Design Project: Physical Therapy App Awear

Matthias Wentink, Niek Pennings, Pepijn Visser,
Rosan Maas, Rutger Witmans
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
@student.utwente.nl

April 24, 2023

Abstract

Anyone who has had some sort of therapy, diet or training goal knows one thing: consistency is key. Something small can make a giant impact if you do it consistently. However, we struggle with fitting this new consistency into our lives because it wasn't there before and it is yet to become what we call a habit. One way of helping people achieve this consistency is rewarding people for showing it, the gamification of simple tasks. This is what awear strives to do for physical therapy exercises at home.

Contents

1 Introduction

Because of heavy workloads, sports, born irregularities and many other causes, people go to physical therapists, personal trainers and coaches. These knowledgeable individuals can help them with strengthening their muscles, easing pain or optimising performance. Often, part of their help consists of recommendations for exercises that will help strengthen a part of the body. These exercises are usually explained and walked through at the appointment together with the therapist. They give you tips to watch out for things such as being aware to use your abs for something instead of your back, slightly bending your legs at an exercise etc. All these tips are with the goal of enabling you to complete the exercises at home as correctly as possible to help you to their full capabilities. However, physical therapists can not be present everywhere all at once and clients might not be able to remember all of the tips they have gotten or know that they are doing their exercises in the proper way.

To help with the scenarios illustrated above, awear has the vision of developing an application that can guide both client and therapist in this process. Together with the help of Movella DOT sensors, they aim to make an application that can guide the clients through exercises, giving them recommendations and helping them when doing their at-home exercises as well as allowing the therapists more insight. The goal is both to help and incentivise clients as well as offload some of the therapist's workload, making the whole process smoother and maybe even helping clients recover faster.

This vision requires some solid design work, not only to make sure that the chosen structure is representative and can provide all needed information in the best way possible but also to support the potential of the application. To support potential additions to the basic system. This means that the base of the application needs to allow this potential change and addition. The way this is regularly done is a modular base. Meaning that all system components and features are separate and only connected through the front-end and the central database. The main advantage of making a system like that is that you can easily swap out one feature for a different version of the feature, making it very resistant to technological advances and changing requirements for different features.

This was thus what our project goal consisted of: creating a good modular base for an application with the vision to assist physical therapy exercises.

In chapter 2, the domain in which the system is situated is analysed. In chapter 3, the different phases of the project are discussed, illustrating how these phases are used in the project and what each phase contains. In chapter 4, we illustrate how different kind of intermediate meetings have helped in the process and what added values they had. In chapter 5, the requirements specification methodologies are explained and the requirements are provided based on the needs of the involved stakeholders. In chapter 6, the results of the requirement analysis are provided and elaborated. In chapter 7, the design of the system is discussed, an overview of the system is given, architectural design choices are explained, and the purpose of the system is elaborated. In chapter 8, the approach of testing the system is described in an elaborate plan and its results are provided. In chapter 9, the future plans for the system are discussed in terms of important placeholders and future vision. Finally, in chapter 10, this design project is evaluated and its final conclusions are provided and discussed.

2 Domain Analysis

In this chapter, the identification of the domain of the system is discussed. The view of the client and users on a support system is issued from an 'outside-in' perspective [26], such that the existing problem is explained in more detail. By understanding the domain, the development will proceed more swiftly and it aids the planning for future development.

2.1 Introduction to the domain

The project is in the domain of physical therapy apps specifically in the field of physical therapy apps using motion tracking. The domain is broad as any kind of movement could be tracked, gamified, analysed, etc. This domain has some similarities with fitness apps as they both stimulate the user to become more physically fit through exercises in an app. However, the key difference here is that fitness apps do not have a limit to the amount and intensity of exercises whereas a physical therapy app is about rehabilitation, which means that the client should be kept in check to avoid further injury.

2.2 General knowledge of the domain

Many of us today have a good view on assisted workout applications, whether they help you to keep your motivation, give you exercise regimens or simply track statistics about your movements and workouts. Many other applications we also use, like Duolingo[1], are also good examples of using gamification to try and encourage people towards an educational or healthy goal. The important point to note here is to still understand the difference between the gamification and the goal of these other applications. Often they encourage you to keep going without limiting the amount of time or exercises you can do, something that is not as desirable in a physical therapy application. We also have a group member that has been to physical therapy before, also giving us insight into how these appointments and therapists work.
Furthermore, there are several apps for physical therapy like Physitrack [4] for example. What most physical therapy apps lack however is the therapist-client connection as well as the usage of motion sensors.

2.3 Client, Users and Interested Parties

Because of the broad application of the proposed platform, the list of interested parties is considerably large. People that do movements with supervision or vice versa can use the platform. Parties which might find this interesting are hospitals, physical therapists, sports trainers, and psychiatrists. These parties in turn are also wide-spanned themselves.

Hospitals could use the platform to make sure that patients get enough movement.

> "On average, critically ill patients lose nearly 2% of skeletal muscle per day during the first week of ICU admission." (Fazzini, 2023)[10]

By providing their patients with simple exercises such as taking x amount of steps per day, this percentage could be significantly reduced.
Physical therapists help their patients in resolving injuries through exercises. For example, there is a patient who has dislocated their knee in the past. A physical therapist can assign them exercises to strengthen their knee and look at the progress that the patient is making at home. In this way,

they can assure that the patient is doing their exercises and that they are doing them in a good and safe manner.
For sports trainers, there is a myriad of sports examples to choose from. One would be rowing, a sport which depends on repetitive but important movements. By having the trainer track the arms of one of their rowers during an exercise, they can analyse this movement and train to optimise it further, thus getting more out of their rowers.
A psychiatrist could help their patients rehabilitate after an accident or sickness. For example, helping people walk again after a severe stroke.

2.4 Software environment

The previous implementation by awear used Flutter. It managed to track the movement of a user wearing two Movella DOT sensors and plot information from these sensors within an app. These sensors capture three types of information. It has a three-axis gyroscope, accelerometer and a magnetometer. The previous implementation already processed this information in a graph. This meant that this was not in the scope of this project.
For the current implementation, it is important to expand upon this functionality which is why Flutter was used again. The previous implementation had no connection to a supervisor side yet, which requires a back end. For this, GO and Swagger have been chosen along with a PostgreSQL database. Further elaboration on these software environments can be found in 7. As mentioned, the platform should make use of movement sensors, for this project specifically, Movella DOT sensors.

2.5 Current status

Motion sensing for rehabilitation and physical therapy has been a field of research that has existed for a considerable amount of time now. A distinction here can be made in optical motion capture and motion tracking through sensors placed on the body. Optical motion capture often times records both colour and depth but accuracy-wise and there are some available and easily adaptable ways to do this such as using the Kinect[15]. However, even with this depth sensor, optical motion captures different and perhaps less accurate data than motion tracking through sensors on the body.
Nevertheless, wearable sensing technology also has significant hurdles. Whilst, there is a lot of promise in research for wearable sensing technology in rehabilitation and physical therapy, actually putting it into practice in a widespread manner is not possible yet according to Lang et. al[17]. However, in the same paper, it is discussed how physical therapy and rehabilitation using wearable sensing technology is possible in the foreseeable future and it is elaborated on how this can be achieved through meeting certain benchmarks and overcoming corresponding hurdles. The benchmarks are as follows:

- Convenient for purchase & use
- Minimal time burden
- Easy & comfortable to wear
- Provides accurate, reliable, valid & responsive information in a user-friendly format

Though the proposed implementation will be a modular stepping stone and cannot possibly satisfy all of the mentioned benchmarks, it does satisfy some of the benchmarks and will cause significant

progress in the other benchmarks. To start, the Movella dot is relatively affordable and its size, Bluetooth capabilities and SDK make it convenient to use. Furthermore, by having a modular system, other sensing wearables could be incorporated into the system. This would imply that, whenever a more affordable and usable sensor hits the market, it could be implemented making the system even more viable.
The connection with the Movella dots does not have a large time burden and the system could be designed to have a UX that minimises the time the user spends setting up the sensor.
As mentioned before, the size makes the DOTs easy and comfortable to wear. Furthermore, it has a waterproof rating of IP 68, meaning that it is resistant to water up to 1.5 meters deep for up to 30 minutes.
The information in a user-friendly format is something that can be achieved by selecting the most important data to view, and visualising it to make it comprehensible for the user. New modules can also be incorporated to analyse data further, for example by using artificial intelligence.
All of this suggests that the proposed project could be a viable product as a stepping stone in the domain of physical therapy apps as well as other movement-tracking apps.

2.6 Conclusions

This section explored the domain of physical therapy applications, fitness applications and gamification. Especially physical therapy with movement sensors has been analysed in the paper by Lang et al. [17] which has been compared to the current situation. All of this is used as a basis for the requirements of the final implementation of the platform which are discussed in section B.

3 System Phases

3.1 Requirements

During the first meeting, it became apparent that the company did not have a specific cause for the design project in mind. Instead, the project purpose could partially be filled in by the project group. At that point, a modular stepping stone for a supervisor-client platform in relation to motion tracking was decided upon. Following a domain analysis and meetings with the supervisors from awear, a scope was determined and requirements were set up. These requirements have been through a feedback round and finalised. These requirements are split between user requirements and system requirements. As requirements can change during design, programming or testing, they were put in a comprehensible table showing how the requirements were updated and in which system phase this occurred. Furthermore, the requirements have been put in a user scenario. This user scenario puts the requirements into context and can be also be used in usability testing as will be discussed in section 8

3.2 Mock-up

For the system, some diagrams were set up as to explain the components of the platform and how they work together. This overview of components can be seen in figure 2 and the underlying database that guides the structure can be seen in figure 3 there are also two diagrams describing the flow of the client application 4(a) and the flow of the supervisor application 4(b). After the architecture and purpose of the platform were determined, mock-ups were made for the different screens and modi in the application both for the supervisor as well as the client. During, development requirements changed and as such, some mechanics in the system changed. Because of this, diagrams were adapted to better fit the new implementation and new screens were designed to resolve problems that occurred with old screens.

3.3 Development

In the development phase, the actual system was developed. This was the largest part of the project. Some of it happened simultaneously with the Mock-up phase. This is also where we ran into the most unexpected problems. As this was by far the largest part of the project it is not weird that this is also the part where our time plannings were off the most often, though as we had taken some time to look into the systems in advance, none of these plannings differed too greatly from the original one. During Development, the Mock-ups as well as the requirements were the guidelines for implementation, some extra added ideas from other meetings were always noted some times taken into account if the added value was high enough to justify the extra workload. This ended up only happening for a few smaller additions that we felt added a substantial value.

3.4 Testing

During the entirety of the project, a testing plan was drafted consisting of unit tests, user tests and a postman test suite. This testing plan was executed in the testing phase. The results of the tests could not be incorporated further into the final system but have been noted down in the section 8. They can be used to further improve the system in the future.

3.5 Final system

The system has been completed in the final week of the project. Obviously, the system that was developed in this project serves as a stepping stone and like most software, this is not actually a final system. Instead, it can be developed further from the current MVP that it is to a full product. It is important to note that 'final' in this case means that the MVP of the project was done and was ready to be presented. In this phase, a poster was developed illustrating the key features and concepts of the platform. This poster was presented along with the posters of all the other projects. Furthermore, a presentation was held to our supervisors separated in two different session: one session with our supervisor from the university and one with the two supervisors from the company. The latter had to be planned after the project deadline, because of planning reasons.

4 Meetings

In this section, the different meetings that were held during the project and how they influenced the project itself will be discussed.

4.1 Stand-up meetings

These meetings were our internal meetings with the team that we held in the morning to get an overview of how everyone was doing, what they were working on what the status of the project was. In the beginning, these were more sporadically once or twice a week with small meetings in between if certain features required collaboration or if we had a meeting or deadline coming up that required us to walk through specific points. Often, the Trello that we made to keep track of the tasks was used to guide us through these meetings.

4.2 Peer review meetings

These were meetings that we had once every two weeks together with Rom and about 5 other project groups, these meetings were a good method to get some feedback and some outside thoughts about the project from a group of people that did not know all the details but only what we showed in our small presentation. This gave us a better insight of what they found to be important information.

4.3 Client meetings

About every 2 weeks we had a meeting with our client. Here we showed him mock-ups and pitched ideas on how to handle certain things. We had the pleasure to work with a client that gave us a lot of design freedom and mostly asked us to justify and properly explain why we chose to do it that way, not only sometimes making us think about choices a little deeper but also allowing us to execute everything in a way that we saw as best. He was also open to brainstorming with us about new features we thought would be a good addition and talking about the final vision for which we were building the base.

4.4 Supervisor meetings

Because our client is an external party, we had to find another supervisor within the UT. We opted for someone from the Human Media Interaction group. We chose this because we believed that building the system itself would not be our biggest problem but building it in such a way that it would benefit the purpose the most. As this purpose had a lot to do with the interactions with the users and their experience with both the exercises as well as the rest of the system, we opted for the HMI group. In these meetings, we often indeed spoke with our supervisor about what to look out for in such a system with sensors and user interaction as well as that he assisted us a lot with making sure that our presentation of the system was clear in the forms of visual overviews and different diagrams to simulate flows through the system. This is especially important in our project as this is a system that other people will need to continue with and thus requires good documentation not only of our code but also of our system architecture and design choices.

4.5 Conclusions

These various kinds of meetings helped us not only keep track of our own progress but also provided meaningful insight from different people, experts, students and owners to help us see viewpoints or issues that we may not have spotted on our own. Especially because we had a system where user experience is very important, the different insights were incredibly useful.

5 Requirements Methods

In this section, we will describe our methodology for setting up the requirements. We started with a base set of requirements, some requirements could have been added later due to various reasons. However, we found that because of our research into the technologies and due to the time spend to set up and discussing the requirements with all parties, they ended up not changing much.

5.1 Requirement formulation

The requirements are formulated using SMART guidelines [20], these guidelines aspire that the formulation makes the requirements Specific, Measurable, Acceptable, Reasonable and Time-bound. These five aspects make sure that the interpretation of the requirements cannot be biased. This kind of formulation of requirements is important in the testing phase so there is no doubt about whether or not a requirement is achieved.

5.2 Requirement Prioritisation

The requirements that we formulated with the previously mentioned SMART guidelines still require prioritisation. This is necessary to set priorities throughout the project and is determined together with stakeholders during the requirements phase. We show this prioritisation with the help of the MoSCoW method [19]. This means that the features defined in the requirements are identified by Must, Should, Could or Won't. indicating the priority of the requirement.

5.3 User Requirements and System Requirements

In order to show both the view of the user and their wants/needs as well as more specific specifications only seen from a system point of view and not necessarily experienced by the user, we formulate the requirements in both user stories as well as system requirements. The second set is in MoSCoW format and shows the developer views and the user stories will be used for understanding where the system requirements came from as well as user testing.

6 Requirements

Here we specify and analyse all requirements. We do this at two levels: User requirements and system requirements. In appendix B you find an extended table with links between all user and system requirements as well as extra information regarding the origin of the requirement and the measures that will be used to specify whether or not a requirement is met. Below you see all requirements and any further specifications if those came up during implementation as well as a note if a requirement ended up not being implemented, the explanation for this can be found in chapter 7.2.8 and 7.3.6. All features that did not get implemented are still possible to easily be implemented later due to the modular structure but are parts that ended up falling out of the scope of our project. How this could possibly be done is detailed in section 9.

