BABEŞ-BOLYAI UNIVERSITY CLUJ-NAPOCA
FACULTY OF MATHEMATICS AND COMPUTER SCIENCE
SPECIALIZATION COMPUTER SCIENCE

DIPLOMA THESIS

HeadFlow: Where Earable Computing Meets Kinesiotherapy

Supervisor
Lector universitar, Dr. Cojocar Dan

Author
Andrioaie Daria Maria

2023

ABSTRACT

This thesis explores the potential of earable technology, a subfield of wearable computing, focusing on devices worn around the user's ears. The goal of this paper is to demonstrate the application of earable technology in assisted rehabilitation for cervical spine issues. HeadFlow, a native iOS mobile application, is presented as a tool that utilizes AirPods sensors to gather head movement data and provide progress updates to both patients and therapists. The novelty of the paper consists in the fact it is the first application in the field of assisted rehabilitation that uses earphones as a tool to maps the sensors data to concepts of kinesiology.

For the first part, the paper describes the rising frequency of cervical spine issues and potential their risk factors, emphasizing kinesiotherapy as a solution and discusses effective exercise strategies. It examines how technology can aid therapy, including tele-rehabilitation, motion analysis systems, and software programs that employ motion sensors and interactive games. For the second part, the paper outlines the specifications of the proposed application, describing its use cases and entities. It then delves into the technical details of the implementation, covering the software architecture, design patterns, data processing, and implementation details of each use case.

Overall, this thesis contributes to the field of wearable computing by showcasing the potential of earable technology in assisted rehabilitation and providing a comprehensive understanding of the development and implementation of a specific application, HeadFlow, for cervical spine rehabilitation.

I have neither given nor received unauthorized assistance on this thesis. This work is the result of my own activity.

Contents

1 Introduction **1**
- 1.1 What is Wearable Technology? . . . 1
- 1.2 The Goal of This Paper . . . 1
- 1.3 Brief Description of Chapters . . . 2

2 Neck Pain Epidemiology **4**
- 2.1 Risk Factors of Cervical Spine Issues . . . 4
  - 2.1.1 Improper Posture at the Workplace . . . 5
  - 2.1.2 Psycho-Social Risk Factors . . . 6
- 2.2 Where and How is Pain Felt . . . 7
- 2.3 Kinesiotherapy as Treatment and Prevention . . . 7
  - 2.3.1 Most Effective Exercises for Neck Pain . . . 8
  - 2.3.2 The Average Range of Motion of the Cervical Spine . . . 9

3 Where Technology Meets Kinesiotherapy **11**
- 3.1 Microsoft Kinect . . . 12
  - 3.1.1 EvolvRehab . . . 13
  - 3.1.2 MIRA Rehab . . . 14
- 3.2 The Accountability Issue . . . 14

4 Specifications of the Proposed Application **17**
- 4.1 Use Cases . . . 17
- 4.2 Model . . . 19

5 Implementation Details **22**
- 5.1 System Design & Architecture . . . 22
  - 5.1.1 SwiftUI and UIKit iOS Client . . . 23
  - 5.1.2 Node.js Server . . . 25
  - 5.1.3 Mongo DB . . . 27
- 5.2 Authentication Flow . . . 28
  - 5.2.1 One-Time-Password Sign In . . . 29

5.2.2 Social SignIn 31
5.3 Stretching flow 32
5.3.1 CMHeadphoneMotionManager Library 33
5.3.2 Path Drawing in SwiftUI 35
5.3.3 Saving the progress 37
5.4 Progress Monitoring 37
5.5 Therapist-Patient Collaboration 39
5.5.1 How They Connect 40
5.5.2 Session Tailoring 42
5.5.3 Session Feedback 42
5.6 Local Notifications 42

6 Conclusions **46**
6.1 Future work 46

Bibliography

List of Figures

2.1 The worldwide prevalence of cases of neck pain, by age and sex, extracted from the Global Burden of Disease Study. 4
2.2 The effects of prolonged Forward Head Posture 5
2.3 Most common pain points, extracted from [ea90] 8
2.4 The Zander Apparatus 9
2.5 Comparison of improvements, extracted from [You16]. 10
2.6 Types of cervical movement, extracted from [Yük18] 10

3.1 How data from sensors placed on the body is mapped to a kinematic skeleton, using Nexus. Picture extracted from one of Vicon's usage guides, available on their website [Vic]. 12
3.2 The usage of Microsoft Kinect for at-home rehabilitation exercises, extracted from the Official Microsoft Blog 13

4.1 The use case diagram of the software system 20

5.1 The architecture diagram of the software system 22
5.2 The architecture diagram of the iOS client app 23
5.3 The architecture of the Node.js server 26
5.4 The Render console 27
5.5 Querying the collection of users in the MongoAtlas cluster 29
5.6 The sequence diagram for the Register feature 31
5.7 Authentication flow (1) 32
5.8 Stretching Executor 33
5.9 A visual representation of Yaw, Pitch and Roll available in CMAttitude, extracted from Apple Developer documentation. 35
5.10 Saving the summary of the completed stretching session 38
5.11 Stretching history 39
5.12 List of patients 40
5.13 Handle collaboration with therapist 41
5.14 Tailoring the stretching session of a patient 43
5.15 Session feedback, from both perspectives 44

5.16 Local notifications and permissions status 45

Chapter 1

Introduction

1.1 What is Wearable Technology?

Wearable computing is an emerging technology in the field of Computer Science and it refers to the use of any kind of electronic device that is worn on the body, usually meant to gather and process data from sensors and transmit it to connected devices, like laptops or mobile devices, in real-time. Some examples of such devices include jewelry, pieces of clothing, medical devices, glasses, headsets, and headphones and they provide a wide range of usability, from health and fitness, to interactive entertainment and assistance.

Since the first wearable computer that history has seen in 1955, a tiny four-button device that could fit into a shoe to help gamblers in casinos cheat at roulette games [1], the industry gained massive popularity. In the 2000s, Bluetooth headsets, Fitbits and Nike + iPod Sport Kit were introduced. Google Glass first appeared in 2013 and the Apple Watch in 2015, while The Oculus Rift VR gaming headset debuted in 2016. In the 2020s, the trend is going towards the development of AR and VR headsets that provide immersive experiences and the research of smart clothing and earables. [2]

1.2 The Goal of This Paper

This paper aims to demonstrate a possible use of earable technology, a subfield of wearable computing, that refers to devices that are worn around the user's ears. It does so by presenting HeadFlow, a mobile application that acts as a tool for assisted rehabilitation in cervical spine issues, by gathering head movement data from AirPods sensors and making the progress available both to the patient and their therapist, with a handful of other features, like daily reminders to practice and the possibility to tailor the stretching session according to each patient's needs.

My motivation to develop this application lies on 2 main pillars. Firstly, several of my family members work in the medical field, serving people in need, including my parents, who are also certified masseurs. Because of this, the idea of developing a health-related app, dedicated to patients and medical professionals and meant to ease their collaboration and make it more efficient, only came naturally. Second of all, I enjoy performing stretching exercises throughout the day and I find that it helps me relax and ease out the aches and pains that accumulate so easily. As a consequence, I was intrigued by the idea of building an application that would allow other people to discover the benefits of stretching as I did.

