi or BLE. Here we will discuss design decisions in regard to communication.

#### 4.5.1 Wi-Fi

We decided against using Wi-Fi as we did not intend to add more than a few buttons to the wearable, we do not have a simple way for the user to directly input their Wi-Fi credentials. We did consider a system where you could initially connect to the wearable via Bluetooth, send over your credentials and then switch to using wifi for communicating settings and logs, however that went against the simplicity request of our clients. Therefore we settled on using the ESP's other wireless communication technology: Bluetooth Low Energy (BLE).

#### 4.5.2 BLE

At the start of the project we decided to use BLE in order to have live feedback on the app side. However due to change in requirements we use BLE to send over settings, and to send over the training files now stored on the ESP.

The BLE communication is implemented with the use of javascript with the react-native-ble-plx library and Arduino cpp with the BLEDevice, BLEServer, BLE2902 libraries. These libraries communicate according to the GATT connection protocol. Within the GATT protocol we utilize two main methods: the write and the read from characteristic. I will describe how each of these methods work.

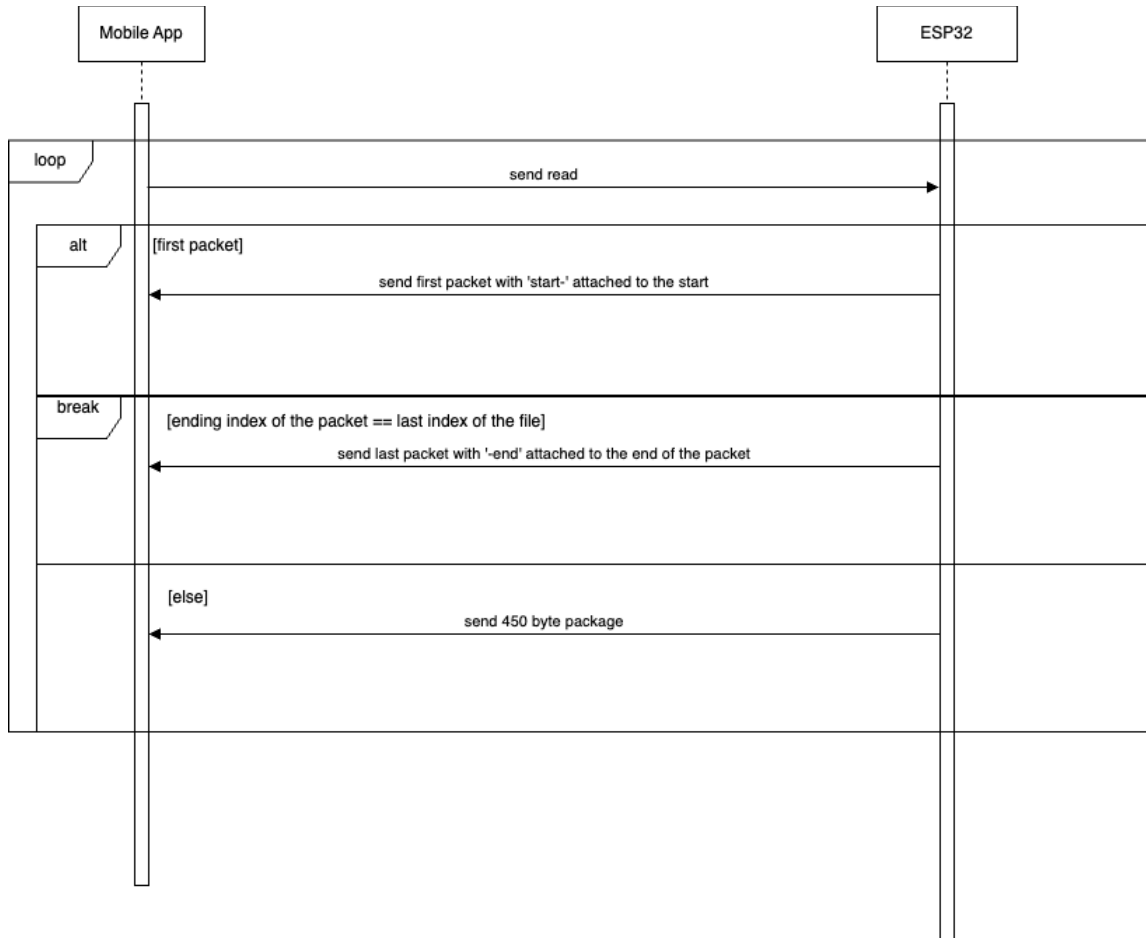
Firstly, connecting to the wearable device is done by the esp32 advertising its BLE connection so that in the app it is possible to tap on the wearable device and connect to it. To make it easier

to know which wearable device to connect to, the app filters bluetooth devices by name. Due to time constraints we have not protected against unwanted connections. Such weaknesses we did not find to be a priority due to this being a project targeted to demo how such technology would work.

Secondly, we will talk about the write to characteristic method. This function is used to send over the settings that a physiotherapist would set for a patient. When they set the settings and click 'Save Settings', a string of a specific format is sent to the ESP, for example this message 'configurations-300-25-75-2-3-2'. Now to explain the message.

The first part should always be the same, and is used to see if we are actually receiving a configuration message. If we do not receive 'configurations-...' we will notify the characteristic with the message 'incorrect'. The second part of the message indicates the duration of the training in seconds, which in the example is 5 minutes (300 seconds). Third part indicates the lower bound of the vibration motors (25% of the strength). The fourth - the higher bound of the vibration motors (75% of the strength). The fifth part decides what kind of vibration pattern we use, to determine this, we have a map of {0: "INHALE", 1: "EXHALE", 2: "BOTH"} meaning that in this example message we would choose to vibrate on both inhalation and exhalation. The 6th part, shows the amount of vibration motors to use, either 1, 3 or 5. The last part is the patientID. We check the validity for every part, if any part is incorrect we will notify the characteristic with the message 'incorrect'. With the use of this protocol we can make our write to characteristic method more resilient, especially since we have not protected against unwanted connections.