6.1 User requirements

1. As a supervisor I want to see all possible exercises

   To be able to select and assign exercises, the supervisor needs to be able to see all exercises. This feature also includes a filter for easier searching

2. As a supervisor I want to add new exercises to the

   To make the system modular, supervisors are allowed to add new exercises

3. As a supervisor I want to assign exercises to a client

   After adding the necessary extra exercises, the supervisor needs to be able to assign exercises to a client. This can be done in batches, here the supervisor also adds a start and end date and how often the exercise needs to be done

4. As a supervisor I want to see the progress of a client

   To be able to follow the progress of the client, some statistics are shown to the supervisor for a client. These statistics could be any kind of data we have stored in our modular system

   This feature did not get implemented, but a placeholder is present

5. As a supervisor I want to see an overview of all clients

   To be able to have an organised overview of all statistics, the supervisor will also need an overview of all clients

6. As a client I want to follow the assigned exercises

   The client needs to be guided through an exercise with regards to sensor placement etc as well as the system knowing that they have completed the exercise

7. As a client I want to see additional information about my exercises (how many times, how often etc).

   This is information added by the supervisor when they assign the exercise, on the client side this will show in a to-do list kind of format

8. As a client I want to follow a clear explanation of how the exercise is done

   An explanation can be added to each exercise, detailing the idea again. Of course, the exercises will also still be explained by the therapist the first time

9. As a client I want visual aid for the exercise explanation.

   To accommodate different memory styles we have added the opportunity to add pictures to exercise explanations

10. As a client I want to have a page where I can see statistics on my progress.

    This is similar to the supervisor statistics page and can be implemented with the same history data

    This feature did not get implemented, but a placeholder is present

11. As a client I want to mark an exercise if something is unclear and provide feedback

    This is to make sure that the quality of the exercises is good even though we allow any supervisor to just add them

12. As a client I want to get some type of reward for following the exercises (points, confetti)

    To incentivise clients we give points and have features like confetti popping up when you finish your exercises

13. As a client I want to get a notification for doing my exercises.

    It would be nice to get a notification for when things are ready as a reminder and another incentive to keep going

    This feature did not get implemented

14. As a client I want to get a notification if new exercises get assigned.

    It would be nice to get a notification for when things are ready as a reminder and another incentive to keep going

    This feature did not get implemented

15. As a supervisor I want to get a notification of marked exercises with feedback to improve it

    It would be nice to get a notification for when things need to be adjusted

    This feature did not get implemented

6.2 System requirements

1. The client-side app must be easy and enticing to use

   As we want to incentivise people to keep using the application regularly, it is important that the aesthetic and usability do not deter people from usage

2. The system must have a mechanism which enables clients to log in and out of the application.

   To be able to distinguish between people and provide privacy, a log-in system is necessary

3. The system must have a mechanism which enables supervisors to log in and out of the application.

   To be able to distinguish between people and provide privacy, a log-in system is necessary

4. The system must have a mechanism which enables clients to specify what supervisor they belong to.

   To be able to assign exercises to a client as a supervisor, and thus to be able to know which clients belong to which supervisor, the system will have a simple initial linking system to store what supervisor the client belongs to

5. The system must have a mechanism which enables supervisors to add exercises to the system.

   To keep it as modular as possible, we allow supervisors to add exercises to the system. There will also be a feedback loop in place to ensure the quality of the exercises

6. The system must have a mechanism to display all exercises to supervisors

   To assign exercises, the system will show the full database of possible exercises to a user with supervisor rights

7. The system must have a mechanism which enables supervisors to assign exercises to a client.

   The supervisor will be allowed to assign exercises to a client in all logical places of the system, these exercises can be grouped by muscle group or other labels. The supervisor will also be able to see what exercises they assigned the client in the past, present and future and potentially edit these assignments

8. The system must have a mechanism to save data from the performed exercises for analysis

   Because we do not only want to show recommendations while doing the exercises but also be able to see statistics, not only to motivate the client but also to give the supervisor more insight, the movement data from exercises will be saved with relevant meta-data

9. The exercises must have visual aid as part of the explanation.

   To accommodate different memory styles we have added the opportunity to add pictures to exercise explanations as well

10. The system must have a mechanism that allows data to be able to be qualified as the correct exercise or not with the help of example data.

    This is used as a placeholder for the analysis and recommendations. In the end we decided to implement another placeholder but this would still be possible with the modular base

    This feature did not get implemented, but a placeholder is present

11. The system must have a mechanism to add feedback about the clearness of an exercise.

    This is part of the feedback loop to enable supervisors to add exercises to the system and guarantee their quality

12. The system must have a mechanism to visualise the client's data to see the progress.

    To allow the supervisor to keep an overview of their clients progress as well as allow the client insight on their progress statistics can be shown from the various meta and movement data

13. The system must have input sanitation.

    Because we allow supervisors to enter data into the system as they please, we will need input sanitation to make sure that this user input does not allow for malicious behaviour

14. The system should show an explanation of the exercise beforehand.

    To make sure that the user knows what they are getting into and to properly guide them through the exercises we give them a good introduction and guidance

15. The system should allow for a point system.

    To incentivise clients we give points and have features like confetti popping up when you finish your exercises, this point system can later allow for more elaborate gamification and other forms of incentives

16. The system should have a mechanism to evaluate how the exercise went for the client.

    To allow a form of data feedback that focuses more on how the person feels about their progress instead of the supervisor only seeing plain data

    This feature did not get implemented

17. The process of connecting and calibrating the sensors should be as smooth as possible.

    To ensure the best user experience we want to make the calibration and connection as smooth as possible, this could later easily be adjusted and/or changed

18. The system should be able to give a notification to the client when their exercises are ready to be performed.

    To add another incentive to keep up with the exercises, the possibility of notifications for released exercises is added

    This feature did not get implemented

19. The system should be able to give a notification to the client when new exercises are assigned to them.

    To add another incentive to keep up with the exercises, the possibility of notifications for newly assigned exercises is added

    This feature did not get implemented

20. The system should be able to give a notification to the supervisor when an exercise they added is marked as unclear.

    to make sure that the exercises have an appropriate quality, the supervisor can also edit the added exercises to improve them when someone has given feedback, as this only happens sporadically, a notification could be useful

    This feature did not get implemented

21. The system could have a mechanism that compiles a weekly review for the client.

    An extra feature that could be useful to keep track of different weeks and have reminders

    This feature did not get implemented

22. The system could have a mechanism that compiles a weekly review for the supervisor.

    An extra feature that could be useful to keep track of different weeks and have reminders

23. The system could have a mechanism that emails the weekly review to both the client and their supervisor

    An extra feature that could be useful to keep track of different weeks and have reminders and keep them on another platform as well

    This feature did not get implemented

24. The system won't have a mechanism to show all exercises to the client.

    As we only support physical therapy exercises, it is essential that the client can only see and do the exercises currently assigned to them

7 Design

This section describes all the design choices made in the project. These design choices include global design about the structure of the platform and tools used, client application screens and supervisor application screens. The global section design includes diagrams describing the overall system. The other two sections include the designs for the screens.

7.1 Global design

7.1.1 Preliminary design choices

For this project, the material design system will be used which is an open-source design system by Google. Material design can easily be implemented in Flutter. Furthermore, there is an extensive Figma kit for material design which allows for rapid prototyping and designing before implementation. Aside from its ease of use, material design is a very modern design system that people are generally used to as it is used in every application by Google among others. Material design also has a rich collection of icons that are easily recognisable.
The colours that will be used are derived from the colours used by awear on their public website. The primary colour was taken directly from the site, and the secondary and tertiary colours are derived from the primary colour, having a more orange tint. These colours have been adapted to the material colour scheme that has four main colours (i.e. primary, secondary, tertiary, and error). For these colours, the colour of the content on those colours is also in the colour scheme, denoted by on_[colour name]. By using a colour scheme, the colours can be easily changed later. As the

Figure 1: Colour scheme

goal is to create a web-based and mobile-based application, we will use Flutter as a development kit to implement both applications. Flutter has a ready-to-use framework that is designed for cross-platform development, which enables cross-platform development for both web and mobile applications. Flutter can easily make API calls to our back-end that deals with both the database data, calculations and user information. Another advantage that Flutter provides over other options is the fact that Dart, the language it uses, is similar to languages we are familiar with like Java and Python. Finally, since another person has already created functionality with the sensor using Flutter, choosing Flutter allows us to use this functionality in our system.
For the back-end, we have decided on using Go with the Gin framework. We have decided on this back-end framework and language because we would like to make the system we are designing future-proof. Go is written to write concurrent fast code and since programming is heading in this direction for performance, we think it is nice that this language supports concurrency as well as it does. Furthermore, we want to make extending the application an easy-to-do process. Go provides a Python-like syntax. This allows most people to be able to understand and further add onto the back-end. We chose Gin because it allows for extensions such as ORM database interactions. These interactions make it easy to interact with the Postgres database connected to the application. Furthermore, we can easily add documentation to the back-end API by using swaggo, a library which created swagger documentation to explain which endpoints exist and what to expect from the endpoints. We think that these features combined will make for a smooth development cycle and a nice experience interacting with this REST API. The back-end application and the database will be running in two docker containers which can be set up using the appropriate docker-compose file. We do this to make sure any system could run this back-end by getting rid of compatibility issues.

7.1.2 System overview

Diagram 2 visualises the platform components and how they relate. The system consists of an application for supervisors and for clients. The two communicate with one another through a central rest API which connects to a database. Both applications make use of a movement sensor. In the case of the current implementation, these are Movella Dot sensors. These sensors communicate to the applications through a buffer, so that they do not get overloaded.
In diagram 2, an abstraction of the database structure is shown. The database is described more extensively in diagram 3.
Apart from the structure of the platform as a whole, we also needed to get a better idea of how the individual applications should work for the user. Because of this, flow diagrams have been made that describe the different modi of both the client application 4(a) and the supervisor application 4(b) and how to navigate between said modi.

7.2 Client application screens

The client application aims to guide users to complete exercises and reward them through gamification. The most important thing is to provide incentives to the user. This is done through motivating statistics, achievements as rewards and a clear guide on exercises. The following subsections explain the layout of the screens and how their design helps to accommodate this purpose. The last section describes the discrepancies between the initial design and the actual implementation and why it was done in this way. Each section includes a picture of the application with annotations. These annotations are letters from the alphabet in dark blue pointing towards a section or a certain component. These sections are referenced in the text between brackets and in

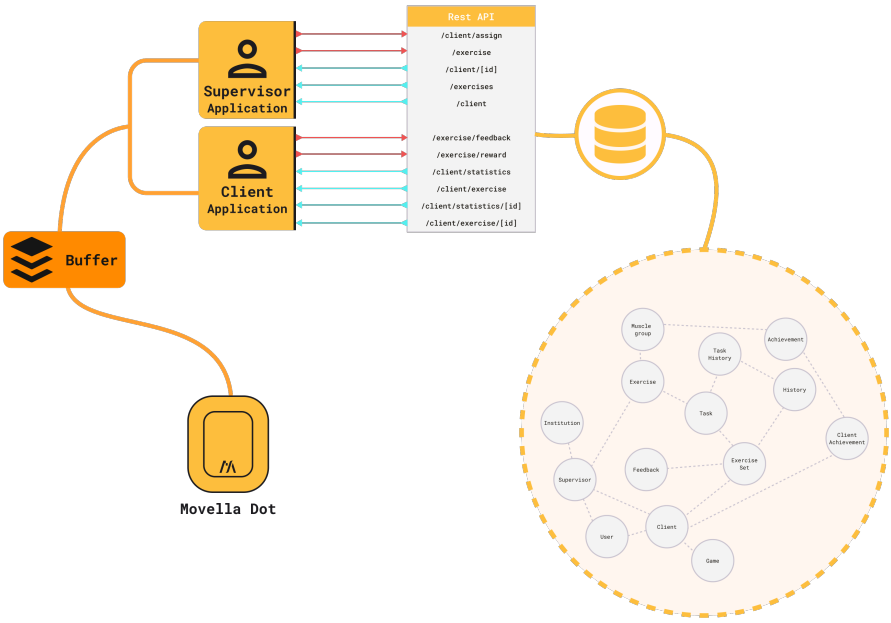


Figure 2: Overall diagram explaining the architecture of the platform

bold. In every section, the alphabetical count resets meaning that the first annotation in each section again starts with A and that references to an annotation are limited to their section.

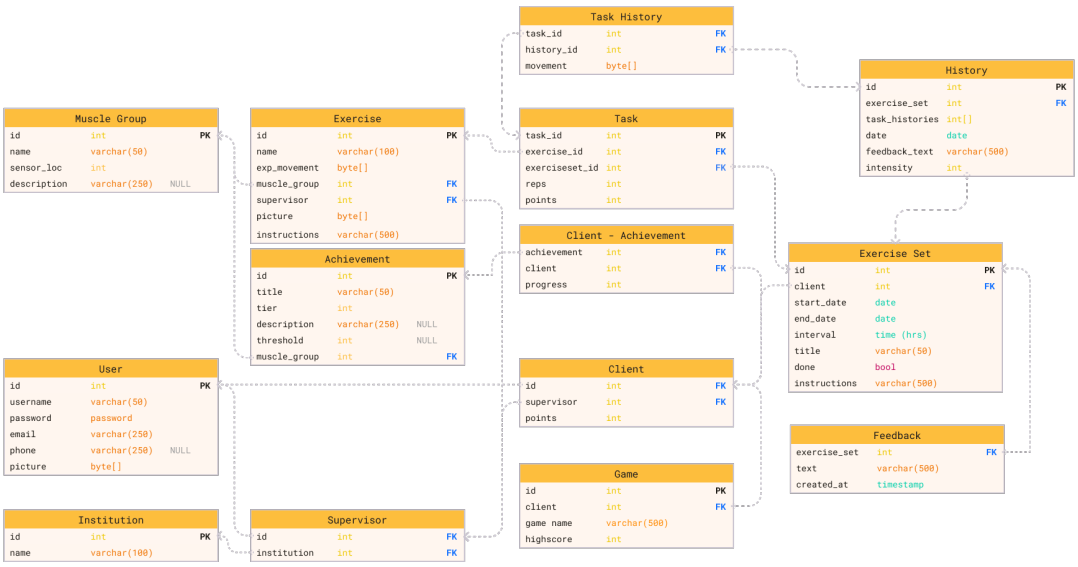
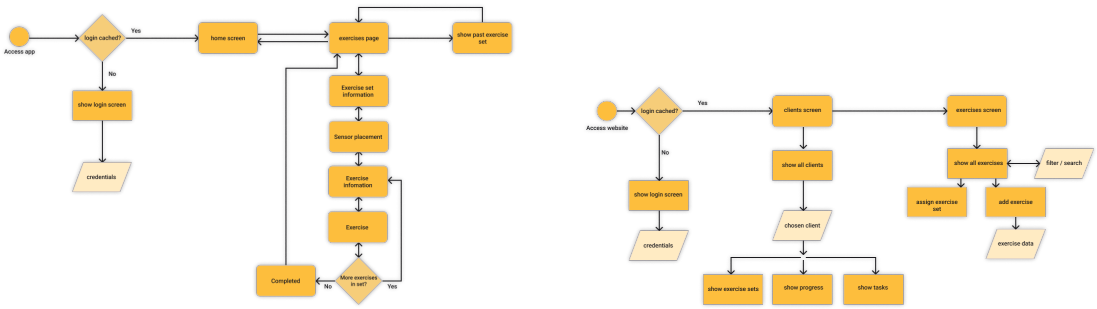


Figure 3: Database diagram

7.2.1 Log In

The splash screen is the first thing that the user sees. It consists of a little animation of the awear logo that symbolises the dynamic quality of the app. Additionally, the time of the animation can be used to load certain aspects of the app if needed.
The login is done through Google single sign-on as this provides an easy way for people to log in and makes it so that we do not have to implement account security ourselves which will save time but will also make for a safer application and platform.



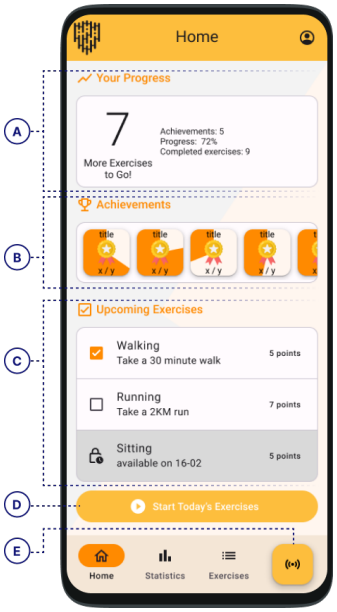
((a)) Client flow ((b)) Supervisor flow

Figure 4: Caption for this figure with two images

7.2.2 Home Screen

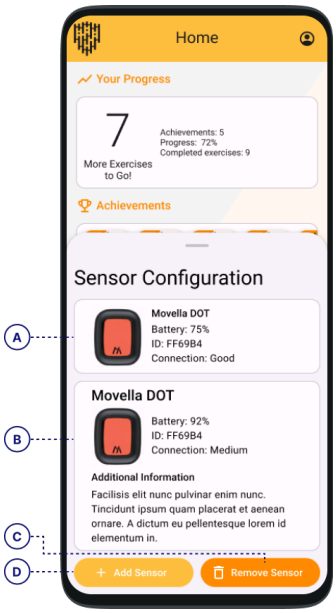
The home screen is the landing page for the app. The home screen contains an overview of the main features of the app. An important aspect of the app is the incentive/gamification that is given to stimulate the user to do exercises.