My original contribution to the field of earable computing is given by the translation of the data gathered from AirPods sensors, which is expressed in rotations on the three axes of the three-dimensional space Cartesian System, into a visual representation that describes the head movement of the user and classifying this data into a comprehensive report that can be used by medical professionals to understand the range of motion of their patients, without the classical measuring method of the goniometer.

1.3 Brief Description of Chapters

Chapter 2 defines cervical spine issues and their rising frequency and explores the possible risk factors that are associated with this problem, such as poor posture at the workplace, a sedentary lifestyle, as well as psycho-social factors, like stress, anxiety and depression. It highlights kinesiotherapy as one of the solutions to this arising problem and which strategies of exercise yield the best results for the patient, alleviating his pain and increasing his mobility.

Chapter 3 explores how advancements in technology can be utilized in therapy, reframing technology as a potential solution rather than an enemy. Two ways to achieve that goal are through telerehabilitation and motion analysis systems, which employ cameras and sensors to track patients' movements and provide real-time feedback on their technique and form. The chapter also discusses specific software systems like Vicon, Optotrak Certus, and Cortex, commonly used for movement analysis. The Microsoft Kinect is another revolutionary technology discussed, known for its application in various fields, including assisting in rehabilitation. It has been used to develop rehabilitation-based software programs like EvolvRehab and MIRA Rehab, which utilize motion sensors and interactive games to engage patients in recovery exercises. The chapter also addresses the issue of patient accountability in home exercise programs and how technology can help overcome this challenge. Features such as in-app coaching, goal setting, and self-monitoring can

be implemented in software to reinforce the healing process and improve adherence to treatment plans.

Chapter 4 outlines the specifications of the proposed application, designed to address chronic neck pain. It describes each use case in detail, for each flow of the app: Authentication, Stretching Executor, Stretching History and Patient Coaching. It also aims to describe each entity from the domain of the application and the relevant data that they are made up of User, Collaboration, Stretching Exercise and Stretching Session.

Chapter 5 dives into the technical details of the implementation, starting with the architecture of the software system and how the components: client, server, and database interact with each other. For each of them, it describes the relevant design patterns that were used, the general approach to processing data, and additional helper tools, such as SDKs or deployment platforms. It also does in-depth on each use case, describing how it was implemented and providing a sequence diagram and pictures of the feature in execution.

Chapter 2

Neck Pain Epidemiology

2.1 Risk Factors of Cervical Spine Issues

Neck pain and, subsequently, decreased neck mobility, is a common problem that affects people from all walks of life nowadays. At first, one may think that it only concerns patients that suffered injuries or maybe students and workers that spend a significant part of their time sitting. In fact, researchers consider it to be a widespread issue in modern society [Kaz22], due to several factors, that, from one day to another, build up tension around the neck and shoulders. An analysis of the Global Burden of Disease Study, performed in 2017, shows that the number of prevalent cases of neck pain was as high as 288.7 million, and it mostly affects people in their adulthood, [Saf20] as it can be seen in figure 2.1.

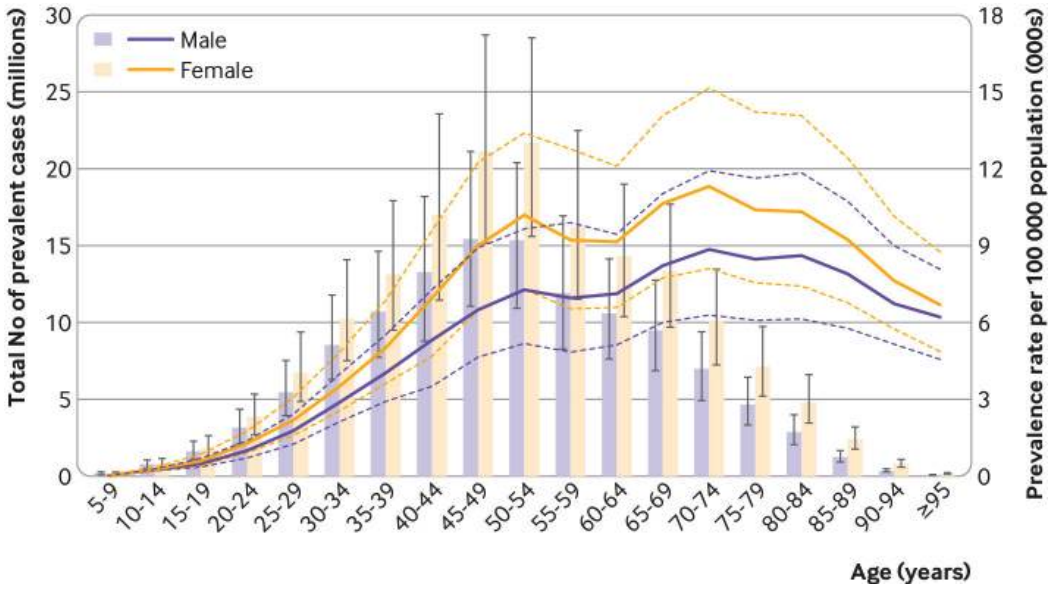


Figure 2.1: The worldwide prevalence of cases of neck pain, by age and sex, extracted from the Global Burden of Disease Study.

2.1.1 Improper Posture at the Workplace

Sometimes, this pain can be attributed to technology use, when individuals assume improper postures while watching television, browsing through social media, or working in front of a computer screen. The detriments of sedentary lifestyles and desk jobs are increasingly becoming a concern, as sitting for prolonged periods has been linked to incurable diseases and adverse effects on spinal health. Despite this, work environments continue to encourage prolonged sitting, which can result in individuals assuming poor postures, such as Forward Head Posture (FHP) or 'computer neck'. FHP is a condition that occurs when the spinal system adjusts itself to counteract gravity, leading to long-term health and posture issues. FHP can result in muscle imbalances where certain muscles become elongated and weakened, such as the longus colli muscle, while other muscles become shortened and tightened, such as the suboccipital muscles, as it can be seen in figure 2.2

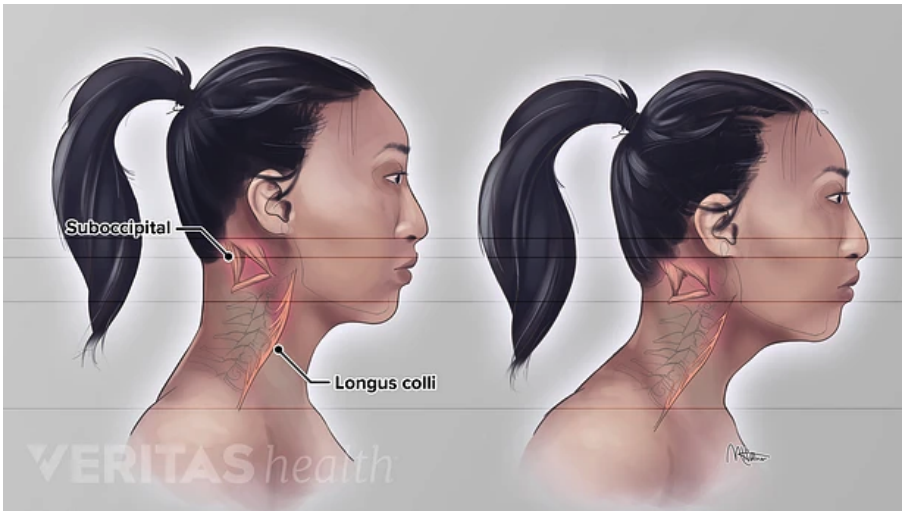


Figure 2.2: The effects of prolonged Forward Head Posture

Studies show that FHP has a greater impact on the musculature of the neck and shoulder, causing persistent pain that is often reported in the workplace, as demonstrated by the study [ea07a]. The research involved a study of office workers in ten different Belgian companies, which varied in size from 20 to 120 employees. The study included a sample of 720 individuals who used computers regularly in their work, and these individuals represented a range of occupational categories, including management/administration, medical secretary, graphic design, engineering, and academic faculty. The study assessed self-reported neck pain experienced by participants over the preceding 12 months, with neck pain being defined as pain in the head and neck region. Among 512 respondents, 45.5% of the population reported experiencing neck pain during the past year, with 18.1% reporting con-