Secondly, we will talk about the read from characteristic method. This function is used to sync the data with the application. Because of the requirement [mf5] we have to send all the training files at once, and this operation we call syncing the data. Due to BLE being able to send over 512 bytes at a single time we had to implement our own protocol to send over the data. We first thought that we could do this in a singular read request, which would have been much easier and nicer, but later on it turned out that the esp would write to the characteristic only after it returns from the onRead() method. This meant that we could fill up the characteristic with 512 bytes of data on a singular read request. If we tried to fit more bytes than that, the esp would get stuck and not send anything at all. The protocol we developed for a singular file is illustrated in a sequence diagram in Figure 23.



**Figure 23** Single File Protocol

In essence the protocol works by sending the entire file over in packets, where the 'start-' indicates the start of a file and '-end' indicates the end of the file. With the 'start\_' message we also send the size of the file and the timestamp of when the training was done. An example of a start message would be the following - 'start\_1886\_2024.4.11\_...' where 1886 is the size of the file, 2024.4.11 is the date for April 11th 2024 and the '...' indicates the data. The file size is used for a progress bar on the app side, so the user could see how much is left in transferring the data. As for the timestamp, it is used for displaying when the training has been completed.

On the mobile app side, the file is constructed with the start of the file being triggered by the 'start\_' message, and the file is sent over to the backend and saved in the database after receiving '/end'. One thing that is abstracted from the diagram is that we also send the file size with the start message. This helps us in implementing a progress bar which shows how far are we from transferring the file.

Sending multiple files works pretty much exactly the same as for sending a single file (described by Figure 30, which can be found in Appendix A). The protocol is fairly similar except we append 'startingsync/' message for starting file transfer and '/endingsync' for ending file transfer. In addition to the 'startingsync/' we also append the patientID and how many files are going to be sent over. Patient ID helps us to link the data to the patient in the backend, and how many



files are being sent over helps with the progress bar of sending files. An example of such a message is the following '*startingsync/2/5/...*' where instead of the '*...*' we would have a '*start\_*' message.

## 4.6 Conclusion

So, for the frontend we wanted to make a fully functional and intuitive interface for the physiotherapist and a smaller application for the patients. We have realized the physiotherapist application by dividing the functionalities over several pages. This way we have managed to give the physiotherapist intuitive ways of creating and logging into their account, connecting to a wearable, a way to add patients and view them in a list that gives access to editing the patient's device settings and view their profiles and training results. Unfortunately, since the patient app did not have a high priority we implemented it with only the very basic functionality of showing the patients profile on the patient side of the application.

To establish a wireless connection to the wearable device, we have decided to use BLE. The reasoning behind this is that the clients preferred bluetooth over wifi. We use BLE specifically, because this is compatible with the ESP32, while regular bluetooth is not. A downside to this is that BLE is not capable of sending lots of data in a short amount of time. So, when sending over the training data of the patient from the ESP to the app, it will take more time compared to regular bluetooth. To counter this, we implemented some data compression techniques. More specifically, we used delta encoding, which means we only saved the change compared to the previous value as opposed to the whole measured value. Since each data entry consists of 4-5 digits, and the changes between each entry - only 1-2 digits - delta encoding allowed us to compress our data quite significantly. We also left out the values of the accelerometer, because we did not implement the requirements that used this data.

Once the system is designed and implemented it is essential that we ensure it works as intended. To do this we have done several tests for different parts of our system, which is described in the following section.

## 5. Testing Plan

In this chapter we will go over how we tested the different parts of the system and their cooperation with each other. First we will focus on how we tested the front end, then we go into the automated tests of the back end of the application, we discuss how we tested the hardware and software parts of the wearable, and finally we cover what we learned from a user test of the complete system which we did with our clients.

It is worth mentioning, that for system testing we have acquired the approval of Computer & Information Sciences (CIS) Ethics Committee (application number 240149).

### 5.1 Frontend

#### 5.1.1 Frontend Tests

For the frontend, creating automated tests would be quite difficult. So, in order to ensure quality of our product anyways, we made sure to test the frontend manually. This has been done in two ways. First of all, during the development of new features, they were constantly tested manually using an emulator on our device. So, for example, when implementing a new button the emulator would be used to see if the button looked aesthetically pleasing and then if the button did the right action after pressing it. Next to that, we had several meetings with both our clients and our supervisor, where we showed our frontend design to them to ask for feedback and if things needed to be changed. This form of testing allowed us to get rid of most of the bias we have as a developer and see the application from a different perspective and make sure it satisfies the needs of the client. Additionally we also received viable feedback about our frontend during the user test which is discussed in 5.4.

#### 5.1.2 Frontend Testing Results

By performing the tests mentioned above we, of course, fixed a lot of small bugs. Next to that, the tests were useful for making new decisions in our design choices, such as changing certain colors for the background, buttons or text on the pages. Especially the meetings with clients helped with this. For example we have made quite some changes in our navigation bar and how the navigation works on our page. Such as the icons that were used and the positions they took. Or, some pages were a bit full according to our clients, so we fixed that by making dropdowns for items such as the patient list or the training results.

### 5.2 Backend

#### 5.2.1 Backend Tests

For the backend, creating automated tests is feasible. We did so by creating automated unit tests. Tests exist for all APIs, except for the training API. We tested uploading trainings and calculating metrics by hand, instead of doing this with automated tests. For the account API,

device settings API, patient API, physiotherapist API, and treatment plan API, numerous tests exist. In total, we wrote 52 tests. These all make calls to certain endpoints, and test both normal functionality as well as edge cases. The tests make use of the regular database, but changes are not persisted. This means that the tests can validate functionality, without actually changing the database. The distribution of tests is:

- 24 account controller tests
- 8 device settings controller tests
- 7 patient controller tests
- 9 physiotherapist controller tests
- 4 treatment plan controller tests

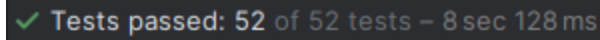
### 5.2.2 Backend Testing Results

The coverage of all tests can be seen in the image below (Figure 24). As we can see, 80 percent of classes are included in the tests, and 61 percent of all lines in the backend are included. There are several untested functionalities, like sending email, the training controller, and some util classes to encode or decode jwt tokens, encrypting passwords and parsing csv files. The coverage of services for all entities other than training, is at least 80 to 90 percent.