To support this concept, the progress tile **(A)** is shown as the first element on the home page. This displays the user's progress and gives them a good overview of how they are doing. The next element **(B)** is a row of achievements. This reminds the user of all their achievements and informs them how close they are to their next achievement. Hopefully giving more incentive to try to complete the achievement. The element beneath it **(C)** gives an overview of upcoming exercises. This gives immediate information to the user regarding their next task. One could argue that this is the most relevant information and should thus be at the top. However, we believe it is better placed underneath the achievements. This is because we first give the user information about the achievements and then what exercises are related, consequently first giving the incentive and then showing them what they can do towards it. Lastly, after all the information is given, there is a button **(D)** as a 'call to action' for the user to immediately start the exercise. On this page, similar to the other pages, a floating action button **(E)** can be found for the sensor configuration. The next section will explain the design of this sensor configuration.



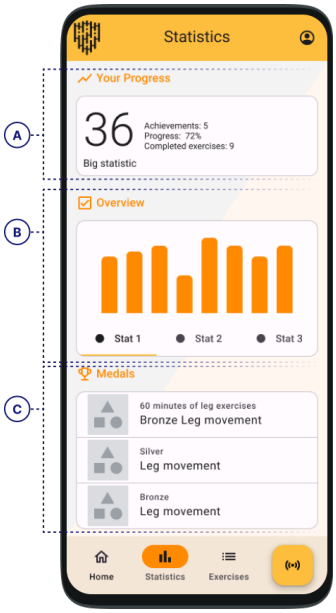
7.2.3 Sensor Configuration

As mentioned, the sensor configuration can be done by clicking the floating action button. This will open up a bottom sheet listing all the sensors and relevant information. By default, these sensor cards show an icon, the title and some information like, battery, id and connection like in **(A)**. When a card is clicked, it shows additional information, like in **(B)**. For now, a lorem ipsum placeholder has been made for this section. Two buttons are included at the bottom of the list. One for adding another sensor **(D)** and one for removing a sensor **(C)**.
This overview is quite important as it is directly supporting the main feature of the app: following exercises. Therefore it should always be accessible, hence its location in the bottom app bar.



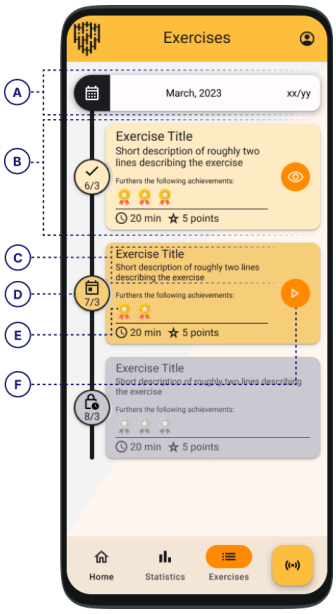
7.2.4 Statistics Screen

The statistics screen consists of three parts; a progress card **(A)**, a tab with statistics **(B)** and a list of medals **(C)**. The progress card is the same as on the home screen. The tab displays different statistics of the user depending on which tab they selected. All the available achievements and statistics are currently undefined and outside of the scope. Nonetheless, the placeholder is there in the design.
In our design, there is a graph of possible statistics in **(B)**, think about exercise frequency. The idea is that it gives the user more inside into their exercises. This way it can stimulate them if there is a clear progression or it can affirm them their progress. These statistics can also be shown on the supervisor's side to give the supervisor some additional insights.
Lastly, there is the medals/achievements list **(C)**. This shows the achieved medals. Currently, this is at the bottom because it gives the least amount of information and requires more interaction.

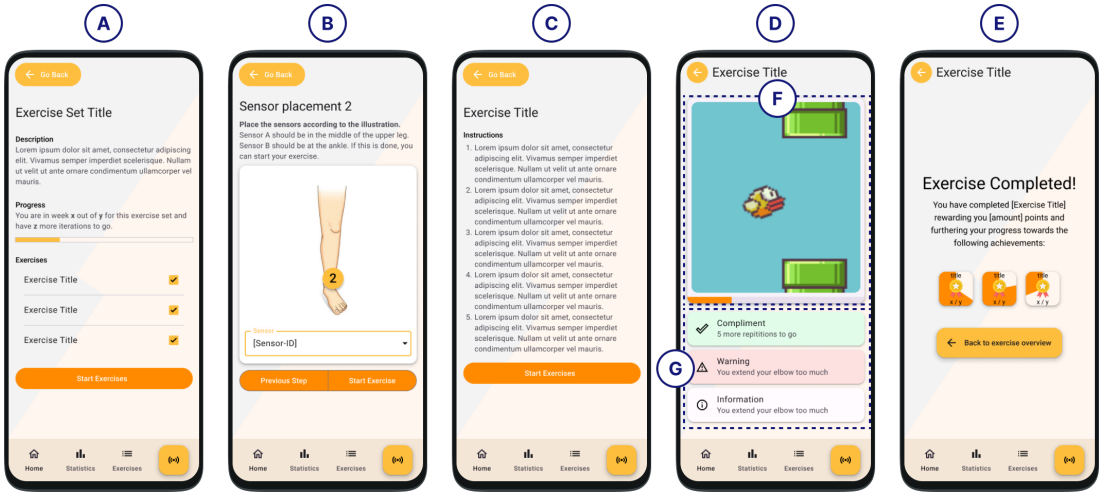


7.2.5 Exercises Screen

The exercises page is the main feature of the application. The screen shows a timeline of the upcoming exercises. This timeline consists of two types of blocks, a month block **(A)** and an exercise block **(B)**. The month block **(A)** consists of the month and year as well as a progress indicator for that month. The exercise block **(B)** can be three different colours indicating the status of that exercise. It is either a light yellow for completed exercises, a more orange colour for exercises yet to be completed, and grey for exercises that are not yet available. The status is also displayed in the icon in front of the block **(D)** along with the date. In the block, a title and explanation are present **(C)** that quickly gives the client an impression of what the exercise is about. Furthermore, they can see in which achievements they will advance in **(E)** and the exercise can be started immediately by clicking the play button **(F)**.



7.2.6 Exercise Walkthrough



The exercise walkthrough consists of five screens. These are the explanation of the exercise set **(A)**, the sensor placement **(B)**, the exercise explanation **(C)**, a game **(D)**, and a completion screen **(E)**. The sensor placement **(B)** is done in two screens for two different sensors. The location of the sensor is explained both in text and in an image so that there can be no confusion about where the sensor should be situated. The sensor can be selected from a drop-down menu. This screen and its contents are repeated for the other sensor.
The actual exercise screen **(D)** consists of a screen **(F)** and notifications **(G)**. This setup is chosen because it is versatile yet clear. The 'screen' can display a game or visualisations controlled by the

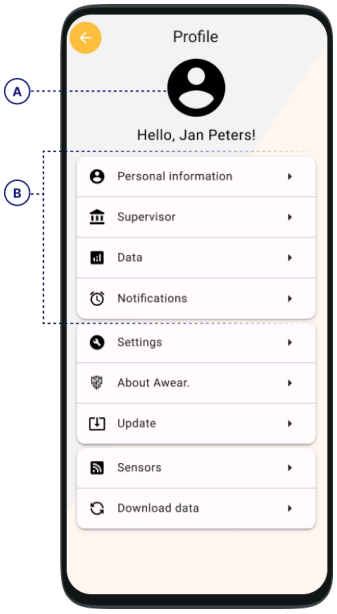
movements of the client. At the bottom of the screen, a progress bar can be displayed. Under the screen, notifications are displayed that provides the user with real-time feedback. The two are separated so that important messages do not get lost in the game the user is playing. There are three types of messages with distinct visual features:

- **Progress**
  This message is in green and denotes encouraging messages to the client such as how well they are doing or how long they still have to go
- **Warning**
  This message is in red to signal danger. For example, movements that should not be made.
- **Information**
  This message is in white to exude neutrality. It will provide information like what you are actually training with the exercise or other facts.

The exercise explanation **(C)** and exercise screens **(D)** will be repeated depending on the number of exercises within an exercise set. After this is done, the completion screen **(E)** will be shown. It congratulates the user on completing the exercise set and shows the progress that the user made in their achievements.

7.2.7 Profile

The profile page is a combination of an information and settings page. The page starts with a picture of the user and a welcome message **(A)**. The application really focuses on personal achievements and as such this personal touch is relevant. The information pieces are grouped together in a logical sense cut-off by a box **(B)**. This creates a logical divide for the user and allows for quicker navigation. Expected information will appear more at the top and more setting-like pages will appear more at the bottom. There are no designs for the individual parts on this page, as there is no need for them yet. However, a mock-up for a profile page seemed necessary as there inevitably will be information that needs to be stored therein for further development of the application.



7.2.8 Discrepancies between design and execution

The design that was made does not fully correspond with the final implementation. This section explains in detail why this is. The login screen does not have a sign-in with Google button but instead, it has a dropdown to select which client the user wants to act as. The authentication did not end up working but because it was essential for the app to have multiple different exercises depending on the client, these were visualised in the app. In the profile page, there is also a log-out

button as this was forgotten in the initial design. A design has been made for a general information page, but as there is no actual general information to tell and it is low priority, this was not made in the final system. The statistics page is missing because the priority was lower than the exercise page. In the end, not enough time was left to also create a function statistics page. The exercise page looks roughly the same except for the fact that there is no block displaying the month.
For the exercise walk-through, there are some minor differences. The sensor placement has some differences in dimensions and padding of elements but the main idea is the same. In the end, no actual game was implemented so that screen looks very different. Instead, it just graphs the sensor information. Furthermore, when an exercise is completed, there are no achievements yet.
Despite the difference, most design choices are still possible and the values of all components stayed the same.

7.3 Supervisor application screens

The supervisor application aims to provide all the information that a supervisor might need to help their clients without becoming too clouded. Because of this, there are only two real modi to the application after signing on. Namely, a client page and an exercise page. All important actions are made clear through floating action buttons and the use of the primary colour. The following subsections explain the purpose of the screens and how their design helps to accommodate this purpose. The last section describes the discrepancies between the initial design and the actual implementation and why it was done in this way. In the appendix, pictures of the design can be found annotated with design choices for specific areas.

7.3.1 Login screen

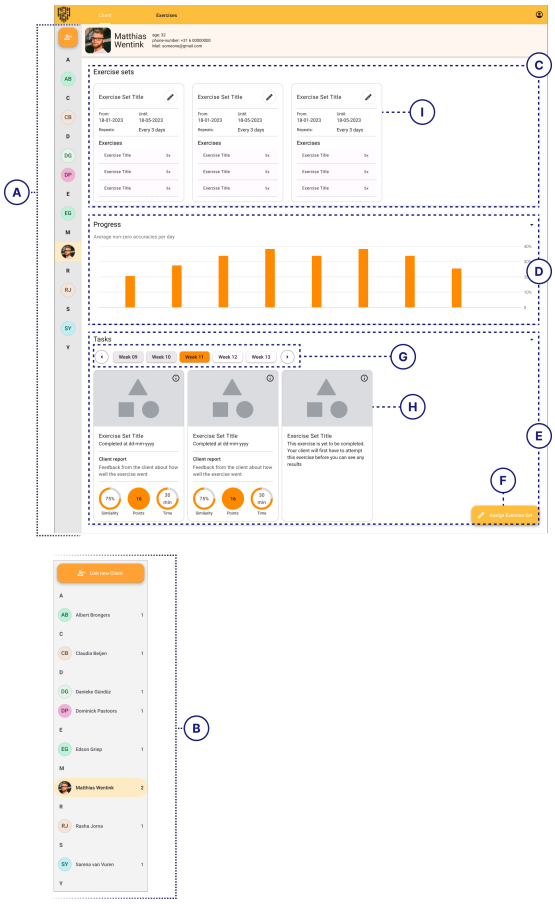
The login page is simple. It only features a fitting background picture, logo and login necessities. A login page could also serve the double function of also being a landing page. However, because awear already has quite an extensive website in place, it has been decided to make a simple login page. The login button that has been used is a single sign-on from Google.

7.3.2 Client overview

The client page offers a dashboard for all a supervisor's clients. A client can be selected through the navigation drawer **(B)** then is collapsed to a navigation rail **(A)** when not hovered in order to maintain space for the dashboard. The dashboard shows the assigned exercise sets **(C)**, a statistics section **(D)** and the tasks from the exercise set, based on the given interval, on a weekly basis **(E)**. All of these sections are collapsible containers, meaning that they can be minimised so that the user can focus on what's most important to them at that moment. On the bottom right of the page, there is a floating action button **(F)**. This button is meant to assign an exercise set to a client. As this is one of the most important actions of the supervisor side of the application, it is visualised as a floating action button. These buttons are displayed on a different layer on top of the main page, ensuring that they are always visible.

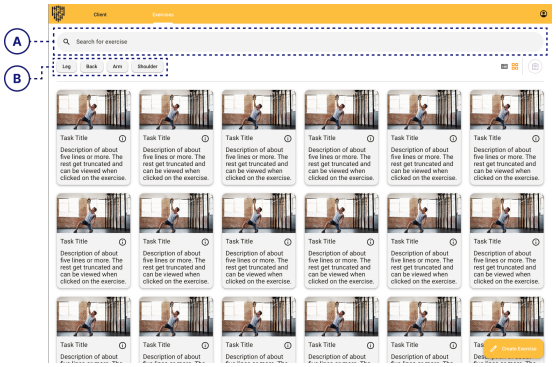
In the exercise sets section **(C)**, Different exercise sets cards **(I)** are shown that should provide the user with all the information they might need about an exercise set. It has some basic information about the set as well as a list of the exercises. Tasks **(H)** that come from these sets are shown in the Task field **(E)**. These have been separated so that the user can get an overarching view as well as an on-week-basis view of the tasks for the client. The tasks, can either be completed or not. When completed, some statistics and user feedback is shown. When this is not the case, it displays a simple message that the user is yet to complete the exercise.

In general, a supervisor will look at a timeframe of 2 weeks before the current week and two weeks after the current week. Because of this, the week selector **(G)** displays 5 weeks with the current week in the middle. This makes the view less cluttered for the supervisor. If the supervisor wants to look back or forward further, they can use the arrow dots to do so.



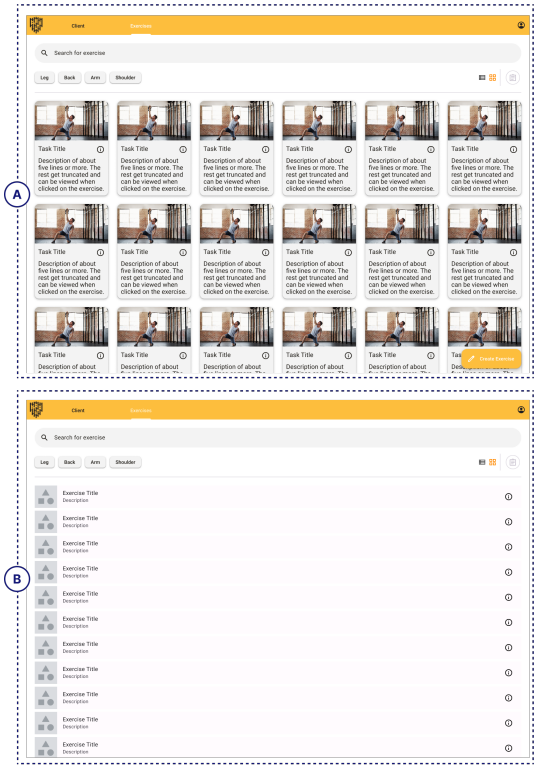
7.3.3 Exercise overview

The Exercise pages show all exercises. They can be filtered either by the search bar **(A)** or by filter chips **(B)** beneath the search bar. The chips have an 'OR' relation. Meaning that if the supervisor were to filter on both leg and back, all exercises that satisfy either criterion are displayed. This is contrary to an 'AND' relation that would only display exercises satisfying both criteria. The 'OR' relation is chosen because there is only one muscle group per assignment. As there are no exercises with multiple muscle groups, having an 'AND' relation would result in selecting no exercises.