tinuous pain. Of those who reported neck pain, 64.3% believed their current job contributed to their discomfort, and 56.2% reported that their neck pain began during their current job. About 10.2% of respondents took sick leave due to neck pain, and 24% reported adaptations to their workplace or equipment to alleviate their neck pain. The study found that women were almost twice as likely to experience neck pain compared to men. Additionally, there were significant differences in the likelihood of persistent neck pain across different age categories, with the highest incidence of neck pain among individuals aged 40-49. Younger subjects reported significantly less neck pain than older participants. Finally, physical inactivity increased the risk of neck pain.

Frequently maintaining a forward bent neck posture for a prolonged period and consistently working in the same position for an extended time are significantly associated with neck pain. [ea01] observed a trend indicating a positive relation between neck flexion and neck pain, implying a higher risk of neck pain for individuals who spend a considerable amount of time working with their necks bent at a minimum of 20 degrees. Neck pain is often linked to repetitive movements performed at a consistent rate. When working with hands and fingers, the neck and shoulder muscles act as stabilizers. To maintain the recommended posture, which is necessary when using a keyboard, the trapezius and other shoulder muscles require static contraction. This contraction is accentuated when the neck is rotated or bent, and the computer screen is positioned to the side of the worker.

Sitting posture has a significant positive relationship with neck pain. According to [ea01], workers who sit for more than 95% of their working time have twice the risk of neck pain as workers who rarely sit. Skov's study [ea96] suggests that the odds ratios for neck pain increase with the duration of sitting, indicating a clear connection between sitting posture and neck pain. [ea91a] reported an odds ratio of 1.49 for the relationship between sitting for more than 5 hours a day and self-reported neck pain. Remaining seated for extended periods, usually accompanied by spinal curvature, puts pressure on vertebral discs, ligaments, and muscles, increasing the risk of neck pain, according to [ea03a].

2.1.2 Psycho-Social Risk Factors

Stress Stress is linked to pain and disability [ea20b], and perceived stress increases the risk of neck pain [ea20c]. Studies have found that adolescents with neck pain experience more stress symptoms and are more likely to report neck pain when they have permanent or regular feelings of stress [ea20a]. Stress can also alter central pain

processing, making individuals extra-sensitive to pain [ea20d].

Anxiety Anxiety is linked to different types of chronic pain, including neck pain, and adolescents with neck pain have higher levels of trait and state anxiety [ea20a]. Lower pressure pain thresholds are associated with increased anxiety levels, and anxiety exacerbates pain and disability. [ea17] Generalized anxiety disorder and PTSD are more strongly associated with spinal pain than social phobia or panic disorder/agoraphobia, but neck pain is more common in patients with mood disorders than in those with specific anxiety disorders. [ea07b].

Depression Depression and neck pain have a bidirectional relationship, with mood disorders and depression related to chronic neck pain and disabilities [ea07b]. Symptoms of depression are associated with high morbidity in neck pain patients, and depressed mood and major depression are strong psychosocial risk factors for chronic back or neck pain. Mood disorders have higher comorbidity with neck pain than other mental disorders, and adolescents with neck pain have more depressive symptoms than asymptomatic adolescents. Depressive symptoms affect central pain processing and can result in remote hyperalgesia [ea18].

2.2 Where and How is Pain Felt

Neck pain can have different manifestations despite having the same anatomical causes. It can occur due to muscle strains, sprains, and tears, leading to small tears in the connective tissue, inflammation, and trigger points, causing sudden and painful contractions of the neck muscles. Neck-related headaches, described as dull or aching, are caused by muscle tension or spasms in the neck. Facet joint pain arises from acute injury or arthritic degradation and causes deep, sharp, and aching pain that worsens with neck movement. Nerve pain is challenging to describe and can cause sharp or dull pain with burning sensations or pins and needles feeling that may shoot down the arm. 2.3

2.3 Kinesiotherapy as Treatment and Prevention

On a more optimistic note, kinesiotherapy has been found to help patients regain their normal range of motion, reduce pain and discomfort, and improve their overall quality of life. With proper guidance from a trained kinetotherapist, patients with neck issues can experience significant improvements in their symptoms and regain their ability to perform daily activities with ease. The duration of kinesiotherapy

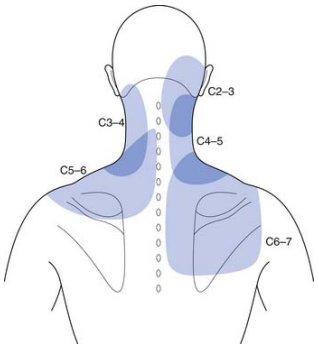


Figure 2.3: Most common pain points, extracted from [ea90]

for neck injuries varies depending on the severity of the injury and the individual needs of the patient. In general, a treatment plan for neck injuries may range from a few weeks to several months, depending on the specific condition and the patient's response to therapy. For example, a mild neck strain may only require a few weeks of kinesiotherapy, while a more serious injury such as a herniated disc may require several months of therapy to achieve significant improvement. One study performed by rehabilitation doctors in Thailand showed that a consistent stretching routine carried out over four weeks can alleviate neck and shoulder discomfort and enhance the overall quality of life for office workers who suffer from persistent and severe neck and shoulder pain. [Tun16]

One of the early pioneers of kinesiotherapy was Dr. Gustav Zander, a Swedish physician who developed a system of exercise machines that were used to help patients recover from various conditions. Zander's machines were designed to target specific muscle groups, and his approach to exercise and movement therapy was grounded in a belief that physical activity could be used to promote health and well-being. Some of his machines can be seen in figure 2.4

While the specific techniques and equipment used in kinesiotherapy have evolved over time, the fundamental principles of using movement and exercise as a form of treatment remain at the core of the discipline to this day.