Element ^	Class, %	Method, %	Line, %
com.designproject.group11.wbt	80% (72/89)	70% (316/448)	61% (632/1021)
components	95% (23/24)	74% (111/149)	66% (318/476)
controllers	100% (7/7)	80% (28/35)	82% (87/105)
account	100% (4/4)	83% (20/24)	89% (60/67)
devicesettings	100% (1/1)	100% (3/3)	100% (11/11)
training	100% (1/1)	40% (2/5)	31% (5/16)
treatmentplan	100% (1/1)	100% (3/3)	100% (11/11)
entities	90% (10/11)	72% (54/75)	66% (85/128)
account	80% (4/5)	100% (24/24)	100% (40/40)
devicesettings	100% (2/2)	90% (10/11)	94% (17/18)
training	100% (1/1)	5% (1/18)	2% (1/38)
treatmentplan	100% (3/3)	86% (19/22)	84% (27/32)
repositories	100% (0/0)	100% (0/0)	100% (0/0)
account	100% (0/0)	100% (0/0)	100% (0/0)
devicesettings	100% (0/0)	100% (0/0)	100% (0/0)
training	100% (0/0)	100% (0/0)	100% (0/0)
treatmentplan	100% (0/0)	100% (0/0)	100% (0/0)
services	100% (6/6)	74% (29/39)	60% (146/243)
account	100% (3/3)	84% (22/26)	93% (101/108)
devicesettings	100% (1/1)	100% (3/3)	88% (23/26)
training	100% (1/1)	25% (2/8)	7% (7/91)
treatmentplan	100% (1/1)	100% (2/2)	83% (15/18)
email	100% (2/2)	100% (5/5)	92% (25/27)
exceptions	72% (21/29)	72% (36/50)	72% (36/50)
filters	100% (1/1)	100% (3/3)	100% (15/15)
representations	76% (16/21)	70% (135/191)	70% (192/271)
security	100% (3/3)	64% (9/14)	80% (20/25)
settings	100% (1/1)	100% (6/6)	100% (6/6)
util	71% (5/7)	37% (11/29)	13% (20/150)
WbtApplication	0% (0/1)	0% (0/1)	0% (0/1)

**Figure 24 Coverage of backend testing**

All 52 tests succeed (Figure 25).

A dark grey rectangular banner with a green checkmark icon on the left and the text "Tests passed: 52 of 52 tests - 8 sec 128 ms" in a light grey font.

**Figure 25** Backend testing results

## 5.3 Wearable

### 5.3.1 Testing the Wearable

For the esp code, we did not write explicit unit tests, because it is relatively hard to properly test separate units. Instead we tested continuously while we were developing. These tests mostly focused on proper sensor readings and functioning, as well as testing whether the vibration patterns were functioning properly. We also did integration testing for the firmware. The instructions, acceptance criteria and outcome is in Appendix B. Since the goal of the wearable side of the project was to store training data and send it to the back end, we did a test where we did exactly this. This allowed us to test all the components of the wearable, except for writing settings to the wearable device. This was tested in a similar integration test, in which we tried updating multiple settings and sending them to the wearable device. After that, we did a training to see whether the settings were used properly.

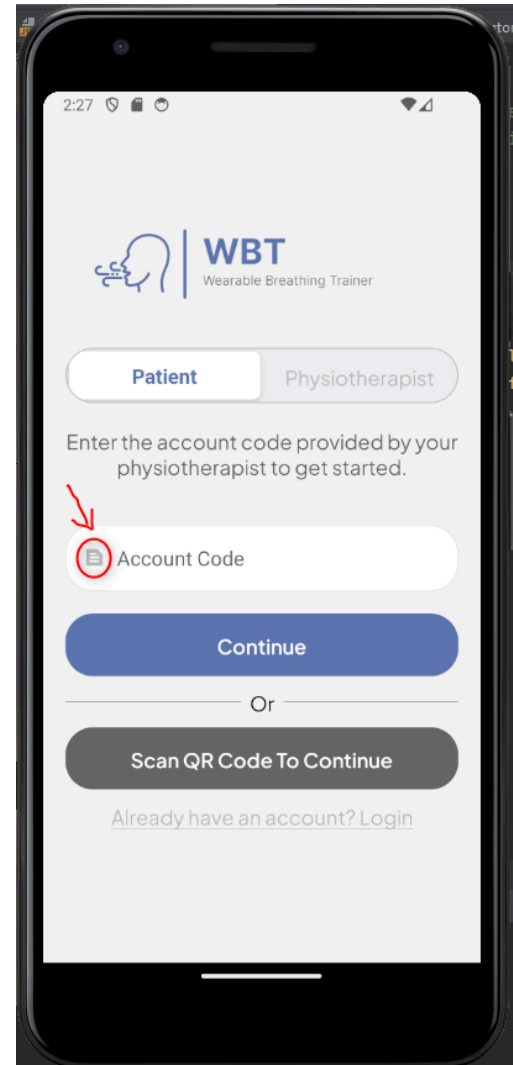
### 5.3.2 Wearable Testing Results

In the tests described above we found multiple small bugs, which we fixed with further development. More notable failed tests include that some of the vibration motors did not function properly, as they were vibrating too weakly. We were limited in our options to solve this, because we did not have spare motors. However, if other motors can be acquired in future development, they can be exchanged for the weaker motors relatively easily. We also encountered some abnormally low readings from the sensor. To counter this, we only accepted sensor values that were within 10% of the previous reading, which filtered out the extreme values while not discarding too much data.