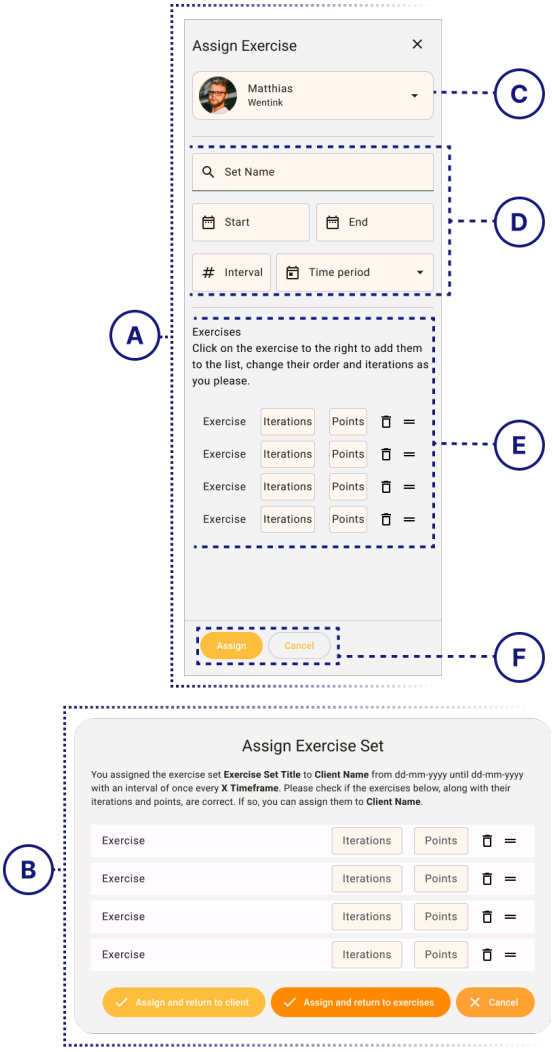
7.3.4 Exercise views

There are two views in which the exercises can be displayed. The first is tiled view **(A)**, where there are cards with an image and little description. Here, the exercises are ordered from left to right top to bottom. In general, tiled view or card view is said to be good for browsing information [21].
In the list view **(B)** everything is sorted from top to bottom. Both the image and text displayed on the main screen are smaller, which allows for more exercises to be visible on the screen. In general, list views are better at searching filtered information [21].
Both views have a purpose at which they excel. Yet, there is a time component to the views. Lists have been common in (web) applications from the beginning as they originated from bulleted lists from texts. In contrast, cards are newer, so older generations might prefer lists whereas younger generations might prefer cards. Since both have a use case and a demographic, our application has both options available. By default, the view is a tiled view because we think it gives a better overview and the provided images can give lots of information at a quick glance.



7.3.5 Assignment view in Exercise overview

Separate from the tile and list view, there is also the additional 'assign' view. In this view, a side panel **(A)** on the right of the exercises screen opens up. This works for both the tile and the list view. In this panel, the user is able to assign an exercise set to a client. The client is selected through a drop-down menu **(C)**. One client can be selected for assignment since, as a supervisor, you are often focused on one client at a time. Beneath the client selection, there is a form **(D)** that the supervisor can fill with the data of the exercise set. The 'set name' field is a standard text field. The start and end date are a date range, a component in material design that takes a start date and end date in an intuitive way. The interval data consists of two fields; The amount of time between instances as a text field and the time period type in a drop-down list, where the user can choose between hours, days and weeks. With these three choices and the amount of time, the interval has a lot of flexibility without being overwhelming for the user.
Further down is the list of exercises **(E)** that an exercise set consists of. The list is a draggable list, meaning that the order in which the exercises are assigned to the client can be changed easily. Furthermore, every exercise has two number fields one for the repetitions (i.e. how many times that specific exercise should be done) and one for the number of points. The points get awarded to the client when all exercise repetitions are completed.
When the supervisor clicks on 'Assign' in the button group below **(F)** a dialog window **(B)** shows up that asks the supervisor to confirm this assignment. It again shows the exercise. This is to make sure that supervisors do not enter the wrong information. If there are any suspicious data entries, these could also be shown here. For example, when the iteration is set to an absurdly high number.



7.3.6 Discrepancies between design and execution

The login, just like for the client side is a few buttons that allow the user to log in as multiple supervisors because there is no authentication system. In the client screen, when no client is selected there is a background design that is not in the actual implementation. This is because the positioning of awear's logo was tricky to do and seemed like a lot of work to figure out but a low priority in getting it right. There are no statistics, similar to the client side. The client profile images are missing, and the images for exercises are currently stock images.

8 Testing

8.1 Testing Plan

8.1.1 Approach

To make sure that we deliver a robust codebase, we want to ensure that the implementation has been significantly tested. We will do this in a couple of different ways to include multiple aspects and views of the system. With all of the testing, we are mainly focused on the flow through the system as this is the most important part of our project: modularity and thus the way that components interact. If we are able to verify that each of the individual module's are working alone and in connection with each other, it makes it easier to build on top of the base that is already there. Further on in the development process, it is assured that these individual parts are working. For the exact implementation, we would like to refer to the manual in appendix E. We will use the following five strategies to test our system:

1. **Unit testing**: One way to confirm the working and expected usage of certain smaller components and pieces is unit testing. Unit testing allows for the verification of very small so-called units. In the system, small bits are possible to test with unit tests. An example could be the getter and setter functions of a certain class. These unit tests form the backbone of each component. It ensures that each component individually behaves as intended. These unit tests are especially important for a system that is and will still be under development. It is one thing to know that it works right now, but perhaps more important to know that later on it is also still working.

2. **Endpoint testing**: When working with the system, you are interacting with the back-end server. In order to test all the specific interactions you are making with this server, you have something called an endpoint test. In endpoint testing, we send a request to the specific point of the server we would like to test. We can then test the data we get back from the server. Using this, we can make sure that interacting with the server goes as expected.

3. **Widget testing**: In Flutter there is this so-called widget testing. It is a form of testing that is very similar to unit testing, but then it is designed for Flutter and widgets specifically. It is possible to check the workings of a single Widget, multiple Widgets or the entire system. The basic structure of these tests is that a Widget is built and the Widget tree is initialised, then we are going to expect to find certain Widgets in that tree. For example, if I have a page that shows some text, I am going to expect this text element to be there when I load the page. This is perfect to test logic within a Widget (or unit). Whenever there are buttons that update the state of the Widget such that other Widgets will be present, it can all be tested using Widget testing.

4. **System testing**: Although this does not necessarily test the entire system, there is a package called 'Mockito' which has been designed to test streams or future information sources [2]. It is ideal for our use case to test the function of the API. It requires only a little modification to the existing system, but then the Mockito package will mock the API calls such that the API doesn't need to get called for each test. This allows us to efficiently and quickly test if the calls to the API work and if they are handled properly. Especially in such a system where a lot of the application relies on data to and from the database.

5. **Integration tests**: Integration tests are possible and available in Flutter. They allow for the whole system to be tested. It will be able to test the flow of the application. This can

Risk	Impact
The database connection does not work	H
The components are not modular	H
The database structure is missing links	H
Supervisor makes a mistake	M
Non material components are hard to implement in flutter	L
No exercise supervision	L

Table 1: Test risks

make sure that all your routing is working and that the widgets are cooperating properly. However, they require quite a big setup and therefore are out of the scope of this project.

6. **User testing**: User testing is mostly used to look at the system from the user's eyes, it will help us understand if the system is intuitive and easy-to-use, aspects we deem important because of the wide variety of computer skills the user is going to have. For this, the user scenarios as defined in D will be used. We focus mainly on the following points:

   - **Intuitive**: The system needs to be as intuitive as possible, requiring as little explanation. Especially on the client side, it should be as little effort as possible to use the application for its main purpose: doing the assigned exercises.
   - **Consistency**: Working together with intuitive design is a consistency of wording and icons, everywhere we use them they should be the same to never confuse the user.
   - **Aesthetic**: As a goal of the application is also to incentivise user to keep using the app, we want to make sure that the app has a simple and easy-on-the-eyes aesthetic. These qualities will help make the user experience as pleasant as possible and give no reason to not use the app.
   - **Help and documentation**: It may be necessary to still have some extra documentation, both keeping into account different ways of thinking as well as potential edge cases that are less intuitive, in this case, it should be easy to find an FAQ and help page with more extensive documentation.

8.1.2 Risks and contingencies

To be sure that no crucial features are implemented wrong, we have done a risk analysis of the project that can be found in appendix A. Table 1 shows the biggest risks in the project and their estimated impact. The risks with a high impact are also otherwise classified as critical risks. This means that these risks are located so much at the core of the application that they have the most disastrous impact. This is not to mean that even if any of these risks happen they can not be mitigated or fixed but depending on the impact of the risks it could mean that the service would not work in its entirety.

These risks will guide the functionalities to be tested and give us a grasp on the importance of functionalities as well as more insight into the hierarchy and completeness of our requirements.

8.1.3 Functionalities to be tested

Based on the testing methods and the risk analysis, we derived the most important features to be tested. The high-impact features are the ones that are the core of the project while medium or low

ones indicate that though this is a feature necessary for the final product, it is also a feature that can be expanded or redone in the future without the system needing to be redesigned in any way.

Functionality	Impact
The database structure	H
Component modularity	H
Full circle flow through the system	H
Add a new exercise	M
Browse exercises	M
Assign an exercise	M
Follow an exercise	M
Log in	M
Log out	M
Connect supervisor and client	M
Gather points from exercise completion	L
Show statistics	L

Table 2: Important functionalities

8.2 Testing Results

Below we show the results of each of the different kinds of tests and the potential changes we made with regard to these outcomes.

8.2.1 Unit tests

An example of a unit test that is used in this project is the following. There is a list that maintains the sensors on the client-side application. This list is a separate class with its own functions. We have written a unit test to verify that these functions are working as intended. When the list is made, it doesn't contain anything. When an element is added, it is expected that the list has grown by 1 and that the element is there. Although there is only an example unit test in the project, it shows its possibility and usefulness of it. The importance of this test does not necessarily show right now, but later on. Whenever the sensor list is extended with the proper sensors, tests like these will ensure that the list still functions like it is supposed to. Even though the code might be modified, this unit test can verify that everything is modified correctly. Another unit test that is used is the testing of JSON encoding and decoding. As all of the data traffic goes through the REST-API which sends it in a JSON format, both the supervisor and client-side have to decode this JSON into normal classes. This process has to work as expected and can be tested using unit tests.

8.2.2 System tests

To test the system we have used two different types of tests. To test how we handle database calls we use Mockito to simulate a database and control the output. This can help ensure that the data that is sent is handled appropriately by the application. We use widget testing to ensure that the widgets we display are working correctly. As Flutter uses a hierarchical structure based on widgets, it is simple to test these widgets in a different file. We provide the widget with some controlled

data and test whether the widget displays the data in the expected way. When the widget is expanded on at a later stage, this ensures that the expected outcome of this widget stays the same.

8.2.3 User tests

These tasks were done on both sides of the system. We did not have access to therapists and clients nor is the main purpose of the application related to previous knowledge needed as a therapist. As such, we asked several of our peers and other acquaintances to help us with the user tests. A small convenience sample was acquired consisting of 6 people out of which 5 people had been to physical therapy before.
We started them with either the supervisor or the client side and asked them to perform a series of tasks listed below. They were based on the user scenario D but changed slightly to the capabilities of implementation of our platform at that time. Before the participants completed the exercises, they were explained what our platform entails and they signed an informed consent form, that can be found in appendix F. They were asked to think out loud when performing the tasks so that we could know what they were thinking about when using the system and thus evaluate the pros and cons of our implementation. Furthermore, they were timed as an extra indication of how easy tasks were to perform.

Client side tasks

- Log in
- Start assigned exercise
- Complete exercise
- Find settings

Supervisor side tasks

- Log in
- View all exercises
- Filter on shoulder exercises
- Add an exercise to the system
- Assign an exercise to a client
- Look at client data

By doing these tests, our insight into how our platform performed on both functionality and user experience grew. The participants did some actions that we, in our own walk through of the system, did not think of. Furthermore, they commented on what parts they felt were nice and which parts left them perhaps a bit more confused. The insights gained will be discussed per application and then per task starting with the Login for the Client side. This interaction felt logical to the participants, as there is a familiar 'login with Google' button. However, some participants were slightly confused by the drop-down menu under this button. This was to be expected, as this menu is a placeholder since no actual authentication is built in. Starting the exercises felt logical to the user and nobody had any trouble with doing so. When performing the exercises there were some issues. The first is with the sensor placement. When switching back and

forth between the sensor placement screens, some sensors could not be selected which is a technical issue. Further along, when participants got to performing the exercises, it was not always clear to them what they should be doing which is an issue of user experience. Then when they got to the last task of going to the settings, there were no issues.

Similar to the 'Login' task for the client, it went very smoothly for the supervisor side. The same goes for viewing the exercises. Filtering on shoulder exercises went smoothly for most participants although one got lost in the assign view instead. Adding an exercise went smoothly, however, the added exercises were not always immediately visible in the system afterwards which indicates a technical issue. Assigning an exercise to a client could be done in multiple ways, which was noted by two participants which they appreciated. Half of the participants did it in the client screen and the other half in the exercise screen. Some trouble with this was that people were not familiar with the date range input field and that people were not aware of what all the input fields mean. Looking at clients was easy for most participants but getting their data was not. This was partially because the progress container was left empty and partially because it was not clear that the user could scroll through the dashboard.

9 Further usage

Further usage of the platform should be possible easily because of the modularity and documentation either in this report or in the user manual that can be found in appendix E. In this section, the four most important components of the platform that require additional development are highlighted and it is shown how they can be improved to acquire a more full-fledged system in the future.

9.1 Movement sensor incorporation

Prior to this project, an implementation was made that couples Movella dot sensors with a flutter and plotted their value on a graph. The plan of our project was to use this code and expand on it further with the rest of the platform. However, when trying to do so, there were some complications. At first, it was hard to get the code working. Mainly because it had to run on an Android phone and the permissions were not always working. In the end, it was working from time to time but it was not consistent and hard to reproduce. Because the original sensor component not properly working, data was substituted with microphone sensor data. This way the whole flow is simulated whilst still having some sensor data to be able to give a good impression of how the system would deal with the sensor data from the Movella sensors.
The sensor component has to be fixed or re-made for a final system. It would also be beneficial if multiple different sensors could be connected.

9.2 Movement analysis and recommendation

Solely getting movement data from exercises is beneficial and can certainly be used in the treatment of patients. However, it would be even better if the movements are analysed prior to the supervisor viewing them. You can think of having AI analyse it and highlight faulty movements. There is an ongoing project for awear that uses Movella sensors and AI to detect whether the client is sitting, lying down or walking. If good results come from that, perhaps it can be extended to other movements that belong to exercises in the system. There is also already some research that shows potential for detecting exercise movement and giving accurate recommendations according to the exercise [13].

9.3 Authentication

As of now, there is no authentication in the system. Both the supervisor and client application login without using a password but just selecting the supervisor or client they would like to use. Adding authentication makes sure that everyone only has one account which they can use, and it make sure that supervisors stay supervisors and clients stay clients. Lastly, the security it adds also is beneficial. Right now, everybody could use the system as everyone, which is not a situation you would want.

9.4 Statistics

There are some statistics within the current implementation. However, all of these have to do with the points gained from an exercise. These points are a relatively arbitrary value to have statistics on, as they are set by the supervisor. These points are meant as an incentive for the client but they do not represent values intertwined with the execution of the exercise such as accuracy and time. The reason that these are the only statistics present is again the connection with the sensors,

but once this component is functional, the data can be used for more meaningful statistics, something that has to be researched again at that point to see what would be the best way of analysing and visualising these statistics.

9.5 Future Vision

After the above-named four features have been added to our modular base, the platform would be fully ready to use. All key features would be present in the application. After this implementation, it would also still be possible to extend the application with elevated versions of features that are already present. Examples of these are:

- Elaborate gamification
- Integration with appointment planning software
- Machine learning applications for recommendations and analysis

More features can of course be added as they later come up as useful. The base is built in a way that the new feature can be added plug-and-play.

10 Evaluation

This section will describe how the project went. Most of the content in this section was acquired from an evaluation meeting we had as a team where we discussed what went well and what could have been better. In this meeting, the planning, communication and team organisation, responsibilities, and final result were discussed which are summarised in the following sections.

10.1 Planning

A general planning was made in the first week of the project, a more detailed planning was finalised in week 3/4 when we had more insight into how the implementation plan looked so our planning would be as accurate as possible. Furthermore, every two weeks we had a meeting with the client and every week we had a meeting with our university supervisor. These meetings were used to get on one line about the direction of the project and updates on the progress. Please refer to Chapter 4 for these meetings.
Even with the more detailed planning that was made in weeks 3/4 of the project, we as a team fell a little bit behind. We think this is due to multiple factors namely, the fact that all programming languages used were new to us and that we lost quite a bit of time doing repetitive tasks. For example, a first diagram of the database was made in week 1 but as we started development, the necessities for our database increased causing a change in the database diagram. This happened quite often which caused for some delay. Another example is that some of the front end was working already, but without a connection to the back-end. Because of this, front-end developers had to mock some data to further develop the application which later had to be completely rewritten. Perhaps this negative effect could have been minimised if the whole group was focused more on design in the first weeks of the project, as this was now mostly done by two people. On the other hand, we decided to schedule the work of the individual group members according to their strengths and this also proved to be viable as is further elaborated in section 10.3.