2.3.1 Most Effective Exercises for Neck Pain

A study published in The Journal of Physical Therapy Science concluded that both deep cervical flexor exercises and general stretching exercises are efficient for improving mobility and posture and alleviating the pain of persons with chronic neck pain [You16]. Over the span of 2 months, the research team examined the Neck Disability Index and neck and shoulder postures, such as head tilt angle, neck flexion angle, and forward shoulder angle, while the patients performed ex-

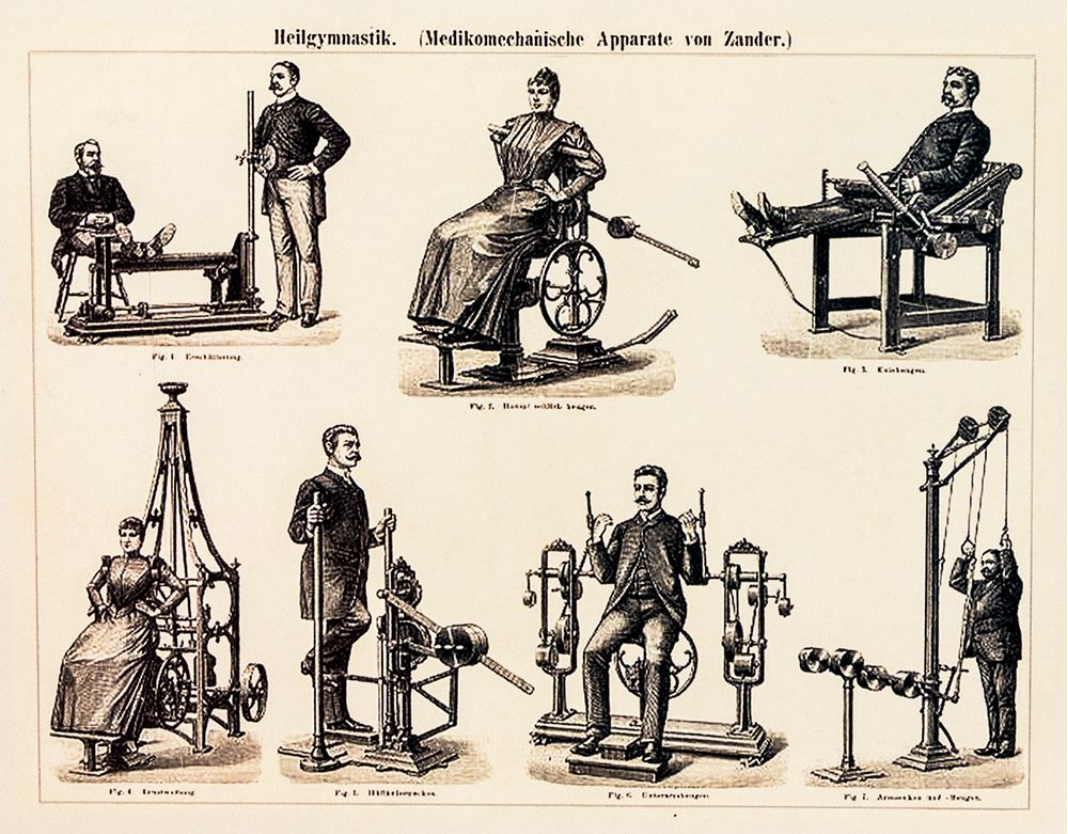


Figure 2.4: The Zander Apparatus

ercises 3 times a week. The participants executed five stretching poses with fixed and straight positions. These poses included stretching the neck, bending the neck forward, bending the neck to each side, and rotating the neck. Each pose was performed 3 to 5 times and held for 10 seconds. The table 2.5 shows the efficacy of these exercises over 2 months, in terms of head mobility.

The general stretching exercises include a combination of flexion, extension and rotation, as seen in figure 2.6, with different duration and frequency.

2.3.2 The Average Range of Motion of the Cervical Spine

Many studies tried to find the average range of motion for each of these types of cervical spine movement (flexion, extension, lateral bending and rotation) and the most comprehensive is [DAP+92]. Using CA 6000 Spine Motion Analyzer, It obtained precise measurements on 150 asymptomatic adults, both males and females, with ages ranging from 20 to over 60, and confirmed clinical impressions that cervical spine motion decreases with age, most dramatically between the decades 30-39th and 40-49th. Moreover, decreased motion occurs sooner and more severely in males, rather than females. It also found that flexion of the cervical spine (mean =

		Baseline	4 weeks	8 weeks
HTA (TA)	DCF	54.6 (1.5)	51.1 (1.5)*	48.4 (0.8)*†
	GSE	54.2 (1.2)	51.7 (1.3)*	51.0 (1.4)
NFA (FA)	DCF	36.6 (1.4)	26.3 (1.6)*	22.8 (2.3)*†
	GSE	35.9 (2.0)	25.7 (1.9)*	24.8 (2.6)
FSA (SA)	DCF	31.3 (1.1)	23.2 (2.0)*	20.7 (1.5)*†
	GSE	30.9 (1.1)	23.5 (2.7)*	23.2 (2.9)

Data are presented as mean ± SD.
*Significantly different from the baseline value ($p < 0.05$).
†Significantly different between the groups ($p < 0.05$).
HTA: head tilt angle; NFA: neck flexion angle; FSA: forward shoulder angle; DCF Group: deep cervical flexor exercise group; GSE group: general strengthening exercise group

Figure 2.5: Comparison of improvements, extracted from [You16].

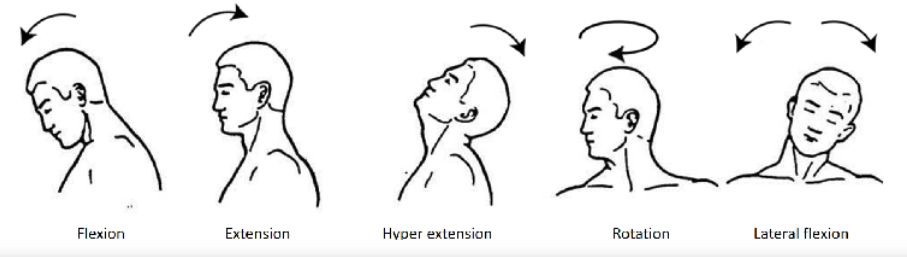


Figure 2.6: Types of cervical movement, extracted from [Yük18]

68.8°) was equivalent to extension (mean = 67.9°). Therefore, the cervical spine is unique in that opposite motions are approximately equal, eg. flexion = extension, right rotation = left rotation and right lateral flexion = left lateral flexion.

Table 2.1 depicts the findings of this study.

Age Decade	Flexion + Extension		Lateral Flexion		Axial rotation	
	Male	Female	Male	Female	Male	Female
20-29	152.7°	149.3°	101°	100°	183.8°	182.4°
30-39	141°	156°	94.7°	106.3°	175°	186°
40-49	131°	140°	83.7°	88°	157.4°	168°
50-59	136.3°	127°	88.3°	76°	166.3°	152°
Over 60	116.3°	133.2°	74.2°	79.6°	145.6°	154.2°

Table 2.1: The average range of motion of the cervical spine, across genders and decades.

Chapter 3

Where Technology Meets Kinesiotherapy

Despite my earlier reference to technology as a source of neck problems, we can reframe the perspective and view it as a potential solution, instead of an enemy. As it has been stated above, kinesiotherapy made use of technology even from its early stages, in the 19th century, in the form of mechanical machines that assist the patients in performing exercises. In 2023, how can we benefit from the current advancements in technology and put them to use in therapy?

One way is by making therapy more accessible to patients through telerehabilitation. Patients are able to access therapy from the comfort of their own homes, reducing the need for travel and making it easier for them to receive the care they need. [MB]

A more interesting example is that of motion analysis systems. Using cameras and sensors to track a patient's movements and provide real-time feedback on their technique and form, they can be particularly useful for patients who are recovering from an injury or surgery and need to relearn proper movement patterns. Motion analysis can be used to evaluate various musculoskeletal disorders affecting both the upper and lower extremities. The analysis can focus on various parts of the body, depending on the patient's needs, such as describing the movement of the hip, knee, and ankle or focusing on the torso, shoulder, elbow, and wrist. By assessing motion in all three planes of movement, namely frontal, longitudinal, and axial, disabilities can be described more accurately than they would be by observation only.