## 5.4 User testing with clients

In our final week of the project we performed a user test with Sarah, our client. We used this test to figure out some final small details that could be fixed before we finished the project, mainly to see how intuitive the frontend design of our app really is. The main points that came forward from this test was that our registration page (Figure 26) had a qr code icon for the account code, which was something we completely overlooked, so it was nice that Sarah pointed it out. Next to that, pages could be slightly unclear for new users, which is why Sarah advised us to add some extra specific info icons for pages, such as the sync settings page, and then add a button for a general info page in the profile page. Furthermore it was apparent that the device settings could only be viewed on the edit page, which was not logical. To solve this, we added the current device settings to the patient profile page as well. Also, it was unclear that the breathing graphs for a specific training were scrollable, so to improve upon this we added the timestamps in the graph, which makes it look nicer in general. Finally, Sarah accidentally logged out of her account three times during the test, so we added a confirmation dialogue to ensure that would not happen anymore.

Overall, the feedback was very positive and in general Sarah found the application to be intuitive and visually appealing.



**Figure 26** Registration Page

## 5.5 Conclusion

In the end, the testing has helped us to verify that our system works as intended. Specifically for the frontend, the tests have helped us check whether the interface is intuitive and has no bugs. Next to that, the tests for the backend were useful for checking if all the api calls work as intended and the tests for the wearable were important for finding bugs in the sensors and motors.

## 6. General Discussion

The goal was to create an application together with a wearable device that would help physiotherapists with treating their patients that have a breathing disorder, so that it can be used for a demonstration. This was done by implementing the various requirements that have been specified in Chapter 3. In the following sections we will talk about how we worked as a team and exactly what requirements have been fulfilled. After that we will clarify the limitations of this application and how it can be improved in the future.

### 6.1 Team Evaluation

Since a lot of the team members already knew each other and had worked together before on previous projects, things went smoothly from the beginning. Roles were divided quickly and arrangements were made on who would be working on what. Also, when there were discussions about anything in the project, which could be the planning, the design, meeting times and so on, the discussions were always very civil and everyone felt heard in the conversation. Overall we think this has led to great working progress and a successful project.

#### 6.1.1 Planning

To ensure a good pace for the development of our product, we had set up a weekly planning from the beginning of the module with sprints that lasted two weeks each. Every week we had at least one scrum meeting, led by our dedicated scrum master Matthias, to check what everybody was working on and if people had any questions or needed help from someone else. Next to that, at the end of each sprint we evaluated our progress and checked if we needed to make any changes in our planning or design.

#### Kick-off sprint. Weeks 1-2

Dates: 05.02.2024 - 16.02.2024

- Meet clients and collect requirements
- Have brainstorming session
- Propose application design
- Divide responsibilities
- Set up the project
- Create project proposal
- Find university supervisor

#### Sprint 1. Weeks 3-5

Dates: 26.02.2024 - 15.03.24

- Verify project proposal with clients and supervisor
- Finish requirements with *must* priority (i.e. Finish MVP)

- Present current state of the product to the clients

## Sprint 2 Weeks 6-7

Dates: 18.03.2024 - 28.03.24

- Finish missing *must* requirements if any are left
- Start working on the report
- Finish *should* requirements
- Present current state of the product to the clients
- Present initial report to the supervisor

## Sprint 3. Weeks 8-9

Dates: 02.04.2024 - 12.04.2024

- Finish *should* requirements if any are left
- Finish *could* requirements, if *must* and *should* are finished (i.e. finish the final product)
- Verify with clients and supervisor that the product meets requirements
- Finish the preliminary version of the report
- Present current state of the product to the clients, possibly with pediatric physiotherapist (only if the project is finished at this point)
- Create project poster
- Create project presentation slides
- Identify the most important tasks for the last sprint

## Sprint 4. Weeks 10-11

Dates: 15.04.2024 - 26.04.24

- Make final adjustments to the code
- Make final adjustments to the report
- Create a manual for the developed app, aimed at the clients
- Finish “could” requirements if any are left

### 6.1.2 Responsibilities

Since the beginning of this module it was quite clear that the interests of our group were a little divided. Some members of our team were more interested in developing web applications, or maybe even some AI, while other members thought it might be nice to do something a bit more hardware related. This is why the Wearable Breathing Trainer project was a good fit for our group. We could combine the different wishes of our team members, and we came up with the following task distribution.

- **Daan:** Backend/database designer & developer. Also helped a lot with frontend development and setting up the project.

- **Valerijs:** Frontend designer & developer. Has done the most work on the frontend.
- **Guido:** Frontend designer & developer. Has also done a lot of the writing and layout for the report.
- **Matthias:** Scrum master and hardware team member. Has also worked on getting the right metrics from training data.
- **Simeon:** Main hardware team member. Has done a lot of work to make sure all the sensors and devices worked correctly.
- **Kipras:** Hardware team member and connection lead. Has made sure the BLE worked correctly.

This is a rough generalization of our task. During the project all team members helped out wherever help was needed.

## 6.2 Did we meet the goal?

All “Must” and “Should” requirements are fulfilled, except for [sf9] that asked for the physiotherapist to have the ability to see how much a patient has moved during a training. Since we did not use the accelerometer data this was not realized. Also not all of the “Could” requirements are done. For example, the patient requirements where the patient could also connect to a wearable [cf2] and start the training from the app [cf3]. We did manage to implement some of the physiotherapist could requirements, but not the treatment plan [cf9], or seeing fake breathing [cf10] and encryption [cf13]. This was mostly due to time constraints, and we viewed the unfinished requirements as disproportionately time consuming compared to the benefit it would bring. Overall, the important project requirements have been met and the finished product satisfies the client’s needs, which would mean that we have met the goal.

### 6.2.1 Finished demonstrator

With the requirements that were finished on time we have delivered a final product that is able to administer training and send data over to the server. Then the web app is able to display a list of patients with the training results and data analytics, as well as set settings on the wearable device. The settings allow extensive modification in the feedback that is given during the training. The web app allows administering multiple patients. The backend is able to calculate multiple interesting metrics, which physiotherapists can use to monitor training progress. However, not all our requirements are fulfilled, and thus we have some limitations and possible future improvements for our product. This will be discussed in the chapters 6.3 and 6.4.