10.2 Communication and team organisation

Part of the reason for our delay in planning was due to our communication and organisation structure. During the project, we had a git available that we intended on using. However, most of the group members were used to a visual GIT system like GitHub or Gitlab which resulted in a lack of use in the GIT that we had. The good thing about Gitlab is that requirements can also be put in there and that they can be managed using a progress board. The way that we did it now, is by having multiple applications and systems for tracking progress without having a nice structure. This made it so that not all communication was as effective as it could have been. Another way we could improve our communication is by commenting on issues or in certain threads. For example, by using an application like slack.
In the beginning, the meetings we had as a group were somewhat irregular. This resulted in people working on their own thing without communicating to the others as much. Later on in the progress this started to improve by having stand up meetings at fixed times and locations at least four times in the week. We noticed that our effectiveness was increased significantly after we did this.

10.3 Responsibilities

Before we chose our project, we had a discussion about the different interests and expertise within the team. Our project was chosen accordingly to what we thought coincided with our strengths and our preferences. The division in interest and expertise was continued during the project with

the division of tasks. While everyone helped with almost every task, the division made sure that someone was responsible for the bigger picture of that component. The assigned responsibilities are listed below:

- **Matthias:** Communication manager with the client, Front-end designer, and poster and slide design.
- **Niek:** Front-end designer and developer client side
- **Pepijn:** Front-end developer supervisor side
- **Rosan:** Requirements specification and analysis, Risk analysis, Ethics analysis, and head of presentations
- **Rutger:** Back-end developer and Head database design

We are satisfied with the responsibilities that we set up as each person was fit for their job. Matthias made a lot of designs and could also quickly change designs whenever needed. Niek helped with some of these designs for the client side and also kept in mind the bigger picture of the project. Rosan set up the requirements in a very convenient way and she also kept in touch with the development so she could take the lead during most of the presentations that were held. Rutger taught himself a new programming language and put together a great back-end with it that will still be viable in the future.

10.4 Results

We are satisfied with the platform that we developed. We would have liked to implement all requirements and some requirements that were implemented, we would have liked to improve. However, we do also recognise that ten weeks is little time for the development of a full platform and that in retrospect, we did a decent job. Furthermore, we think that we provided a good stepping stone that awear can build upon further especially in combination with the manual and report that we have written.

10.5 Conclusions

There were some delays in our planning caused by a lack of organisational structure within our team and insufficient communication causing not all requirements to be implemented to their fullest. However, the goal set out with our supervisor at the start of the project was to provide a good stepping stone for awear to build on. We think that we did achieve this and that this is in part due to the way we set up the responsibilities within our team as well as the fact that we really banded together the last couple of weeks to make the most out of it.

References

[1] Duolingo. URL https://www.duolingo.com/.

[2] Mockito package. URL https://pub.dev/packages/mockito.

[3] Movella dot sensor. URL https://www.movella.com/products/wearables/movella-dot.

[4] Physitrack. URL https://www.physitrack.nl/.

[5] Awear., Jul 2022. URL https://awear.tech/.

[6] J. Bentham et al. *Panopticon*, volume 162. London: Verso, 1995.

[7] I. Cofone. Beyond data ownership. *Cardozo L. Rev.*, 43:501, 2021.

[8] A. F. Cooper, E. Moss, B. Laufer, and H. Nissenbaum. Accountability in an algorithmic society: relationality, responsibility, and robustness in machine learning. In *2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 864–876, 2022.

[9] M. Dural and C. Kohls. The design of fitness apps. In *Proceedings of the 24th Conference on Pattern Languages of Programs*, pages 1–12, 2017.

[10] B. Fazzini, T. Märkl, C. Costas, M. Blobner, S. J. Schaller, J. Prowle, Z. Puthucheary, and H. Wackerhage. The rate and assessment of muscle wasting during critical illness: a systematic review and meta-analysis. *Critical Care*, 27(1), Jan. 2023. doi: 10.1186/s13054-022-04253-0. URL https://doi.org/10.1186/s13054-022-04253-0.

[11] A. M. Froomkin. Big data: Destroyer of informed consent. *Yale JL & Tech.*, 21:27, 2019.

[12] K. L. Galbraith. Terms and conditions may apply (but have little to do with ethics). *Am. J. Bioethics*, 17:21, 2017.

[13] A. Hannan, M. Z. Shafiq, F. Hussain, and I. M. Pires. A portable smart fitness suite for real-time exercise monitoring and posture correction. *Sensors*, 21(19):6692, 2021.

[14] J. P. Higgins. Smartphone applications for patients' health and fitness. *The American Journal of Medicine*, 129(1):11–19, 2016. ISSN 0002-9343. doi: https://doi.org/10.1016/j.amjmed.2015.05.038. URL https://www.sciencedirect.com/science/article/pii/S0002934315005379.

[15] H. M. Hondori and M. Khademi. A review on technical and clinical impact of microsoft kinect on physical therapy and rehabilitation. *Journal of Medical Engineering*, 2014:1–16, Dec. 2014. doi: 10.1155/2014/846514. URL https://doi.org/10.1155/2014/846514.

[16] S. Klenk, D. Reifegerste, and R. Renatus. Gender differences in gratifications from fitness app use and implications for health interventions. *Mobile Media & Communication*, 5(2):178–193, 2017.

[17] C. E. Lang, J. Barth, C. L. Holleran, J. D. Konrad, and M. D. Bland. Implementation of wearable sensing technology for movement: Pushing forward into the routine physical rehabilitation care field. *Sensors*, 20(20):5744, Oct. 2020. doi: 10.3390/s20205744. URL https://doi.org/10.3390/s20205744.

[18] E. Luger, S. Moran, and T. Rodden. Consent for all: Revealing the hidden complexity of terms and conditions. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, page 2687–2696, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450318990. doi: 10.1145/2470654.2481371. URL https://doi.org/10.1145/2470654.2481371.

[19] Q. Ma. The effectiveness of requirements prioritization techniques for a medium to large number of requirements: a systematic literature review. 01 2009.

[20] M. Mannion and B. Keepence. Smart requirements. *ACM SIGSOFT Software Engineering Notes*, 20, 03 2004. doi: 10.1145/224155.224157.

[21] M. Schenker. Cards vs. lists, how the impact ux. URL https://www.webdesignerdepot.com/2017/01/cards-vs-lists-how-they-impact-ux.

[22] J. Singh, I. Walden, J. Crowcroft, and J. Bacon. Responsibility & machine learning: Part of a process. *Available at SSRN 2860048*, 2016.

[23] C. R. Sunstein. Nudging: a very short guide. *Journal of Consumer Policy*, 37:583–588, 2014.

[24] E. Tjon Tjin Tai. Data ownership and consumer protection. *J. Eur. Consumer & Mkt. L.*, 7: 136, 2018.

[25] P. M. Vanderburgh. Occupational relevance and body mass bias in military physical fitness tests. *Medicine and science in sports and exercise*, 40(8), 2008.

[26] A. Vijverberg and R. Opdenakker. Vitale organisatiestrategie. *Een ontdekkingstocct naar waarde, positie en identiteit*, 2013.

[27] D. Woods and S. Dekker. Anticipating the effects of technological change: A new era of dynamics for human factors. *Theoretical Issues in Ergonomics Science*, 1(3):272–282, 2000. doi: 10.1080/14639220110037452. URL https://doi.org/10.1080/14639220110037452.

[28] W. Yue, Z. Wang, J. Zhang, and X. Liu. An overview of recommendation techniques and their applications in healthcare. *IEEE/CAA Journal of Automatica Sinica*, 8(4):701–717, 2021.

A Risk Analysis

It is important to take the goal of the project into account with this risk analysis. The goal is to create a modular system that Awear can improve upon. Some components will not be implemented fully and instead, a placeholder will be made to show the component's basic functionality and place in the system. Because of the fact that some components do not need to be implemented fully, the consequence of the risks associated with these components is relatively low. Risks that pose a threat to the connection between components and thus the modularity of the system do have a high consequence.
Below we set out possible risks divided into critical and non-critical components. For every risk we give some explanation, rate the chance and impact of the risk and explain what kind of consequences it could have on our project as well as a possible later final version of the system.

Table 3: Risk analysis

What	Explanation	Chance	Impact	Consequence/Fix
Critical				
The database connection does not work	The Rest of the API or any of the connections between the application and the database do not work.	Low	High	This is not impossible to fix and it is one of our top priorities. Therefore, we do not see this happening. The front-end and back-end can also already sufficiently be built before the connection is established
The components are not modular	Some of the components are not modular	Medium	High	This will not break the implementation but it will make it harder for Awear to use the current implementation as a stepping stone for the rest of their ventures. A way to minimize this damage, the project should be well documented so that a lack of modularity can be fixed.
The database structure is missing links	Some links necessary to fulfil features are not present in the database	Low/ Medium	High	However thought out the structure may be, it is of course possible that something is missed. This is very unlikely but if such a miss happened it can be found through testing and can then be fixed.
Non-Critical				
Movement data is hard to work with	Movement is the centre of our application and we should have it working at least a bit	Medium	Medium	The system is only an architecture for exercises but we can't actually do anything with the exercises at all. A placeholder can still be used to show the flow of data through the system.
Non-material components are hard to implement in Flutter	A majority of the components used are from material design. However, custom-designed components might be harder to implement in Flutter.	Medium	Low	Less productivity / longer implementation time
No exercise supervision	Supervisors can make exercises that include unhealthy movement that can harm patients	High	Low	The responsibility of having good exercises is with the supervisor. If the system is not capable of providing extra supervision, it is up to the supervisor to do so.
Supervisor makes a mistake	Supervisor assigns the incorrect exercise to a patient or the correct exercise to the wrong patient.	Medium	Medium	This is not great for the patient but a feedback system can be in place that makes the impact of this problem relatively low
Data overload from sensors	The application receives too much data from the sensors to process	Medium	Medium	There should be a buffer in place, if this buffer were to overload too easily, it might be possible to replace it with a better one.

B Extended requirements

Table 4: Extended requirements

ID	Description	Criterium	Performance	Stage	Priority	Source
1	As a supervisor I want to see all possible exercises					
1.1	The system must have a mechanism which enables supervisors to log in and out of the application.	Time it takes to successfully login or out	Maximum 30 seconds	Initial Design	Must	Developer
1.2	The system must have a mechanism to display all exercises to supervisors	Time it takes to successfully display a page of exercises	Maximum 20 seconds	Initial Design	Must	Product owner
2	As a supervisor I want to add new exercises to the system					
2.1	The system must have a mechanism which enables supervisors to add exercises to the system.	Usability + Time it takes to upload the exercise to the database	Maximum 10 minutes of filling in details + Maximum 30 seconds upload time	Initial Design	Must	Developer
2.2	The system must have input sanitation.	SQL injection must not be possible	Pass or fail	Initial Design	Must	Product owner
3	As a supervisor I want to assign exercises to a client					
3.1	The system must have a mechanism which enables supervisors to assign exercises to a client.	Usability, time to assign an exercise to a client	Maximum 1 minute	Initial Design	Must	Product owner
4	As a supervisor I want to see the progress of a client					
4.1	The system must have a mechanism to visualise the client's data to see the progress.	Time to load a visualisation	Maximum 20 seconds	Initial Design	Must	Developer
4.2	The system must have a mechanism to save data from the performed exercises for analysis	Time it takes to save data performed	Maximum 1 minute	Initial Design	Must	Developer
4.3	The system could have a mechanism that compiles a weekly review for the supervisor.	Time to compile the weekly review	Maximum 10 minutes	Initial Design	Could	Developer

Continuation of Table 4

ID	Description	Criterium	Performance	Stage	Priority	Source
4.4	The system could have a mechanism that emails the weekly review to both the client and their supervisor	Succesfullness to send a weekly review on Monday	Pass or fail	Initial Design	Could	Developer
5	As a supervisor I want to see an overview of all clients					
5.1	The system must have a mechanism which enables clients to specify what supervisor they belong to.	Time to link a client to a supervisor	Maximum 1 minute	Initial Design	Must	Developer
6	As a client I want to follow assigned exercises					
6.1	The system must have a mechanism which enables clients to log in and out of the application.	Time it takes to successfully login or out	Maximum 30 seconds	Initial Design	Must	Developer
6.2	The system must have a mechanism that allows data to be able to be qualified as the correct exercise or not with the help of example data.	Time to qualify incoming data as correct or not	Maximum 1 minute	Initial Design	Must	Product owner
6.3	The system should have a mechanism to evaluate how the exercise went for the client.	Time to fill out an evaluation	Maximum 1 minute	Initial Design	Should	Developer
6.4	The system won't have a mechanism to show all exercises to the client.	A client cannot see other exercises than those assigned to him	Pass or fail	Initial Design	Won't	Developer
6.5	The process of connecting and calibrating the sensors should be as smooth as possible.	Time to connect the sensors	Maximum 2 minutes	Initial Design	Should	User
7	As a client I want to see additional information about my exercises (how many times, how often etc).					
3.1	The system must have a mechanism which enables supervisors to assign exercises to a client.	Usability, time to assign an exercise to a client	Maximum 1 minute	Initial Design	Must	Product owner
6.1	The system must have a mechanism which enables clients to log in and out of the application.	Time it takes to successfully login or out	Maximum 30 seconds	Initial Design	Must	Developer
8	As a client I want to follow a clear explanation of how the exercise is done					

Continuation of Table 4

ID	Description	Criterium	Performance	Stage	Priority	Source
8.1	The system should show an explanation of the exercise beforehand	Percentage exercises accompanied with an explanation	Minimum 98 per cent	Initial Design	Should	User
9	As a client I want visual aid for the exercise explanation.					
9.1	The exercises must have visual aid as part of the explanation.	Percentage exercises accompanied by visual aid	Minimum 98 per cent	Initial Design	Must	User
6.3	The system should have a mechanism to evaluate how the exercise went for the client.	Time to fill out an evaluation	Maximum 1 minute	Initial Design	Should	Developer
10	As a client I want to have a page where I can see statistics on my progress.					
10.1	The system could have a mechanism that compiles a weekly review for the client.	Time to compile the weekly review	Maximum 10 minutes	Initial Design	Could	Developer
4.1	The system must have a mechanism to visualise the client's data to see the progress.	Time to load a visualisation	Maximum 20 seconds	Initial Design	Must	Developer
4.2	The system must have a mechanism to save data from the performed exercises for analysis	Time it takes to save data performed	Maximum 1 minute	Initial Design	Must	Developer
4.4	The system could have a mechanism that emails the weekly review to both the client and their supervisor	Succesfullness to send a weekly review on Monday	Pass or fail	Initial Design	Could	Developer
11	As a client I want to mark an exercise if something is unclear and provide feedback					
11.1	The system must have a mechanism to add feedback about the clearness of an exercise.	Time to mark an exercise and add clear feedback	Maximum 2 minutes	Initial Design	Must	Developer
12	As a client I want to get some type of reward for following the exercises (points, confetti)					
12.1	The client-side app must be easy and enticing to use	Usability, Time to find a specific feature	Maximum 1 minute	Initial Design	Must	User

Continuation of Table 4

ID	Description	Criterium	Performance	Stage	Priority	Source
12.2	The system should allow for a point system	Percentage of times where the system successfully assigns points to a person when an exercise or set is completed	Minimum 99 per cent	Initial Design	Should	Product owner
13	As a client I want to get a notification for doing my exercises.					
13.1	The system should be able to give a notification to the client when their exercises are ready to be performed.	Percentage of the times that the client successfully receives a notification	Minimum 99 per cent	Initial Design	Should	Developer
14	As a client I want to get a notification if new exercises get assigned.					
14.1	The system should be able to give a notification to the client when new exercises are assigned to them.	Percentage of the times that the client successfully receives a notification	Minimum 99 per cent	Initial Design	Should	Developer
15	As a supervisor I want to be able to get a notification of marked exercise with feedback to improve it					
15.1	The system should be able to give a notification to the supervisor when an exercise they added is marked as unclear.	Percentage of the times that the supervisor successfully receives a notification	Minimum 99 per cent	Initial Design	Should	Developer

End of Table

C Ethical Analysis

Introduction

Our project consists of building an app to assist with at-home physical therapy exercises. For your imagination you can picture a fitness app but then specifically used in situations where you work with some sort of supervisor. The users of this application will thus consist of movement professionals (therapists or trainers for example) and their clients. The goal of this application is twofold: to help guide people through physical exercises that got assigned to them by a therapist or trainer and to help the supervisor with maintaining an overview of his clients exercises. The app will guide the client through the assigned exercises, take data about them, allow for feedback and incentivise the client to get into the habit of regularly doing these exercises [14]. The data collection is done with Movella DOT sensors[3] that can be put on the body and record movement data of the exercises. This data can be used for analysis as well as for real-time recommendations to the user.
On the other side of the app, the supervisor can assign these exercises to their clients (as they would normally verbally do) and keep track of their clients through the data that the exercises generate. The final goal is of course to support the clients in their process as much as possible without overworking the supervisors.
An important thing to note is that though the application will incentivise the client to do their exercises with forms of gamification lke gaining points, it is not the case that the client is able to do more exercises than assigned through the app. Once the set of exercises for a day is done, there is no way to keep doing more exercises to score more points. This is a distinction with a normal fitness apps that we believe to be important. The app is here to support recovery and overexercising would only damage the recovery [5].