The leading available softwares for movement analysis are Vicon, Optotrak Certus and Cortex.

Vicon Vicon is a system that was created in Oxford, England, and it is the most commonly utilized system in clinical settings. It employs passive markers to capture

and monitor motion and provides regular components that are frequently employed by researchers and clinicians during gait analysis. It employs Nexus software to record motion data and synchronize signals from other measuring devices, such as EMG (electromyography) and force plates. [Vic]

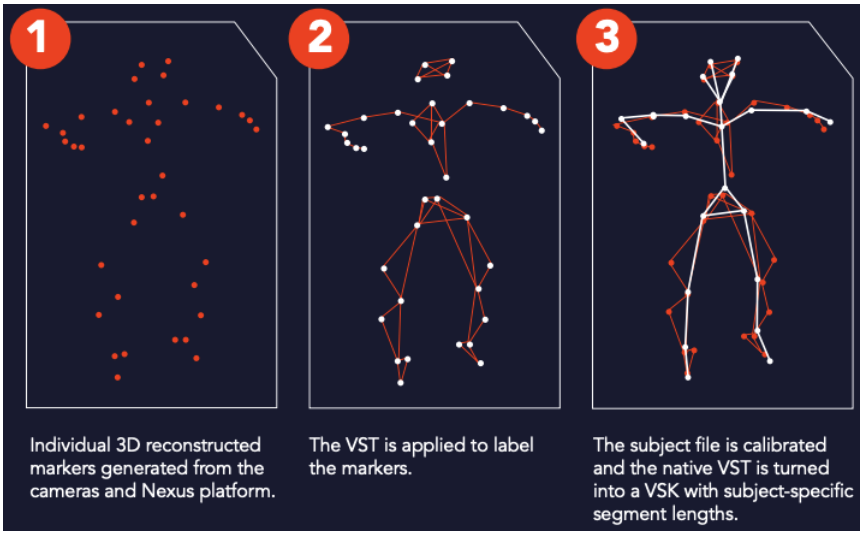


Figure 3.1: How data from sensors placed on the body is mapped to a kinematic skeleton, using Nexus. Picture extracted from one of Vicon's usage guides, available on their website [Vic].

Optotrak The Optotrak Certus, developed by Northern Digital Inc. in Ontario, Canada, is another system that utilizes an active marker system called "Smart Marker". This system eliminates the need for wires, since battery-powered strobes are used instead. Up to 50 strobes can be used at a time for a single battery system. It can integrate force plates, EMG, eye trackers, and other third-party tools.

Cortex Similar to the Vicon system, Cortex, developed in Santa Rosa, CA, uses passive markers and it is utilized throughout all stages of recording, such as calibration, tracking, and post-processing. It can also compute and display kinematic, kinetic, and electromyography data. [Cor22]

3.1 Microsoft Kinect

Microsoft Kinect has revolutionized this field with its first appearance in 2010. It's an input device designed for Xbox 360, Xbox One, and Windows PCs that detects motion, using webcams to allow users to interact with their console without a controller. The sensors feature 3D depth cameras that let people interact with games using their bodies in a natural way. One of its unique characteristics is that

the games are played without controllers because the sensors monitor whole limb movements. The Kinect's impact has extended far beyond the gaming industry, with applications in fields like assisting doctors in operating rooms, helping children with autism, and aiding in the rehabilitation of neurological patients, as it is depicted in figure 3.2.

Figure 3.2: The usage of Microsoft Kinect for at-home rehabilitation exercises, extracted from the Official Microsoft Blog

There are many companies and organizations out there that have utilized the unique technology of Microsoft Kinect to create rehabilitation-based software programs. But in this discussion, I want to focus on EvolvRehab and MIRA Rehab, with the latter having been developed by a team of students at our university.

3.1.1 EvolvRehab

EvolvRehab is a physical rehabilitation system that uses motion sensors and video game technology to provide rehabilitation services, and it has been clinically validated. This software is the first virtual rehabilitation tool that has been designated as a medical device and has obtained the CE mark of approval under the EU's Medical Device Directives. This software allows patients with various functional limitations to participate in innovative and engaging rehabilitation exercises, targeting multiple pathologies such as neurodegenerative diseases, neuromuscular disorders, neurovascular disorders/trauma, and mobility issues in the elderly. Functional training is provided to improve balance, coordination, weakness, fatigue, and spasticity, and the exercises can be customized to suit the patient's disability level. By utilizing the unique features of Microsoft Kinect's motion technology, EvolvRe-

hab tracks and records the patient's movements, immersing them in a 3D environment where they can interact with the game. [Reha]

3.1.2 MIRA Rehab

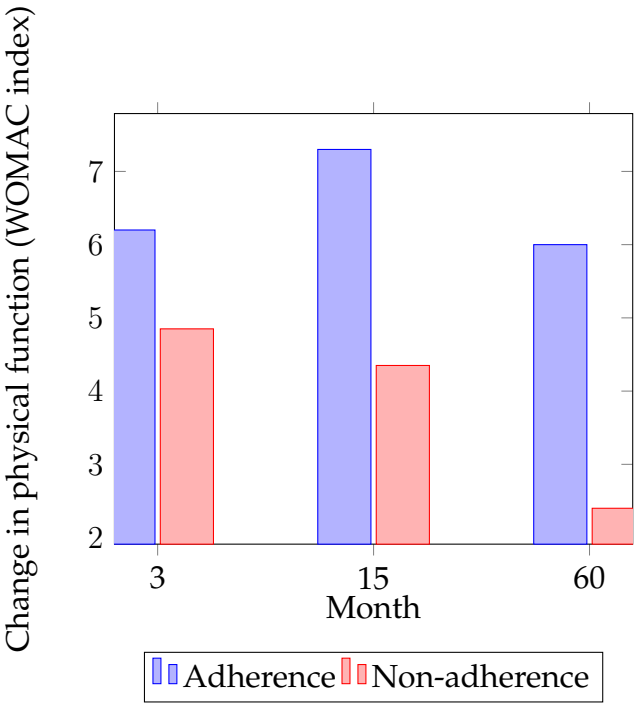
Another similar project that has been adopted at an international scale is MIRA Rehab, which intends to assist injured patients in performing their recovery exercises, targeted toward the upper limbs. It does so by means of interactive games, that use the in-depth camera of Kinect to monitor user movement, provide real-time feedback, gather statistics and also make the information of each patient's progress available for their doctor. [Rehb]. A study performed on a male, 81 years old patient, showed that consistent use of the system integrated into the rehabilitation process generated improvements over several metrics and the gamification of the therapeutic process made it more enjoyable. [MTU+17]

3.2 The Accountability Issue

A major problem that is solved by these assisted rehabilitation systems is that of patients' accountability. It seems that people have a hard time adhering to and staying consistent with a home exercise program, also known as HEP, leading to inefficient, delayed or never-achieved recovery. As of [AR18], the rate of noncompliance could be as high as 50 percent and this is due to a plethora of factors, both psychological and situational. Individuals may not find the time to fit the exercise routine into their daily schedule or even forget to do them at all. In some cases, when the progress from one session to another is so small that the patients do not notice it, the incapacity to measure their exact progress may lead them to believe that they are not making any progress at all, which builds up disappointment and may cause them to give up, eventually. Another psychology-related motive is the external locus of control. When a patient unconsciously believes that the forces that influence the outcome of his life are external to his own person, he may not see the reason why he should engage in recovery exercises, because, ultimately, he doesn't have much control over the results of the therapy.