## 6.3 Limitations

Even though we have delivered a product that satisfies the client’s needs, it is still important to realize and define the limitations of the wearable and application.



### 6.3.1 BLE

At the start of the project we had decided to use BLE, because it has been used before for other devices like fitbits, which are similar to the WBT. However, as we found out closer to the end of the project, BLE comes with a limitation. Since it is low energy, the rate at which it is able to send over data is low. For now, we feel like the amount of time it takes to send over data is sufficient, but still it could be considered slow. Especially if you would need to send large amounts of data at some point.

### 6.3.2 Patient App

Since our focus was mainly on the physiotherapist application, the patients side of the application could be described as a bit basic. This is no problem for the scope of this project, but still the functionality of the patient app only consists of the absolutely necessary functionalities.

### 6.3.3 Security and Privacy

Since the system was intended to be used as a demonstration and not handle actual patient data, we did not focus our design efforts into implementing adequate security measures in the system.

We consider the mobile application to be conventionally secure as login passwords are hashed and salted, before they are stored in the database, and the application utilizes json web tokens for authentication(which is also one of the requirements that were added throughout development - mn4).

When it comes to the wearable on the other hand, since we ran into a bunch of issues during development we prioritized getting a working system to our clients other than a secure one. This means that the settings and logs are stored in plain text files on the sd card of the wearable device. This would be less than desirable as anyone with physical access to the wearable device would also have access to sensitive patient data. Additionally because of our simplified Bluetooth protocol once the device starts advertising, theoretically every other bluetooth device could connect to it, and send read requests to the characteristic, thus getting access to the log files on the sd card.

It goes without saying that if this system were to be brought into market all of the above mentioned security concerns would need to be addressed. Especially since it is a medical product with sensitive patient information, which should be protected by higher standards in order for the product to be brought to the market.

### 6.3.4 Other limitations

There is also no functionality yet for explicitly viewing missed training sessions, although the physiotherapist can see how many training sessions are done. We also did not yet use the accelerometer data, which would potentially allow us to notify the users when there is too much

movement during the training. In future development we also recommend encrypting training data when sending it, and encrypting it on the ESP as well.

## 6.4 Future planning

What will happen with the project when we are finished?

This project will end at the end of this study year. Our prototype will be used for the demonstration and our clients hope that they can write a new proposal for the follow up, where they can improve the robustness and design of the application with more user testing with the physiotherapists and patients. Also they would like to research how visual feedback impacts the trainings and see what could be improved upon in that field.

As the system is medical in nature, if it were to be deployed all the security issues mentioned in 6.3.3 would need to be addressed, and you'd need to ensure that it complies with GDPR and all the requirements and regulations for storing medical data.

### 6.4.1 Wearable improvements

When it comes to data processing on the wearable device, the accelerometer values of each reading could be processed along with the thorax and abdominal values. This would allow the system to detect when the patient makes sudden movements during, which has been known to disturb the readings of the BreathPal. This means that the trainings where this occurs can be then discarded or marked as potentially misleading, which would allow physiotherapists to better understand their patients' progress. Alternatively, these readings could also be stored and sent over to the back end as part of the training logs.

### 6.4.2 Patient app improvements

For improvement on the application side, it would be nice for the physiotherapist to be able to get live feedback on trainings, instead of only being able to see the data after the training has been done. With having live visual feedback, the physiotherapist would be able to help the patient with helping and giving feedback live while the patient is doing their exercises. Next to that, if we, as stated above, gain access to the accelerometer data, it would be nice to have some visual feedback in the graphs when the child moves a lot, because this could indicate that the sensor data was inaccurate.

### 6.4.3 General app improvements

The app in the current state has several points that need improvements. First, some of the libraries used are outdated and thus have severe security vulnerabilities, which would have to be addressed. Secondly, the important variables, such as the backend url, are stored in the plain text in the code. In the future, environmental variables would have to be used instead, to add security to the app. Finally, a caching layer could be added to reduce the load of the backend, which would increase if the app was to go live.

#### 6.4.4 ESP Casing

Since our product was only used as a demo we did not think about how the ESP would be secured on the wearable. In the future it might be nice for a team to think about what would be the best casing for the ESP and what batteries it should include to make sure it lasts long enough and that it could be charged easily.

#### 6.4.5 Different Microcontroller

Through figuring out the limitations of BLE, we also want to note that the ESP32 is limited to BLE. For the purposes of what this project wants to achieve, we would need faster communication/ data transmission, thus the ESP32 is not an ideal microcontroller for this purpose.

## 7. Conclusion

To conclude, the project was a success. There are still a few things that can be improved upon, but overall a nice product was delivered. First of all, the problem that was apparent was that physiotherapists in the Netherlands have quite some (young) patients with breathing difficulties, but little time to meet with these patients, which made it difficult to monitor the project. To solve this a wearable was created that would allow children to do exercises at home with the use of haptic feedback. We were tasked with designing and implementing an application for this wearable that would aid the physiotherapist in setting the settings on the wearable and being able to view the result of training done at home, while also improving the wearable itself with haptic feedback. Together with our clients we came up with a system proposal where agreements were made on how the application, hardware and communication would look like. Then, through meetings with the clients, requirements were set up to make clear what should be done in order to realize a working system to the needs of our clients. After the requirements were made a design for the system could be created and implemented, which was done with constant feedback from our supervisor and client to make sure the app design was visually appealing and functional. Next to that, we did our own testing with manual testing and user testing for the frontend. Also there are automated api tests for the backend and again manual testing and integration testing for the wearable. After extensive testing, our product was finished. We met our goal, but we still came up with some future improvements such as some better security and privacy, adding accelerometer data and having a casing for the esp. All in all, we hope to have contributed to the project and feel like with some future improvements the Wearable Breathing Trainer could really help physiotherapists and patients with treating dysfunctional breathing.