Our part in this final vision consists of the modular base of this application. Making sure that the application is implemented in a modular fashion and extra features can easily be added later without having to redo our work. So at the end of our project, we will not have implemented all features, we will not yet give proper recommendations to the client or do any kind of machine learning on the data, we will not have elaborate gamification or the best log-in system. But our application will allow to seamlessly switch the placeholder we placed there for proper features and has a solid data structure to support all of these features in the future.
In this reflection, we will look at the possible implications of the finished product vision beyond our projects modular base.

Ethical Issues

We believe the most obvious ethical question here is the data. As with any application that works with data and with machine learning, some questions arise. In our case the most important questions are: Who has data ownership?, How do we ensure bias does not mess with the decision making and What happens if the decision-making turns out to be malformed?
The goal of our application is purely to guide clients through these exercises and allow the supervisors insight into these at-home exercises. This means that, at least for now, we do not aim to connect this system to the client management system that the supervisors likely already use. This also means that we can keep the amount of personal data saved to a minimum, effectively minimising a potential privacy breach [24]. This personal data is inherently owned by the person it belongs to. The more difficult data ownership question lies with the collected movement data during the exercises. This data is collected by the application's company to assist the users but has

the potential to be used for other purposes [7]. With the data collected from these sensors and how it is saved to keep a proper overview you could deduct things like someone's pattern, knowing at what point in the day they usually do their exercises. How consistent they are with their exercises. Potentially we could even see what kind of physical state they are in due to their movement data (for example, how low or fast they squat could say something about their leg strength).
Furthermore, if we are set on saving as little personal data as possible, what is the potential for bias? Often, some form of knowledge about the body is necessary for knowing how people do exercises, after all, your body functions differently if you are male or female, elder or a child [25]. Not only might this potentially be visible in the data (data privacy again) but it is likely that the decision-making based on the data should be different for different people, or at least, the decision-making process should have some knowledge of the person to make sure it gives the right recommendations [14].
Which brings us to our last concern about the data: What if, due to the options raised above, the decision-making becomes malformed? Not knowing some personal information (like height or gender), or a bias in the data (for example, a lot of elder people) might skew the decision-making. With missing such information comes the risk of not knowing if it affects decision-making. The recommendations could accidentally become over-tailored to a certain group, or even unrepresentative of any group [16].

The next concern starts with a similar premise: What if we give wrong recommendations due to bias or lack of data? But rather than acknowledging that this is inconvenient, we believe that this could potentially be harmful [9][28]. As our application deals with the physical recovery of people, it is very important not to steer them in the wrong directions. We already do this by not allowing more exercises to be done than assigned but giving wrong recommendations during the exercise could be as much, or even more harmful to the client's recovery. This brings us to the important question: Who is responsible if we give a wrong recommendation? And who is responsible to make sure that people know how their data will be used? And by extension: if we use machine learning, can we really ensure informed consent?
We all usually just click accept on the terms and conditions, often without reading them or trying to read them but quickly giving up because of all the legal lingo that most people simply don't understand [12]. Yet it does make us agree to something and thus makes us legally liable [18]. However, does this also make the customer responsible? Often now this becomes the case, with arguments like "just don't use it", but what about the responsibility and accountability of the product owner or the developer? Especially with something like wrong recommendations, even if told it does not work perfectly, it is unreasonable to expect the user to know when it is correct and when it is wrong. Thus it would be hard to place all of the responsibility on the user. Within a reasonable margin, the application should inform the user what to expect and what their data is used for. In the end, at the very least, the makers should be known and accountable [8].
Another issue in machine learning applications is that sometimes, not even the makers themselves know exactly what is happening to the data. They could lose the control that is necessary to accurately inform others about what happens with the data [11]. The informed consent could include that the data is only used inside the application for recommendations and statistics, giving some form of knowledge about the use of the data, but likely after that it becomes somewhat of a black box as to what happens with the data [22]. Making the term "informed consent" true as to what the data will be used for but how exactly it will be used for this goal would still remain a black box.

And the last area of concern might be the vaguer or less obvious one, The change in the

relationship between the supervisor and the client. Often, something like this is hard to see but all around us, relationships are changing due to evolving and emerging technology [27]. Whether it is about the pace of correspondence or the constant presence of technology to think about. Something as simple as a hearing aid changes the relationship between the user and the people they interact with. Most of the time you don't even realise this change until it's over and you look back, or when speaking to someone who lived through a change you did not experience or the other way around. The way that kids cannot imagine a life without phones or the way our younger siblings got a phone way earlier than we did. The advancements in technology are not only changing our surroundings and our reach but consequently, also the way we interact with each other. This application would in its own regard, change the way that the client and supervisor interact. Before, the supervisor had no insight except for the client's word and maybe noticeable progress at the next appointment. Now the supervisor can monitor the client from afar. This might feel more pressuring for the client because not doing your exercises for a day would be noticed by the supervisor now whereas before it would not have necessarily been. This is something we see often when a change is made to technology. It allows for extra monitoring but also for increased pressure on the monitored side as they now feel like they are watched and cannot slip up [6]. This changes the dynamic not only for the client but also for the supervisor. Different supervisors might deal with this new information differently. It's a question of whether or not we want this change and if we believe the change to be worth it in comparison to the added benefits.

Possible solutions

As we concluded, there are things that can potentially go wrong. We already talked about the goal to keep the personal information stored to a minimum and the fact that we do not connect the application to any sort of other management systems. This eliminates any information flows to and from we cannot control or oversee. As said above, not having some personal information might skew the decisions based on a biased dataset. It should be tested to see where the best point is between eliminating bias as much as possible and saving as little personal information as possible. It is likely that this will not eliminate all bias possible but the goal is for the application to function adequately for everybody. In this situation, it is likely impossible to have the best of both worlds so it will be a compromise between function and personal data.

After we find the best compromise between function and personal data, it is still unsure how accurate these recommendations will be exactly [13]. When thinking about this issue we already decided that in the case of giving recommendations, it is always better to not give a recommendation or give it too softly than to make a wrong recommendation, and thus the implementation of this feature would need to be as careful as possible without ruining recommendations altogether. We also decide that we likely won't stop people from taking a dubious action but rather warn/recommend them not to [23]. For example, if someone does their exercises at 23:30 at night and the ones from the next day at 00:30. This is obviously not the right way but rather than disable this, we thought it would be best to make a pop-up asking if they are sure because the last workout was so short ago. This way it is only a nudge and give transparency in the process.
These kinds of "ignorable" warnings also try to make sure that at every step of the application, the user knows what they are getting into, being transparent[12] of potential flaws in the system. This makes sure that next to any terms and conditions signed, we give the user a reasonable expectation of the workings of the application. This is the core value of informed consent: the reasonable expectation of what will happen to your data. Even if the algorithm feels like a black box, for data

use this is not so much an issue when the use and intentions are clear.
This transparency that nudges and warnings provide together with informed consent directly correlates with the responsibility. Responsibility is always unsure and likely only to be judged in a case-to-case fashion. Defining the responsibility beforehand is usually not possible. The important part is that the makers remain visible and accountable so that potential breaks can be solved with everyone that bears some form of responsibility [8].

For as far as the relationship change goes, the important thing is always awareness. Awareness of how this is different and potential positive and negative impacts. The change has its pros and cons and every client-supervisor pair will value these differently. Change in relationships is unavoidable with the world around us constantly changing, but the way changes are made can be influenced by design choices.

All in all, As with any data-driven application, transparency and awareness are of incredible importance to make sure the system is designed in line with ethical system standards, Problems and solutions will need to be thought of as part of the process and options will need to be weighted, always keeping issues like privacy and interaction change in mind. The technology can be of added value to our lives only if the development can be done properly.

D User Scenario

Client scenario

There you are, suddenly at physical therapy because you moved wrong. Your therapist gives you some exercises to do at home and strengthen your muscles. He recommends you use the AWear app so he can see how you do and you get guidance.
At the end of your appointment, you install the app. You log in [**.Lets the client log in**]. The supervisor gives you his personal code so you can link to him in the app [**.Link client and supervisor**] and he can see your progress.
The next day at home you get a notification that your exercises are ready [**.Give a notification when clients' exercises are ready**] and start up the app on your phone. You see the exercises you talked about with your supervisor yesterday [**.Let the supervisor assign exercises to the client**] and click on start. The app starts to explain what you'll be doing and guides you through placing the sensors on your body. While doing the exercises the screen shows a game and you try to score as many points as possible [**.Have visual aid to an exercise/ Gamify client progress**]. After each exercise, you'll rate how you felt doing the exercise [**.Evaluate how the exercise went**]. One of the exercises is somewhat unclear and you leave feedback on the clarity of the exercise [**.Allow feedback on the clarity of an exercise**]. After doing some exercises, your app tells you that you gained an achievement with the points gathered in your exercises [**.Gamify client progress**]. After you are done for the day and every progress bar shows completion. You move to the progress tab to check on your stats.

Supervisor scenario

Day in, day out you send clients home with exercises and no idea if they do them. Until you stumble upon the AWear app. A way for you to see your clients progress at home and gain more insight into your client's needs.
You sign up for the application [**.Lets the supervisor log in**]. In your profile you add the institution you're a part of. After this, you scroll through the list of exercises to see what you are working with [**.Shows all exercises to a supervisor**]. You notice that a particular exercise you often hand out is missing. Luckily you see that it is possible to add an exercise to the list [**.Lets the supervisor add exercises**]. You click on add and fill out the form of information about the exercise, you put on the sensors yourself and do the exercise a couple of times to add accurate data for the system to work with [**.Include base data to an exercise as supervisor**].
The next time a client comes to you, you ask them to download the app and link to you as supervisor [**.Link client and supervisor**] so you can assign them exercises to follow [**.Let the supervisor assign exercises to the client**]. One of the exercises you assign to this client is the one you have added before. The next day you get a notification that there were some unclarities with the exercise [**.Get a notification when the supervisor exercise gets feedback**]. You click on the notification to see what the feedback is and change the exercise accordingly. While you are in the system anyway, you click on this client to see the progress they have made [**.Visualise client data**].

E User Manual

E.1 Introduction

This manual gives an overview of how to get the system running, how do you use it, how to build on top of it and some design choices

E.2 Build Instructions

This section describes the build instructions in a detailed and specific manner for this project. As the front end is built using Flutter, the setup and build instructions will be quite similar to any other Flutter projects, and therefore require the same preparation. Any uncertainties regarding the setup can be found online quite easily.

E.2.1 Front-End supervisor- & client-side

In order to run the front-end applications, main.dart has to be run in web mode for the supervisor-side. For the client-side, an emulator needs to be started or a phone needs to be connected to the laptop. Running either one of these applications can be done in android studio or in visual studio for example. Before being able to run the applications, the dependencies need to be installed. These dependencies are in pubspec.yaml and can be installed by running the command 'pub get'.

E.2.2 Back-End

Building the back-end application requires having **Docker** and **Go** installed. After having the repository on your system, use 'go get .'. To install all the dependencies of the **Go** project, you have two options.
1) You can test the run on the server. In the repository found you have a docker container and a main.go file in the root of the 'backend' folder. Before running the database you will need to set the variable environments, both in the '.env' file located at the base of the project and the environment variables in the 'docker-compose.yml'. After setting the environment variables, you have to run the **PostgreSQL** database in the docker container. To do this, go to the '/docker' folder and run 'docker-compose up' (of course make sure that the docker service is running before using docker) if you would like to see what happens and with '-d' if you would like to run it as a background process. If you would like to rebuild the database, use the '–build' flag to rebuild the container. After having started the database, you can use the command 'go run main.go' in order to run the go server in debug mode. If you would like to run the back-end in "release mode", you can either achieve this by adding the following line to the code: 'gin.SetMode(gin.ReleaseMode)', or you could use the preferred option of setting the environment variable "GIN_MODE" to "release".
2) You could also run the server in production. This requires mostly the same steps, but you would like to instead use an executable and run everything dockerized. You can find all files needed in the 'production' folder. In the base folder, you will find a 'docker-compose.yml'. Again, make sure the environment variables are set how you would like them to be. In the service folder, you will find a compiled version of the main.go called 'backend' file, a '.env' file and a 'Dockerfile' file. To run the newest version of the server dockerized in production, first compile the executable by going to the base folder of the and using the command 'GOOS=linux GOARCH=arm64 go build -o backend'. The first two variables indicate the system you would like to build for and 'go build -o backend' specifies that you would like to compile the program to a file called 'backend'.

After compiling, replace the 'backend' file in the services folder with the new one you just created. After performing this step, running 'docker-compose up –build -d' will run the back-end service, starting both the database and the back-end server.

E.3 Technical Overview

The project has, due to the nature of Flutter, a very hierarchical system. This section of the manual aims to explain the different levels and components within this hierarchy and how they work. As such, first, the upper layer will be discussed, and every succeeding section will expand on a component in a lower hierarchical layer. The last layer, consisting of widgets will be explained in a table describing the component name, where it is used and what it does. Although these widgets are almost exclusively used in one screen, they have been separated because this allows for less nested code and the possibility to use it elsewhere in the future as well as ensuring a modular system.

E.3.1 Front-End client-side

The client-side part of the application starts with two pages. Although these are hierarchically on the same level, they appear sequentially in the application. The login page is naturally the first page that is seen by the user. When the login is successful, the `main_page` will be shown. The `main_page` controls the navigation between its child pages and subsequently is also responsible for the navigation between the pages. It manages the bottom app bar accompanied by the floating action bar that opens the bottom sheet to control the sensors. This bottom sheet is not in a separate file.
These two pages can be found next to the other pages in the Screens folder. All the major screens are available in the screens folder. The extracted components that these screen use can be found in the Widgets folder. This folder contains smaller widgets that do not have any functionality themself and are mostly UI components. For example, the `home_page` uses the `custom_app_bar_item.dart` as a custom class for the items on the bottom app bar. Most of the functionalities related to the exercises are in the Exercise folder. This folder contains the `main_exercise_page.dart` which is the main scaffolding for the exercise. It maintains the state of the exercise, manages the data and controls the flow. In its body, the other bodies, which can also be found in the Exercise folder, are inserted into the `main_exercise_page.dart`. They are given certain callback functions to influence the state of the `main_exercise_page.dart`, such as going to the next screen and passing on data. Finally, when the exercise is completed, the `main_exercise_page.dart` can simply be popped and the entire tree is disposed of. Lastly, there is also the Sensors folder where most of the sensor files from the original Awear application are.