Not only that noncompliance with the recommendations of the therapist during and after the treatment period can slow down the recovery process, but it can also cause a relapse of the symptoms, meaning a deterioration of the state of health after a temporary period of improvement. The probability of a recurrence of persistent pain after an initially successful treatment seems to be considerable, with estimates ranging between 30% and 60%. Findings from research conducted on populations

with arthritis and mixed pain clinic attendees propose that there may be a correlation between relapse and non-adherence to treatment [ea91b]. The graph below, extracted from [ea10], depicts the difference in the average change in physical function over a period of time between patients who adhere to a treatment plan and those who do not. The measurement used for physical function is the Western Ontario and McMaster Universities Osteoarthritis Index, abbreviated as WOMAC.



If designed well, technological systems can be used as tools that support both the patient and the clinician in the therapeutic process and overcome this problem of non-adherence to at-home treatment plans. Below are some features that can be implemented by software, that have been proven [AR18] to reinforce the healing process.

In-app coaching An element that lies at the base of the clinician-patient relationship is coaching. By incorporating supervision, feedback, and reinforcement strategies into the design of connected health interventions, patients can receive coaching similar to that offered in clinics by their healthcare professionals in their homes. When physiotherapists provide positive feedback and monitor exercise performance and symptom progression, adherence rates increase. Remote monitoring via cloud-based portals can offer supervision, while remote communication through video calling, instant messaging, or email can provide additional coaching components.

Goal setting Setting objectives encourages patients to adopt a more proactive attitude toward their recovery. In a study involving a younger athletic population,

setting goals with the help of a psychologist and incorporating other adherence-improving interventions alongside clinic-based rehabilitation led to significant differences in adherence [LA08].

Self-monitoring The last feature that can have a positive impact on the engagement of patients, if implemented, is that self-monitoring. By providing visual feedback on their physical activity, users may feel more motivated to continue doing their prescribed routine, because there is objective and reliable data that shows them that their effort pays off. Visual feedback using activity monitors has been shown to improve adherence to physical activity and exercise frequency, according to [ea14]. However, the study was not specific to the musculoskeletal population and targeted general physical activity rather than targeted home exercises. Another study, [ea03b] found that using an accelerometer for self-monitoring as part of an arthritis self-management program led to a significant increase in overall physical activity.

Chapter 4

Specifications of the Proposed Application

Given the problem described in the previous chapters, the solution that I propose as a tool for the treatment and prevention of chronic neck pain is HeadFlow, an iOS app. Essentially, while the user performs a sequence of stretching exercises, the app will detect his head motion, using the sensors built into AirPods Generation 3, AirPods Pro, AirPods Max or Beats Fit Pro, and track his progress over time. He can also connect to a therapist, that will provide feedback and tailor his stretching sequence for his own needs.

4.1 Use Cases

Authentication & User Management

- The user can register as a patient by filling in his first name, last name, email and phone number. If he wants to register as a therapist, he also has to insert his Medical License ID. His phone number will be verified through a one-time password (OTP) received through SMS.
- To log in, the user will insert his phone number and will receive an OTP through SMS. The code can be resent after 30 seconds.
- The user can also log in using a Social Sign-Up method, Google.
- Once logged in, the user will remain so from one launch of the app to another, until he decides to log out.
- The user does not have access to the app if he is not logged in.

The app also offers therapists a method to create an account, using their Medical License ID.

As a patient, perform the stretching sequence

- The user will have a default sequence of stretches that he will have to perform. In the beginning, the default duration of each exercise is 5 seconds. From one session of stretching to another, the duration of each stretch will be increased by 1 second.
- The user can pause or resume the execution of the stretch.
- Before he starts executing the sequence, a countdown screen is shown.
- While the user performs the current stretching exercise, a visual representation of his move will be displayed dynamically on the screen.
- At the end of the session, the user will be presented with a screen with the summary of his stretching session, including duration and average range of motion achieved.
- If the AirPods are disconnected, an alert message is shown and the timer is paused. The alert only disappears when the user connects his headphones or abandons the stretching session.

As a patient, see the stretching history

- The user can access a list of all his stretching sessions, ordered by most recent.
- A graph with the average ranges of motion by time and date will be displayed at the top of this list.
- If he taps on one of the sessions, a bottom sheet will appear. This bottom sheet will display the range of motion achieved for every stretch that was executed, with a visual comparison of the achieved mobility between the left side of the neck and the right side.
- The user will have the option to export this report as a PDF and share it through Email, WhatsApp or AirDrop.

As a therapist, connect to a given patient

- The therapist can search for his patient and send him an invitation to connect.
- The therapist can see a list of all his connected patients and filter them by name.
- Once the patient accepted the invitation, the therapist can see his stretching history and provide feedback for each session.

- The therapist can see different types of reports and graphs with progress over time.
- The therapist can change the duration of each stretch and modify the default set of stretches for each of his patients, according to their own needs.

As a patient, collaborate with a therapist

- The patient will receive a notification if the therapist sent him a connection request. He can either accept or decline it.
- If he accepts the invitation, he will receive a notification every time the therapist provides feedback for one of his sessions.
- At any time, he can choose to interrupt the collaboration.

Local notifications

- Each morning, at 9 AM, the patient will receive a notification reminding him to perform his stretching session.
- If he opens the notification, it will redirect him to the stretch player.

A compact, visual representation of the application requirements is depicted in figure 4.1.

4.2 Model

User A user has the following data: first name, last name, email address, phone number, profile picture, type (patient or therapist) and, if he is a therapist, Medical License ID.

Collaboration A collaboration links a therapist with a patient and has a status (pending, active, declined).

Stretching exercise A stretching exercise is defined with a type and a duration. A stretching exercise type can be one of the following and has an associated maximum range of motion, expressed in degrees. According to [DAP+92], the maximum range of motion for each type of these exercises differs in dependence on age and gender, so the therapist will set the values for each patient, in line with his needs. However, if the patient does not collaborate with a therapist, the default mean ranges for each exercise are the following:

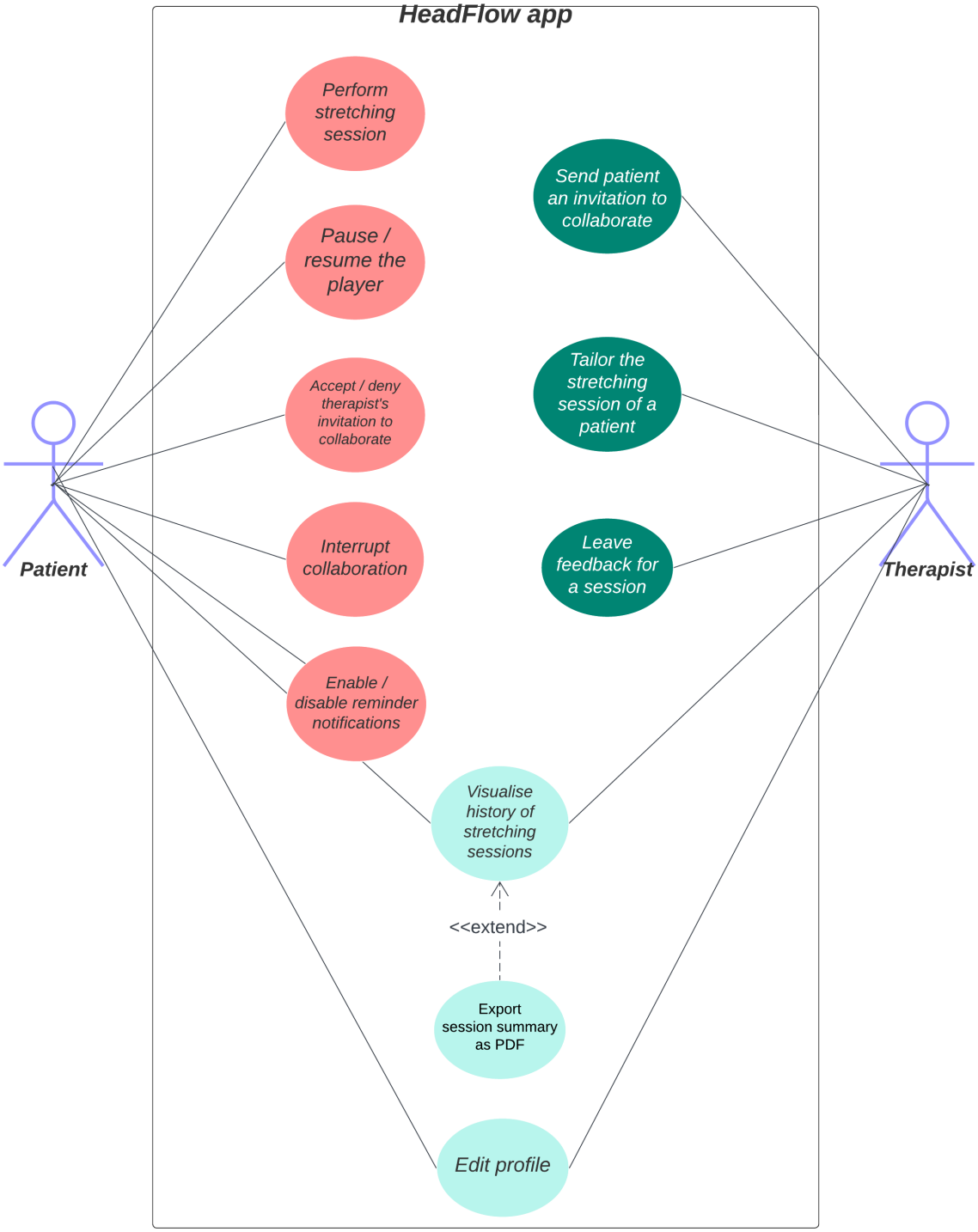


Figure 4.1: The use case diagram of the software system

- flexion (forward bend) - 45°
- extension (backward bend) - 45°
- lateral flexion to right - 45°
- lateral flexion to left - 45°
- axial rotation to right - 60°
- axial rotation to left - 60°

Stretching Session A stretching session is composed of a map between the set of stretching exercises and the range of motion that was achieved for each exercise, in percentage, the date, the total duration, and the total average range of motion.

Chapter 5

Implementation Details

5.1 System Design & Architecture

The architecture diagram 5.1 below shows the components of the software system, consisting of an iOS client app, a Node.js server, and a MongoDB database. The client app communicates with the server using HTTPS, and the server communicates with the database using the Mongoose library. The server is deployed on Render and also uses the Courier SDK to send one-time passwords (OTPs) via HTTPS, while the database is stored in a Mongo Atlas cluster.

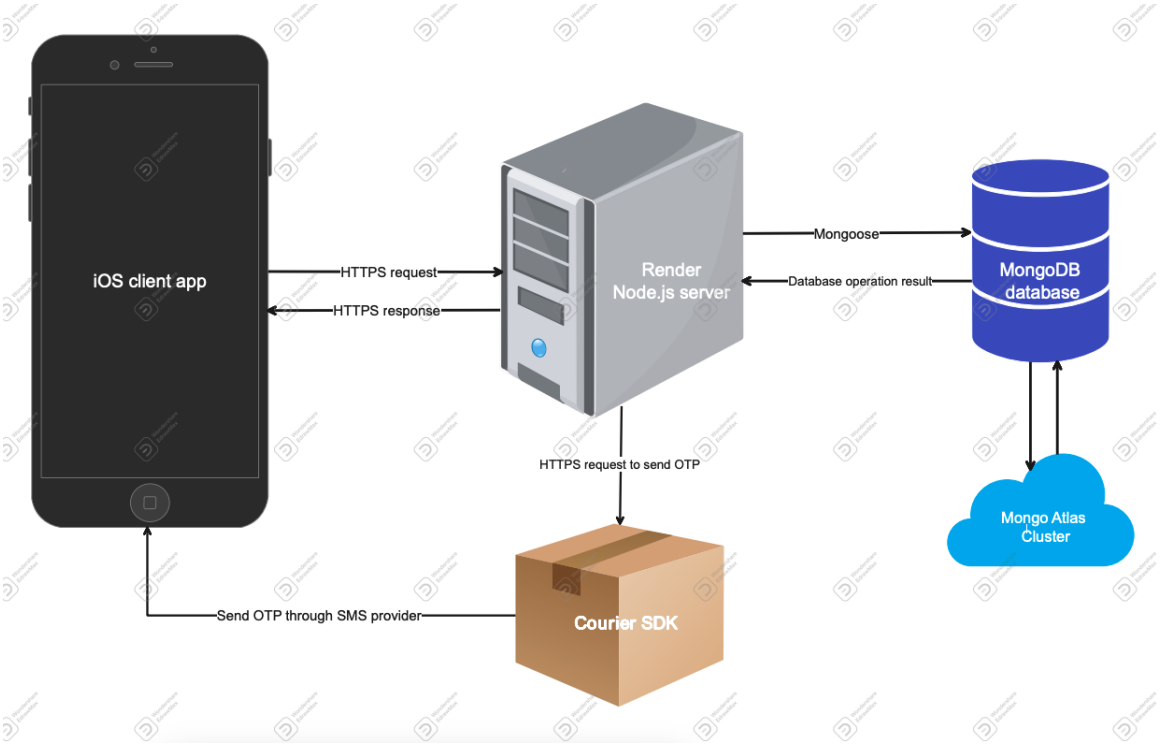


Figure 5.1: The architecture diagram of the software system

5.1.1 SwiftUI and UIKit iOS Client

The iOS app is written in Swift and is built using both UIKit and SwiftUI, the consecrated UI frameworks from Apple. The 'skeleton' is built on a UIKit foundation, that handles and controls the navigation throughout the app, which is achieved using `UINavigationController`. The actual views are built in SwiftUI, and they are 'placed' on the navigation skeleton using `UIHostingController`.

MVVM - architectural pattern In order to achieve **separation of concerns**, for the architecture I used the MVVM pattern, which stands for Model-View-View Model and it is an enhancement to the MVC (Model-View-Controller) pattern. The **Model** is used to encapsulate data and it is not aware of any other component. The **View** has a limited set of duties, namely to display data to the user and receive their actions. It is only aware of the ViewModel, which is a middleman between the Model and the View. As a result, the View should not contain any business logic. The **View Model** serves as an intermediary layer between the Model and the Controller. It has control over the Model and handles data manipulation in a manner that prepares it for presentation by the View. The **Controller** remains responsible for configuring and managing its owned components. However, its role is less complex compared to the traditional MVC since the tasks related to modifying the Model's data are now handled by the View Model layer. Additionally, the Controller now owns the View Model rather than the Model directly. A visual representation of this architecture is depicted in the below diagram 5.2.