## 8. Acknowledgements

We would like to thank our supervisor Randy Klaassen for continuously guiding us during this project. More specifically, he helped us realize certain things we should be doing that we were not aware of, like the request to the ethical board. He also helped us in creating our poster, and he looked extensively over all the versions of the report we asked feedback for.

Besides our supervisor, we would like to acknowledge Melissa van Schaik and Sarah Pichon for helping us extensively in forming the requirements and for the initial research they did on the Wearable Breath Trainer. We also want to thank them for providing the Breathpal sensor and the vibration motors.

We would also like to thank Angelika Mader for writing an initial version of the code that reads from the sensor. This helped us get a running start in hardware development, for which we are very grateful.

## 9. References

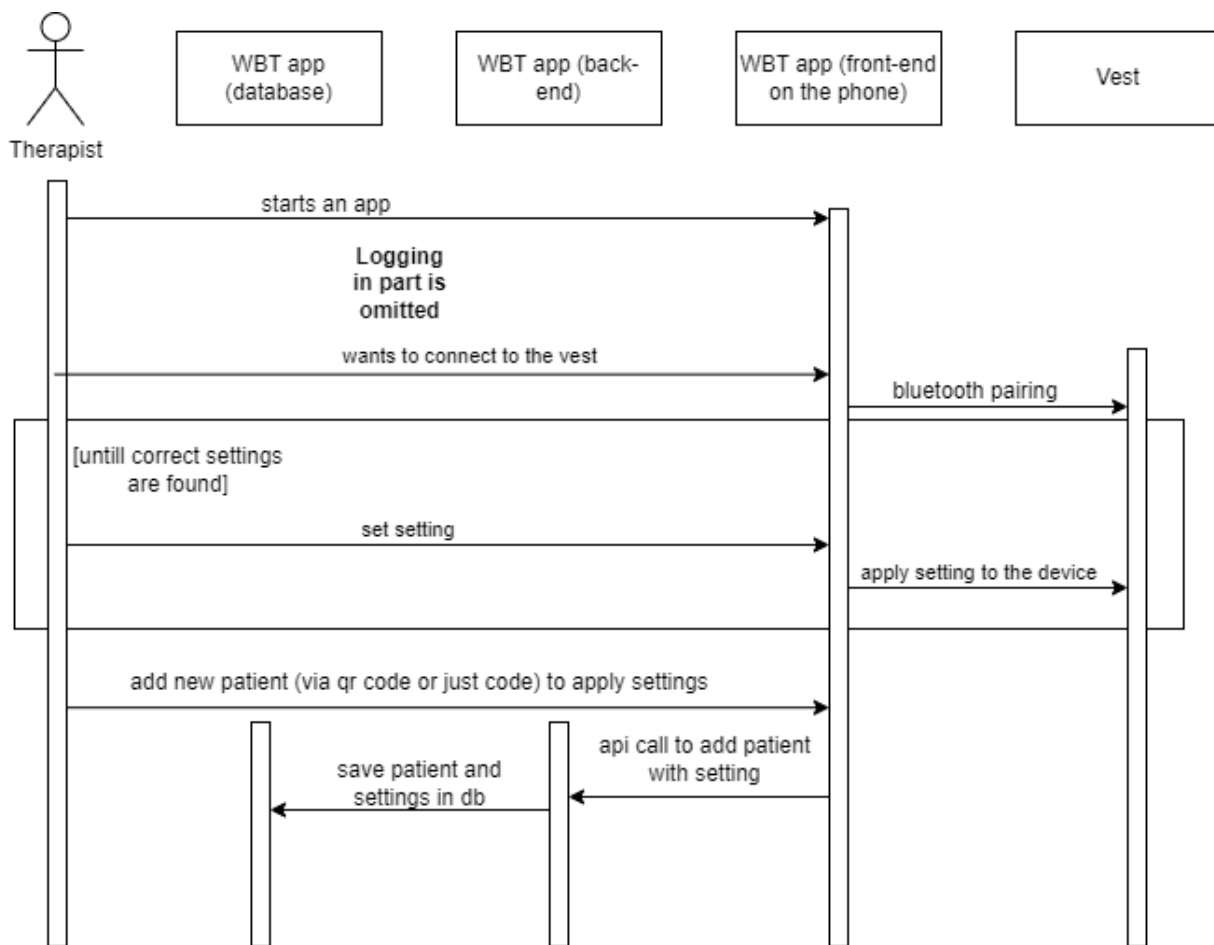
1. M.E. van Schaik & S.A.S. Pichon (2023) DESIGNING A FUNCTIONAL PROTOTYPE FOR THE WBT, Manuscript in preparation.
2. Ben Bulsink. (2020). Breathpal V3.1. Retrieved from <https://breathpal.nl/>
3. Dr. A.H. Mader, Angelika. (May 2023). BreathingSensor.v0.2.ino. [Computer software] University of Twente.
4. Juffermans, Ellen (November 2023). WBT project - Saxion.

# Appendix A: Additional Diagrams

During development we also made some diagrams to make our design more clear for our clients. Next to that, we used figma to visualize the frontend of our application in the beginning and to show what we were thinking of to the clients:

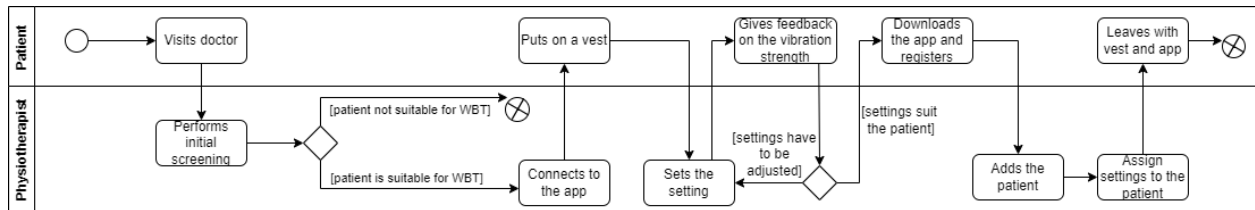
<https://www.figma.com/file/sMp7qMF8ICPrK8jU4K4NfW/WBT-app?type=design&node-id=229-3923&mode=design>

Figure 27 depicts the process of saving a patient's settings and sending it to their wearable.