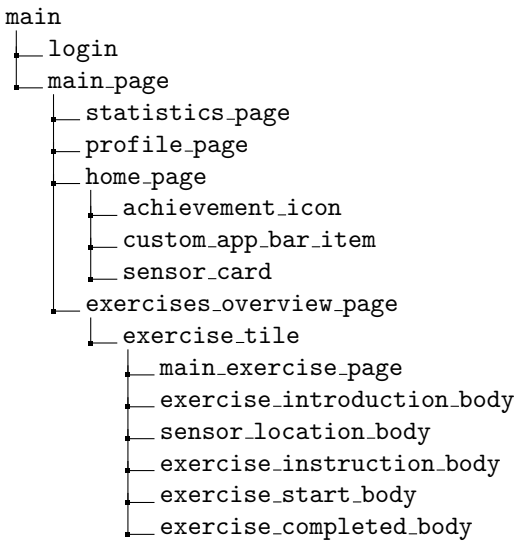


Figure 5: A simplified view of the hierarchical page structure of the client-side

E.3.2 Front-End supervisor-side

On the top level of the supervisor-side, there are five files, namely, `auth`, `color_scheme`, `data`, `main` and `routing`. `main.dart` is the file in which the app is initialised and run. It includes a route parser, which is a list with allowed paths. If the user were to try a different path than the ones included in this list, they would get redirected to the default path. Meaning, that if future development were to include new screens that require new paths, they should be appended to this list. `main.dart` includes the other top-level files in the initialisation of the app. `auth.dart` is the file responsible for the authentication of the application. Currently, this uses a stub implementation, but in the future, this can be expanded to use an actual authenticator service to receive user info. `color_scheme.dart` is the material 3 colour scheme that is used to colour everything in the app. `data.dart` and `routing.dart` both combine every file in the corresponding directory into a single file that can be imported, reducing clutter. If any file would be added to the data directory for example, all the user has to do is add this file to `data.dart` and any file automatically import it. Besides the data and routing folders, there are folders for screens and widgets.
The files found in the data directory are classes that contain all the data structures. Some of these classes are fetched from the database and therefore have functions to go to and from JSON format, allowing them to communicate with the REST API. The files in the routing folder handle the URL creation, URL parsing and redirection. These files handle any pagination that happens on the application.
The screens folder contains the visual aspects of the system. This is ordered in a hierarchical manner as well. First, there is `nagivator.dart` which switches between the screen when a user is not logged in and the screen where the user is logged in. This top-level navigation only changed between these screens because any deeper-level screens have shared assets, allowing a modular and hierarchical system. The screens that are switched between are `home_page.dart` and `sign_in.dart`. `home_page.dart` has the top navigation bar, that allows for switching between the

different pages, as well as logging out. This page has `home_page_body.dart` as the body, further supporting the modularity. This file takes care of showing the different screens based on the tab, either the client screen or the exercises page.
The client screen consists of a list of clients assigned to the user, and when a client is selected, a client detail screen is displayed on the right. Some widgets are displayed in this detail screen, showing data fetched from the database.
The exercise screen consists of a search bar and filter chips, and the list view with fetched exercises from the database. The contents of the list are filtered based on the search and filter chips. When assigning an exercise set, there is a bar on the right side of the screen. This sidebar contains a client selector dropdown and a form with the necessary data. A simplified view of the hierarchical page structure is found in figure 6.

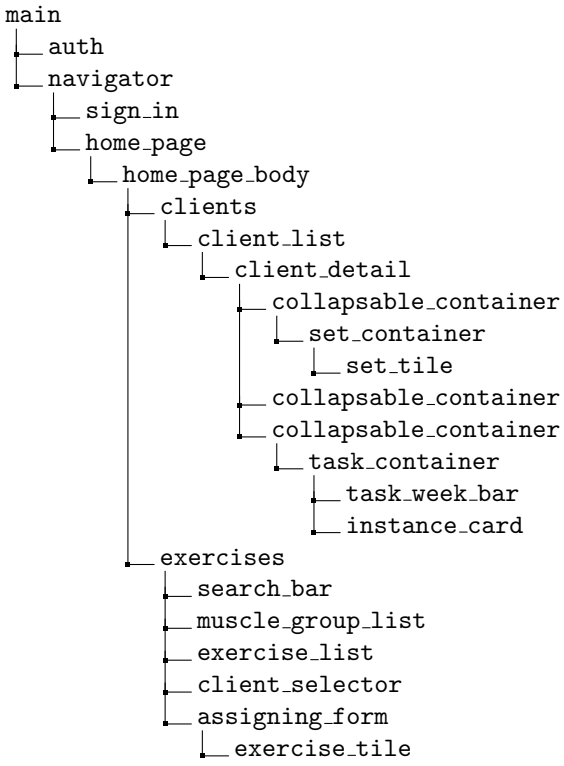


Figure 6: A simplified view of the hierarchical page structure of the supervisor-side

E.3.3 Back-end

We have a main file which starts the API. Packaging in **Go** works by having a different folder for each package. When looking at the 'api' folder, we can see the 'api.go' file which we just called and three other folders. In the API file, we load all environment variables, initialise the database and start the server, fill the database with data and then run the server on the IP address specified in the string. The other three folders contain the `models`, `seeder` (which is used for filling up the initialised database) and `controllers`, which handles most of the logic of the application. The seeders functionality is dropping all existing tables, recreating the tables and then filling the

dummy data in the database. If you do not need the seeding again, you can get rid of this functionality by not calling the `seeder.Load`.
Then we have the `models` package. The `models` contains all models which are in the database. All of these models have methods, which communicate with the back-end. Communication with the database is done using **GORM**, an ORM library for **Go**.
Finally, we have the controllers package. The controller's package contains a controller for every group of endpoints. In most controllers, we call the designated method for achieving the goal of each endpoint. For example, performing a get for all clients performs the method call on the clients model which returns all clients stored in the database. In some cases, more has to be performed in order to call the method. For example, a POST request first requires the parsing and validating of the sent data before we can pass it to a method which saves this data.
Next to this, the controllers contain the setting up of all routes, done in `routes.go`. All routes have a base URL, followed by the type of endpoint, and finally, they all have a specific request type, URL specification (the requirement of an ID or not) and the server method which will handle the request. When we combine all of those features, we end up with a fully functioning back-end.
At the base of the project, we have the '.env' file for environment variables. Next to this file, we have the main file and a 'go.mod' file. This file contains all dependencies needed in order for the application to run. Last, the base folder contains four folders, the `api` folder which contains all the server logic, the `docker` folder, which contains the test database, both for testing the application and running the application locally, the `production` folder, which is an example deployment of the back-end service and finally the `tests` folder which contains the unit tests for the back-end. These are subdivided in controller and model testing.

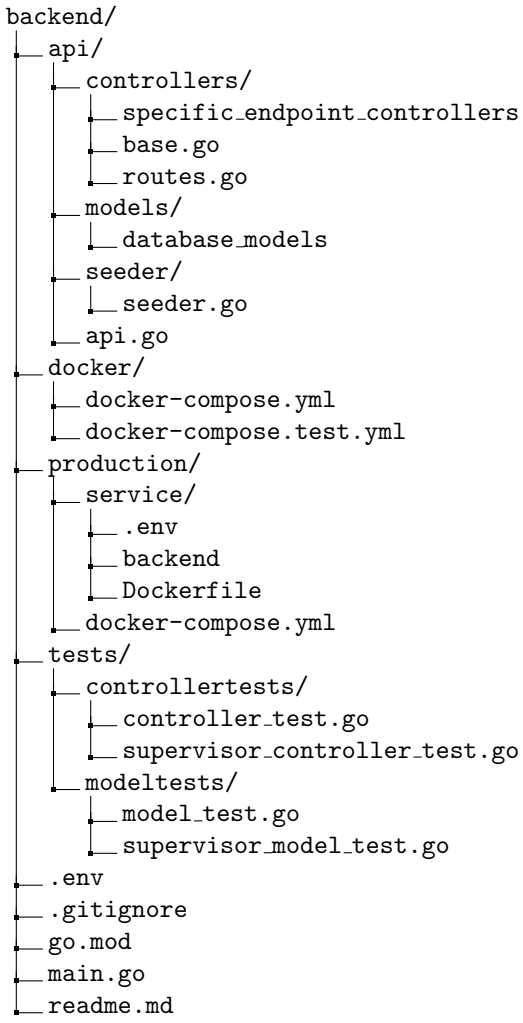


Figure 7: The file structure of the back-end

E.4 Features

E.4.1 Front-End client-side

The core functionality of the system is performing exercises. This has also been the main focus point for the project. It is possible to complete an exercise with dummy microphone data as a replacement for the actual sensor data. There are five steps and accompanying screens to completing an exercise:

- Overview of the exercise set containing an introduction, all the exercises to be completed and their corresponding points
- A screen that explains where the sensors are supposed to be for these exercises
- An individual screen for each exercise instructing how the exercise should be performed

- A screen that shows how the exercise is going
- Finally a screen will be displayed when the exercise set is completed. This will also tell you how many points you got and what achievements you made progress for

Once the exercise is completed, the data is then sent to the supervisor-side where the supervisor can see and analyse the data. On the exercises overview page, the exercise is marked as complete and the upcoming exercises are displayed.
Next to the features for the exercises, there are also some auxiliary features. The home page functions as the main landing page for the users. It gives an overview of the three main features that are important in the design of the application, but only two are implemented currently. It shows how many points the user has achieved and it gives a quick overview of the upcoming exercises. Both of these elements get their information from the back end. At the bottom of the list of upcoming exercises, there is a button that will immediately start the first exercise. There are two smaller features that are available but not fully complete. There is a profile page that allows the user to log out of their account. Lastly, there is a window for the sensors as they play a vital role in the system. This window allows the user to add, but not remove sensors. Right now the sensors are purely fictional, but the added sensors do show up on the sensor placement page of the exercise. They are currently required to complete the sensor placement.

E.4.2 Front-End supervisor-side

The front-end for the supervisor mostly consists of two sides, namely the client and the exercise side. On the exercise side, there is also an assignment view to assign exercise sets to the client. When arriving at the client page, the user will first be met with an empty screen telling them to select a client from the list on the left. When a client is selected, information about them can be viewed. However, if the client has never been assigned an exercise set, there is no information to view anything yet, so this is the first step. There are many ways to get to the assigning page, and this will be explained further on. For now, a client with existing exercise sets will be assumed in order to explain the rest of the client page.
The client page has the functionality for multiple tabs as the client overview might need to be expanded in the future. However, for now there is only functionality on the first tab which is what will be discussed in this section. There are four sections to the client page. Namely, the profile bar, exercise set overview, progress section and tasks. The profile bar displays basic information about the client and allows the supervisor to switch between tabs. The exercise set list consists of cards displaying the exercise sets assigned to the client. These cards show the exercise set info and allow the user to edit it. The progress section visualises the points acquired by a client. The tasks section shows instances of exercise sets, both history elements that are fetched from the database when they are completed by a client and instances for the future, calculated based on the interval and time range. The history instances have some movement data sent with them which can be viewed when inspecting the instances.
The other page is the exercises page. This page allows the user to search and filter on specific exercises, as well as create new exercises. Currently, this is a stub implementation of adding an exercise, as the movement data is still a work in progress. Next to displaying the exercises, this screen also functions as the assigning page. To assign an exercise, you have to arrive at the assign view of the exercise screen. There are multiple ways to do this:

- In the client screen, Via the floating action button on the bottom right of the client page.

- In the client screen, When there are no exercise sets assigned to a client, the section will show a text with a link 'assign exercise set here'.
- In the exercises screen, when you click on an exercise.
- In the exercise screen, when you click on the todo-list icon.

Furthermore, you also get to the assign view with the pencil icon of an exercise set on the client page. However, this is for editing an exercise set, not for creating a new one. When in this view, the client can be selected through a drop-down menu. Then, basic information about the exercise can be filled in like the title and description. Afterwards, exercises can be put in the exercises by clicking on them in the list on the left. The order can be altered by dragging the exercises in their list. The buttons on the bottom can be used to assign an exercise set or to cancel.

E.5 Usage

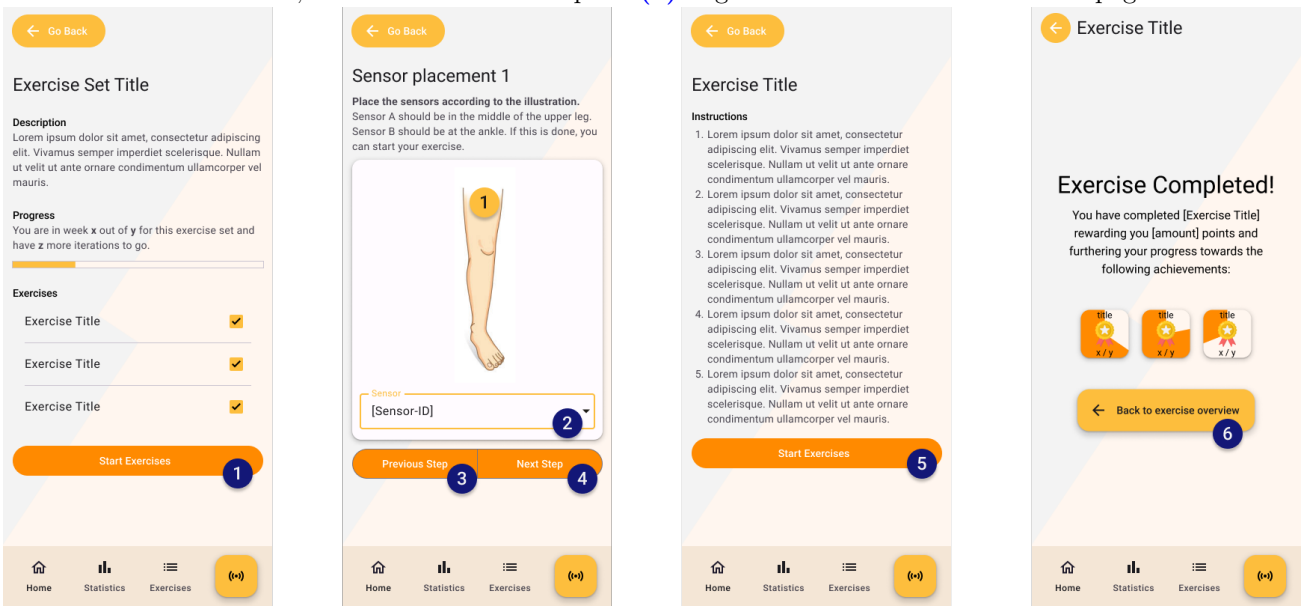
E.5.1 Front-End client-side

The following section will describe how to perform certain actions and how to experience certain flows in the system.

To perform an exercise, the user can either press **(1)** to immediately start the first exercise available. The other option is to go to the exercises screen and from there open an exercise. This can be achieved by pressing button **(2)** and then button **(4)**. Before the user can start with an exercise, the user needs to make sure that the sensors are working by going to the sensor screen. This can be done by pressing **(3)**. The second screen shows the overview of the exercises. Only the middle exercise is ready to perform. This can be done by pressing on the card with the **(4)**.

When the user has started the exercise, he will go through the following set of screens that will guide him through the exercise. Pressing **(1)** will start the process leading up to the exercise. This is a point where the user would commit to starting, rather than looking just at the exercise. This will lead the user to the second screen. Here the user can press the dropdown menu near **(2)** to start selecting the sensors. When the right sensor is accompanied by the right placement, **(4)** can be pressed to go to the next sensor placement. If a mistake was made, then the user can always press **(3)** to go to the previous sensor placement. When the sensor placement is done, an instruction for the upcoming exercise will be shown such that the user understands what he is about to do. When that is all clear the user can actually start the exercise by pressing **(5)**. After the exercise is completed

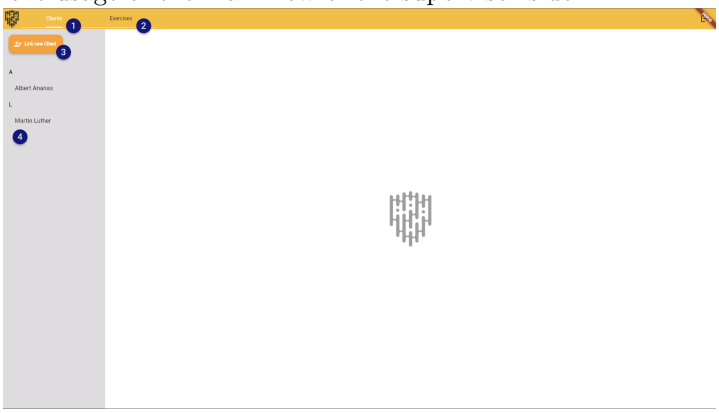
and the data is submitted, the user will be able to press (6) to go back to the exercises overview page.



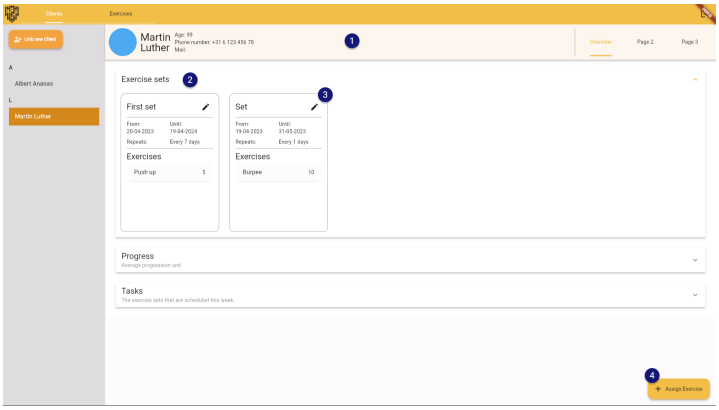
E.5.2 Front-End supervisor-side

The following section will explain the usage of the main flow of the supervisor side.

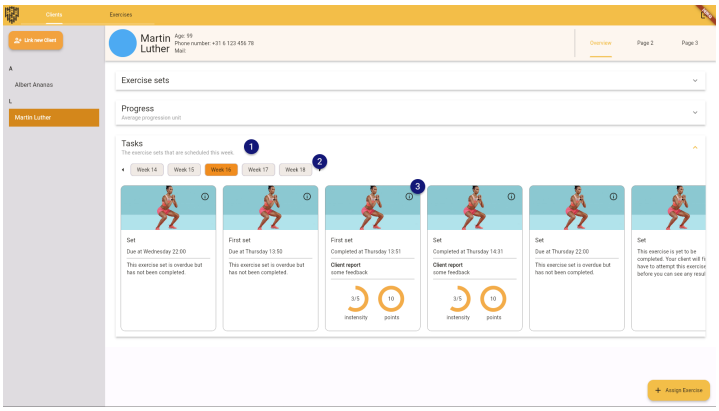
After logging in, the user is greeted by the home page. The user can switch pages using the navigation bar and the tabs. Tab (1) is the client's screen, and tab (2) is the exercises and assigning screen. The client's screen shows a list of clients that are connected to the current user (4). To connect more clients, you can press the action button (3).



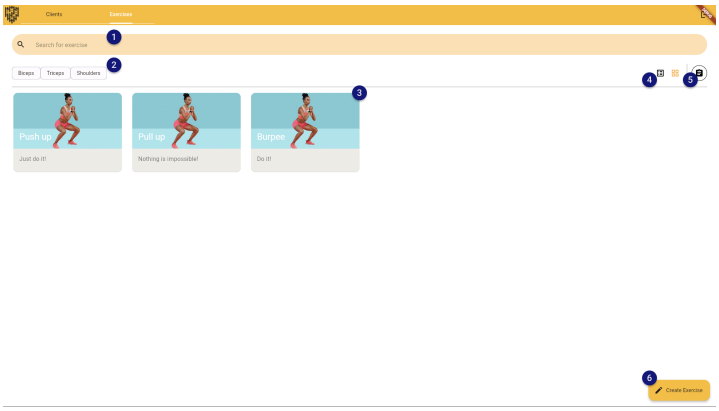
When a client is selected, the detail screen is shown to the user. Consisting of the profile bar (1), collapsable containers displaying relevant information (2) and a floating action button to assign more exercises to this client (4). The exercise sets container (2) displays the assigned sets for this client and displays each in a card (3), which shows the relevant information and allows the user to edit the exercise set.



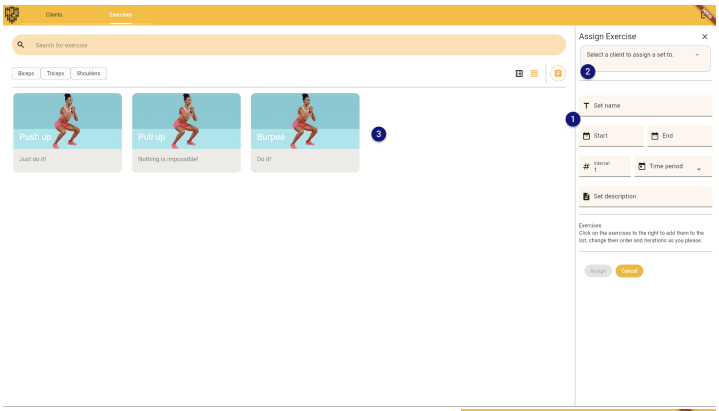
The tasks container (1) fetches any exercises completed by the client and calculates any future and overdue exercise set instances based on the interval, and displays these on a weekly basis. The user is able to switch weeks using the week chips (2). When the user wants more information, they can click on the exercise card (3) to display the set information and any movement data.



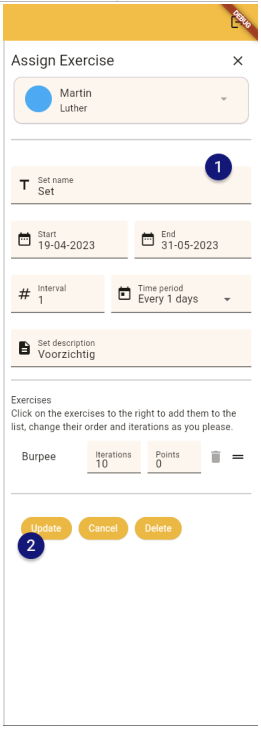
The exercises page consists of a search bar (1), filter chips for the muscle groups (2) and the queried list of exercises (3). The user is able to switch between a grid view and a list view (4), and when they want to add another exercise, they can use the floating action button (6). When the todo button is pressed (5), the assigning sidebar is opened.



When assigning an exercise set, the side bar opens up (1). There are multiple ways to get to this screen, and in this case no user has been selected. The user can select a client to assign to using the dropdown (2). To add exercises to the set, the user can press on the exercises in the list (3).



The user can fill the form (1) with the set data, such as a title, instructions, interval and date range. When the user is satisfied with the set, they can assign it using the action buttons (2). A popup will show with the finalised exercise set, and the user can return back to the clients page.



E.6 Testing

The following section will give an overview of how certain tests are implemented on the front-end side and on the back-end. What kind of test it is, how to set it up and what variations are possible for it.

E.6.1 Front-End supervisor- & client-side

Unit tests
Due to the simplistic nature of unit tests, it is not complicated in Flutter. It requires a built-in test package. This package allows you to create tests and groups of tests. Next to that, it provides elementary functions like expect(expected value, actual value). Below is an example of how a unit test could look like. Here we can see that there needs to be a main to run the test. In that main class, there is a group of tests, that can also be run separately from the whole file. In this test, we update a value and expect that value to be increased. The test can be varied in a lot of ways depending on how much functionality a unit has.

```
import 'package:test/test.dart';

void main() {

    group('some description', () {
        test('some description', () {
            Element element = Element();
            element.incrementValue();
            expect(element.value, incrementedValue);
        });
    });
}
```

To test any functions that have calls to the database, we can simulate the returns using Mockito. Mockito can catch a specific request to the server and return a given answer. In this test, we request a client from the database and test whether the JSON decoding works correctly.

```
@GenerateMocks([http.Client])
void main() {
    test('returns a Client if the http call completes successfully', () async {
      final client = MockClient();

      when(client
              .get(Uri.parse('https://jsonplaceholder.com/client/1')))
          .thenAnswer((_) async =>
              http.Response('{"userId": 1, "username": "user"}', 200));

      expect(await fetchClient(client), isA<Client>());
    });
}
```

Widget tests

Just like unit tests, widget tests are not complicated in Flutter. Widget tests can help test whether widgets keep working as expected, even when later iterations change it. Below is an example of what a widget test looks like. It can be used to test the entire system or a single widget.

```
void main() {
  testWidgets('MyWidget has a title and message', (tester) async {
    await tester.pumpWidget(const MyWidget(title: 'T', message: 'M'));
    final titleFinder = find.text('T');
    final messageFinder = find.text('M');

    expect(titleFinder, findsOneWidget);
    expect(messageFinder, findsOneWidget);
  });
}
```

E.6.2 Back-end

In the backend, we make use of two tests: Unit tests and endpoints tests.

Unit test

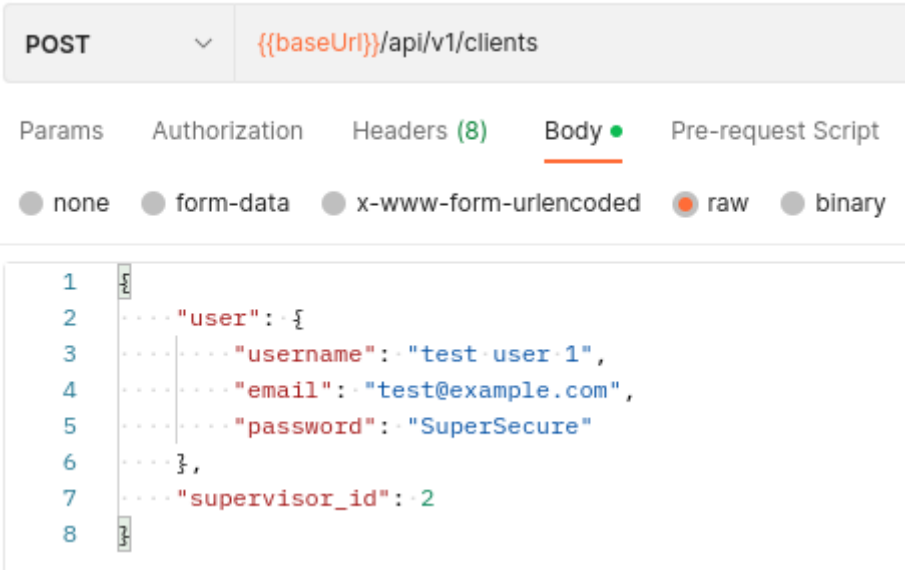
Unit testing in go is built in with the toolset that Go has. To run any test in Go, run the 'go test' command in a repository which contains tests. Using a 'TestMain()' function makes sure that all tests can use an already set-up database. An example of a unit test can be found below in order to test all functionality of the server.

```
func TestFindAllPosts(t *testing.T) {

        err := refreshUserSupervisorAndInstitutionTable()
        if err != nil {
                log.Fatalf("Error tables %v\n", err)
        }
        _, _, _, err = seedUsersInstitutionsAndSupervisors()
        if err != nil {
                log.Fatalf("Error seeding tables %v\n", err)
        }
        supervisors, err := supervisorInstance.FindAllSupervisors(server.DB)
        if err != nil {
                t.Errorf("this is the error getting the supervisors: %v\n", err)
                return
        }
        assert.Equal(t, len(*supervisors), 2)
}
```

Endpoint test

In endpoint testing, we send a request to an already running server. In the test, we can look at how long the server takes to respond, what message we get back and what the status code is of the message. With this, we can check whether the server outputs intended behaviour when interacting with it. We have done this in postman. Here you can build an entire collection and run all tests at the same time, to speed up the process of testing.



E.7 Extensions

Here we explain how the current code base can be extended and we gice some recommendations on how to do this.

E.7.1 Front-End client-side

There are a number of possibilities to extend the client-side application. For each of the features, it will be explained why the feature should or could be added and how to add them.

- The first and perhaps most important feature is implementing and connecting the movella sensors. The platform needs to have sensors to accurately capture and display movement from the clients. Only then will the platform be able to see something worthwhile about exercises. This is done by following the steps. In `lib/object_classes/SensorList.dart` is an object that maintains the current sensors. This class should be extended to accept SensorBase in its list. This SensorList is exposed to all the widgets by the provider found in `lib/main.dart`. The floating action button for the sensor screen is in the file `/lib/main_page.dart`. The bottom sheet that shows in response to the pressing of the button starts from row 147. The list shows the current added sensors. Each card could be updated to display additional or more appropriate content. The dialog that shows after adding a sensor should be extended to display the available sensors like the DiscoverPage in `lib/Screens/DevicesPage.dart` does in the original Awear application. Optionally the add sensor button can also show an entirely different screen. Once the connection is established with the sensors, most of the work should be done. On the sensor placement screen in `lib/Exercise/sensor_location_body.dart` within the exercise, the name of the sensor should be accessible such that the user can distinguish between them. In the `lib/Exercise/exercise_start_body.dart` the sensors are accessed to capture the data of movement. A choice should be made about the starting of the sensor capture, either this can

start right after showing the screen, then this should be added to the initState function. Otherwise, a button should appear that start to data capture. Currently, there is a submit button that calls two functions. The first function performs a callback to the `/lib/Exercise/main_exercise_page.dart` to go to the next exercise or to the completed page. The other function performs a callback to the data capture to add an array of data to the data already stored on the main exercise page. This function can be reused to provide the flow of data. These steps should allow the use of movella sensors. Some additional things that need to be thought about, are displaying correctly during the exercise and on the supervisor's side.

- A second extension that would add a lot of positive user experience to the application is a statistics page. Although the system provides the possibility to perform an exercise and for the supervisor to review the data, however, there is no overview of completed tasks for the client. Adding a statistics page could further motivate the user to perform exercises and see their stats go up. Right now only the points element on the home page gives an update on the user's progress. A graph can be shown on the statistics page about the progress of the user's points throughout the month. A graph needs to be made based on the information from the API. A call needs to be made to `/histories/user_id` with a GET request. This will return a list of the exercises that have been completed by the users. This shows the date on which it has been completed and the tasks and accompanying points the user has achieved. Using this data, a timeline can be made with the cumulative points.

- Currently there is no error handling in most of the client-side application. For a positive user experience and for preventing some bugs or perhaps unwanted features, this would also be high on the priority list. There is for example no error handling for when an exercise is stopped halfway through, for the data that arrives from the API or when certain elements in the application are empty. A lot of items could be empty like the exercise list, the sensor list or the achievements, however, it is not always caught.

- Achievements are currently displayed in the front end as placeholders, but naturally can be extended to contain actual information. They have been designed to be a certain threshold for points or exercises completed within one specific muscle group. The overview of the current exercises, it could show what achievement that exercise could work towards. Suppose that that exercise set contains exercises within the arms muscle group, it could work towards an achievement for the arms. For this to be a functionality, it would require more than just work on the front-end. It is already considered in the database, but not in the sense of endpoints. It is however not extremely heavy on the client-side of the application since most of the information will be coming from the API. This information would just need to be fed to the already existing placeholder on the home screen and exercise overview screen.

The aforementioned features are some of the major features that could be implemented in the near future. However, it is also worth mentioning what kind of smaller features could already make a difference. These features should not be neglected although they seem small.

- It is currently only possible to add sensors and not remove them. In the spirit and given the importance of the sensor, being able to remove them is not much work, but will include core functionality.

- Right now there is some UX missing in the sensor placement body. Whenever an exercise would require two sensors to be placed, there is a lack of UX confirming when a sensor is

actually allocated to that spot. It could very well be the case that a user clicks on the sensor placement without the awareness of needing to assign different sensors to different places.

E.7.2 Front-End Supervisor-side

The current implementation of the supervisor-side includes many placeholders for future expansions. These expansions are hard to predict, as they can be specific to certain use cases. The main functionality and process flow is existent, and any further extensions can add to this experience.
Another expansion to take into consideration is the data display and management. The progress container is currently a placeholder and can be used to display specific trends. Most of these decisions go hand in hand with any expansions in the back-end or client-side, as the supervisor side is mostly a place to display the data that is being fetched.
Lastly, there is no real authentication working within the application, but this will be explained further in the next section.

E.7.3 Back-End

While the back-ends functionality is mostly complete, it could be further extended with the following two features which we would have loved to add:

- Right now, the application is missing an authorisation system. This can be implemented using middleware in the back-end, where certain endpoints would be locked behind an authentication middleware, which can only be accessed if you have logged in and received a JWT token, also known as a Bearer token. Using this token, you can put an authorisation lock behind certain actions in the database, thus securing the system. This could be simply added by extending the back-end further with code to support this. The front-end needs to adapt to this by sending a Bearer token with every request, but this is trivial to add.
- Finally, we would have loved to have some documentation on all of the endpoints available in the system. Right now, the Postman package has a test on all endpoints available, so you would be able to find it from there. There is a better solution for this, however. You could document the endpoints using a swagger, a system for documenting the API. There is also a library in Go called Swaggo. This library will generate the swagger documentation using annotations above the code which handles a certain endpoint. This documentation could then be accessed by an endpoint, thus giving access to a better-documented endpoint list.

F Consent Form for awear design project user test

Please tick the appropriate boxes	**Yes**	**no**
Taking part in the study		
I have read and understood the study information dated 20/04/2023, or it has been read to me. I have been able to ask questions about the study and my questions have been answered to my satisfaction.	○	○
I consent voluntarily to be a participant in this study and understand that I can refuse to answer questions and I can withdraw from the study at any time, without having to give a reason.	○	○
I understand that taking part in the study involves performing some tasks on a platform where I can be timed and some questions can be asked	○	○
Use of the information in the study		
I understand that information I provide will be used for the results of a 'Design Project'	○	○
I understand that personal information collected about me that can identify me, such as [e.g. my name or where I live], will not be collected or shared beyond the study team.	○	○

Signatures

---	---	---
Name of participant [printed]	Signature	Date

I have accurately read out the information sheet to the potential participant and, to the best of my ability, ensured that the participant understands to what they are freely consenting.

---	---	---
Researcher name [printed]	Signature	Date

Study contact details for further information:
Matthias Wentink m.wentink@student.utwente.nl