**Figure 27** Sequence diagram of saving patient's settings

Figure 28 depicts the process of patients acquiring the wearable and using it.



**Figure 28** Activity diagram diagram of acquisition and use of the wearable

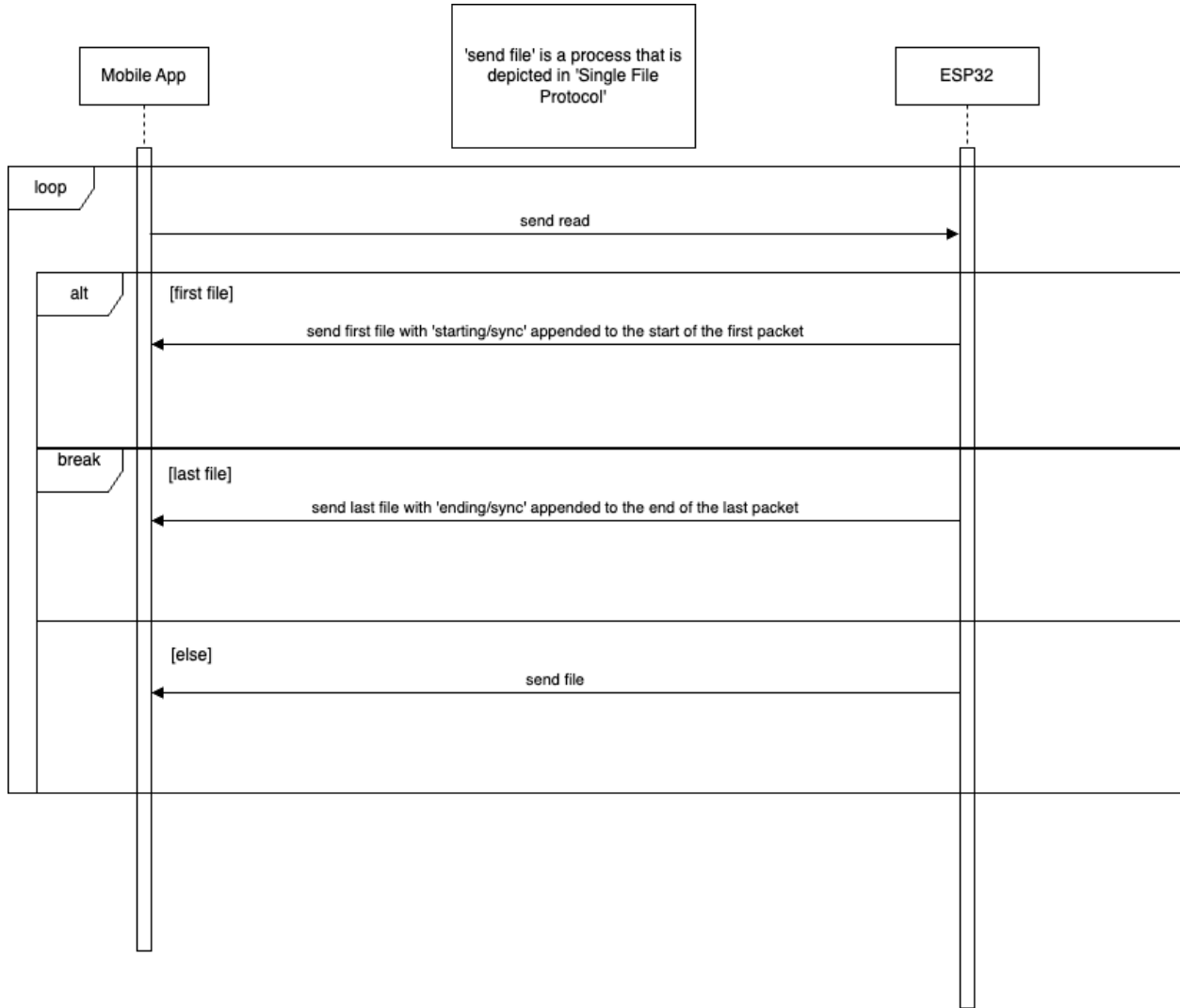
Figure 29 depicts the use case diagram of the system.



**Figure 29** Use case diagram

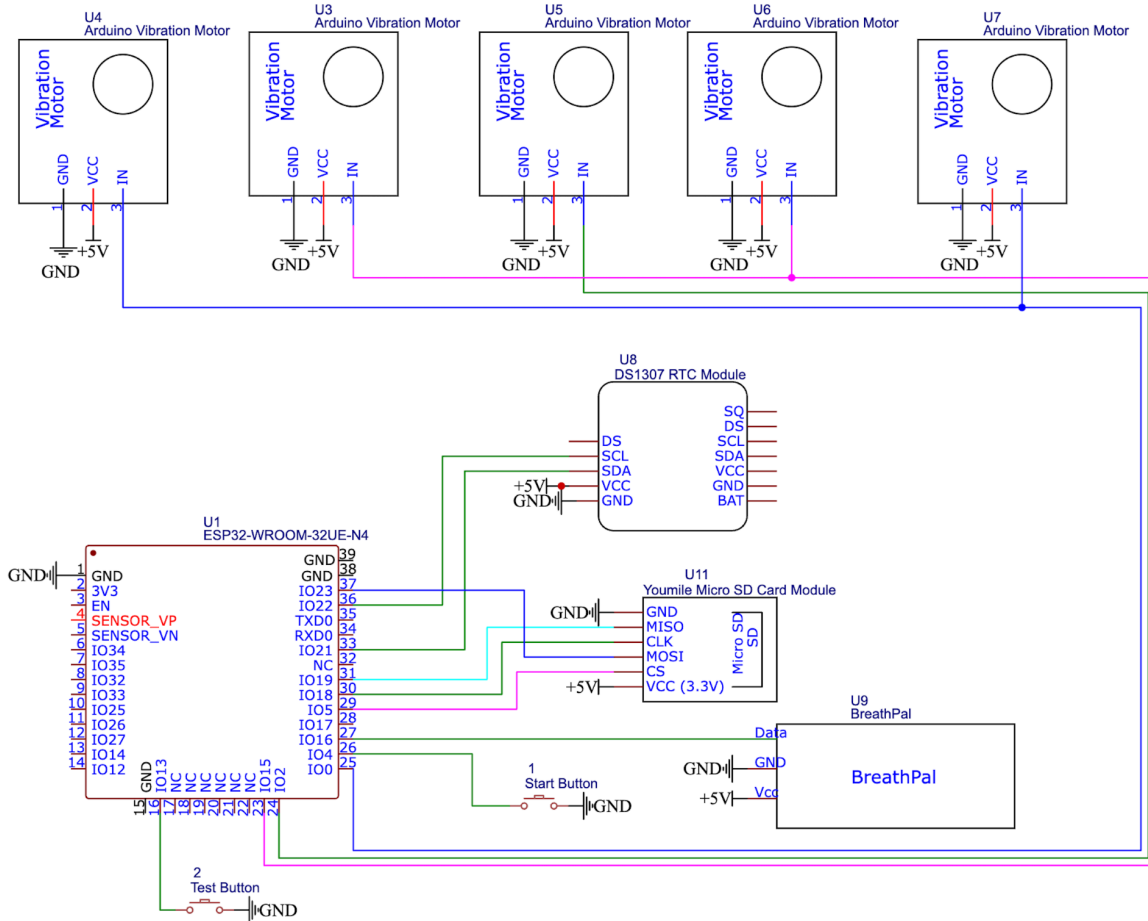


Figure 30 contains a sequence diagram depicting the protocol for sending over multiple files over BLE from the ESP to the application.



**Figure 30** Sequence diagram of sending multiple files

A wiring diagram that illustrates the separate components of the WBT, as well as the pins we used to connect them to the ESP can be found in figure 31.



**Figure 31** Wiring diagram of the system

## Appendix B: Wearable testing

In this section we will outline our system test for the hardware components of the WBT.

### Testing SD card reader

To test the SD card reader we used the example code from the following tutorial:

<https://randomnerdtutorials.com/esp32-microsd-card-arduino/>

### Testing RTC module

To test the RTC clock module we used the ds1307 example sketch from the RTCLib Arduino Library.

### Testing Breathpal

To test the Breathpal sensor, you need to connect the micro usb port of the ESP32 to your computer and find a way to read it's serial output. The way we did this was through the serial monitor of the Arduino IDE. Make sure the system is powered on, put on the wearable and press the start button. The serial port should start receiving readings in the following form:

```
13:01:55.367 -> data: 10667,9804,-37,21,1033
```

For an explanation of what each of the values mean please refer to section 4.4.2. Since the actual value of the sensor will be different based on the wearer, it is important to ensure that the abdomen and thorax values(the first 2) increase as you inhale(so the straps stretch out) and decrease as you exhale. After 10 seconds the device should start giving haptic feedback, which based on the feedback settings can be used to test the functionality of the vibration motors.

## Appendix C: User testing results

### Tasks:

1. Find the screen for creating a new account [mf3]
2. Login to the physio account with the credentials physiotherapist1 [mf3]
3. Visit a patient's profile
4. Connect to a wearable device[mf1]
5. Edit and save a patient's device settings [mf2]
6. Look at the duration overview of a patient [mf7]
7. Inspect a specific training of a patient
8. Delete a patient [cf6]
9. Look at the info page [sf10]
10. Create a new patient [sf2]
11. Link an existing patient using their qr code [cf9]
12. Change your phone number in your account details
13. Log out [mn4]

### Sarah

Task	Time	Extra Comment
1	30 sec	The visualization of the qr code is confusing. The physio might want to fill in their clinic. Back button
2	56 sec	Types slow. Show password is nice.
3	7 sec	Nice feature

4	20 sec	<p>It should update when the device disconnects because of an error. There should be more visualization on the device itself maybe.</p> <p>Add explanation for syncing settings at that page.</p>
5	1 min 30 sec	<p>It would be nice to have live calibration instead of sending over the settings first. The settings page is very intuitive, nice! Maybe change it so that there is also a page where you can see the current device settings instead of only being able to see it in the edit menu.</p>
6	37 sec	<p>Maybe not intuitive that it is the total for each week. Maybe it would be nicer to put the thorax and abdomen in the same graph. Explain what the percentages mean.</p>
7	7 sec	<p>It is intuitive to find the training. The drawings are nice. Nice! It would be nice to have the time in the training graph. In this way it would be easier to show that it is scrollable.</p>
8	Failed	<p>It is not very intuitive. But it looks nice</p>
9	3 sec	<p>The info page should be a separate page and then if you want an info icon it should be specific to the page. And then maybe put a general info page in the home page or something</p>
10	1 min 1 sec	<p>No comments, it is intuitive.</p>
11	30 sec	<p>Looks nice. Intuitive</p>
12	59 sec	<p>Maybe it is not necessary to fill in your password when changing your phone number, but it is good because you don't use it so often.</p>
13	4 sec	<p>Maybe make it harder to find or add a popup notification</p>

General comments:

What will happen with the project when we are finished?

- This project will end at the end of this school year. Our prototype will be used for the integrated version and we hope that we can write a new proposal for the follow up where we can improve the robustness and design of the application with more user testing with the physiotherapists and patients and we will check how visual feedback impacts the trainings.

Add a search field for the patients list

For the future it would be nice to add accelerometer data, to see whether a child was moving during the training and the data would be inaccurate at that point.

For the future it would be nice to export data.

If the treatment is done, you should probably delete the data and patient forever. But it is also still a bit of a question that needs to be asked to physiotherapists.

The haptic feedback is nice.