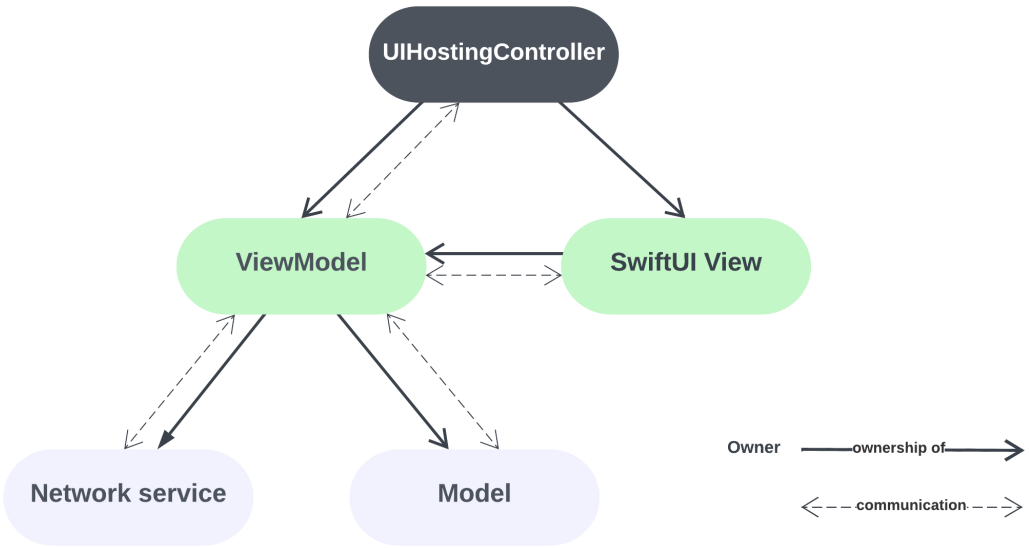


Figure 5.2: The architecture diagram of the iOS client app

Coordinator - behavioral pattern Coordinator, also known as Mediator in other frameworks, is a popular design pattern used in iOS apps, as it ensures **low coupling** and **high cohesion** between view controllers. The responsibility of the coordinator is to encapsulate navigation logic, such that view controllers are isolated and invisible to each other and can be reused easily and independently. Usually, there is a Root Coordinator, which keeps references to child coordinators, each corresponding to another flow inside the app. Upon starting a new flow, the child coordinator is instantiated with an `onFinished()` callback, which it calls at the end of the flow. In this way, the child coordinator is devoid of the responsibility of what will happen next, allowing greater flexibility and scalability of the component.

In my application, there are 3 major navigation flows: Authentication, Main flow for a Patient and Main flow for a Therapist. At the launch of the app, the Root Coordinator checks if the session has a valid access token. If it does, it launches either the Main Coordinator for a patient or the Main Coordinator for a therapist, depending on the type of user that is logged in. Each of the main coordinators receives a callback to start the Authentication Coordinator when the user logs out. If no user is logged in at the launch of the app (valid token in Session), the Root launches the Authentication Coordinator, with a callback to start the appropriate Main Coordinator after the user is logged in.

```
class RootCoordinator: Coordinator {

func start(connectionOptions: UIScene.ConnectionOptions?) {
    if let currentUser = Session.shared.currentUser {
        switch currentUser.type {
        case .patient:
            showPatientMainCoordinator()
        case .therapist:
            showTherapistMainCoordinator()
        }
    } else {
        showAuthenticationCoordinator()
    }
}

func showAuthenticationCoordinator() {
    let coordinator = AuthenticationCoordinator(
    window: window,
    dependencies: dependencyContainer,
    onEndAuthenticationFlow: { [weak self] userType in
        switch userType {
        case .patient:
            self?.showPatientMainCoordinator()
        case .therapist:
            self?.showTherapistMainCoordinator()
        }
    })
    self.authenticationCoordinator = coordinator
    coordinator.start(connectionOptions: nil)
}

func showPatientMainCoordinator() {
    let coordinator = PatientMainCoordinator(
    window: window,
    dependencies: dependencyContainer,
    onLogout: { [weak self] in
            Session.shared.close()
            self?.showAuthenticationCoordinator()
        })
    self.patientMainCoordinator = coordinator
    coordinator.start(connectionOptions: nil)
}

func showTherapistMainCoordinator() {
    let coordinator = TherapistMainCoordinator(
    window: window,
    dependencies: dependencyContainer,
    onLogout: { [weak self] in
        Session.shared.close()
        self?.showAuthenticationCoordinator()
    })
    self.therapistMainCoordinator = coordinator
    coordinator.start(connectionOptions: nil)
}
}
```

Singleton - creational pattern The goal of the singleton pattern is to ensure that only one instance of a class is instantiated and provide global access to that instance throughout the whole program. To implement Singleton, the initializer must be private, such that it cannot be called from outside, preventing the class from being

instantiated multiple times by accident. Secondly, the class must have a public static property that is instantiated using the private constructor, which will serve as the sole global access point to that class. In my iOS app, I used Singleton for the class Session, whose responsibility is to handle the session of the logged user. It saves the authorization token and the current user in `UserDefaults`, such that this data remains "alive" from one launch of the app to another. The singleton pattern ensures that this data is consistent throughout the whole application, preventing 2 instances from existing and holding the data of 2 different users at the same time. The implementation of Session is depicted in the screenshot below.

```
class Session {

@UserDefault(key: StorageKeys.accessToken,
             defaultValue: nil)
var accessToken: String?

@UserDefault(key: StorageKeys.currentUser,
             defaultValue: nil)
var currentUser: User?

@UserDefault(
    key: StorageKeys.notificationsEnabled,
    defaultValue: nil)
var notificationsEnabled: Bool?

@UserDefault(
    key: StorageKeys.notificationFromTherapist,
    defaultValue: false)
var hasNotificationFromTherapist: Bool

static let shared: Session = Session()

var isValid: Bool {
    return accessToken?.isEmpty == false
}
func saveCurrentUser(user: User,
                     token: String) {
    currentUser = user
    accessToken = token
}

func close(error: Error? = nil) {
    guard isValid else { return }
    removeSessionData()
}

func removeSessionData() {
    accessToken = nil
    currentUser = nil
    hasNotificationFromTherapist = false
}
}
```

5.1.2 Node.js Server

The Node.js server is implemented using the Express framework. It listens for incoming HTTP requests at specific routes, and for each of these routes, there is a corresponding controller that handles the request. These controllers are responsible for calling different functions based on the type of request (GET, POST, PUT DELETE), and they also have references to various services that allow them to access and manipulate data stored in Mongoose models.

In this architecture, the controllers act as an intermediary layer between the server and the services. They receive the incoming requests, parse the data, and then pass it on to the appropriate service for further processing. The services themselves are responsible for interacting with the database and performing any necessary operations on the data. This separation of concerns makes the code more modular and easier to maintain, as each component has a clear and distinct role to play in the overall system. 5.3

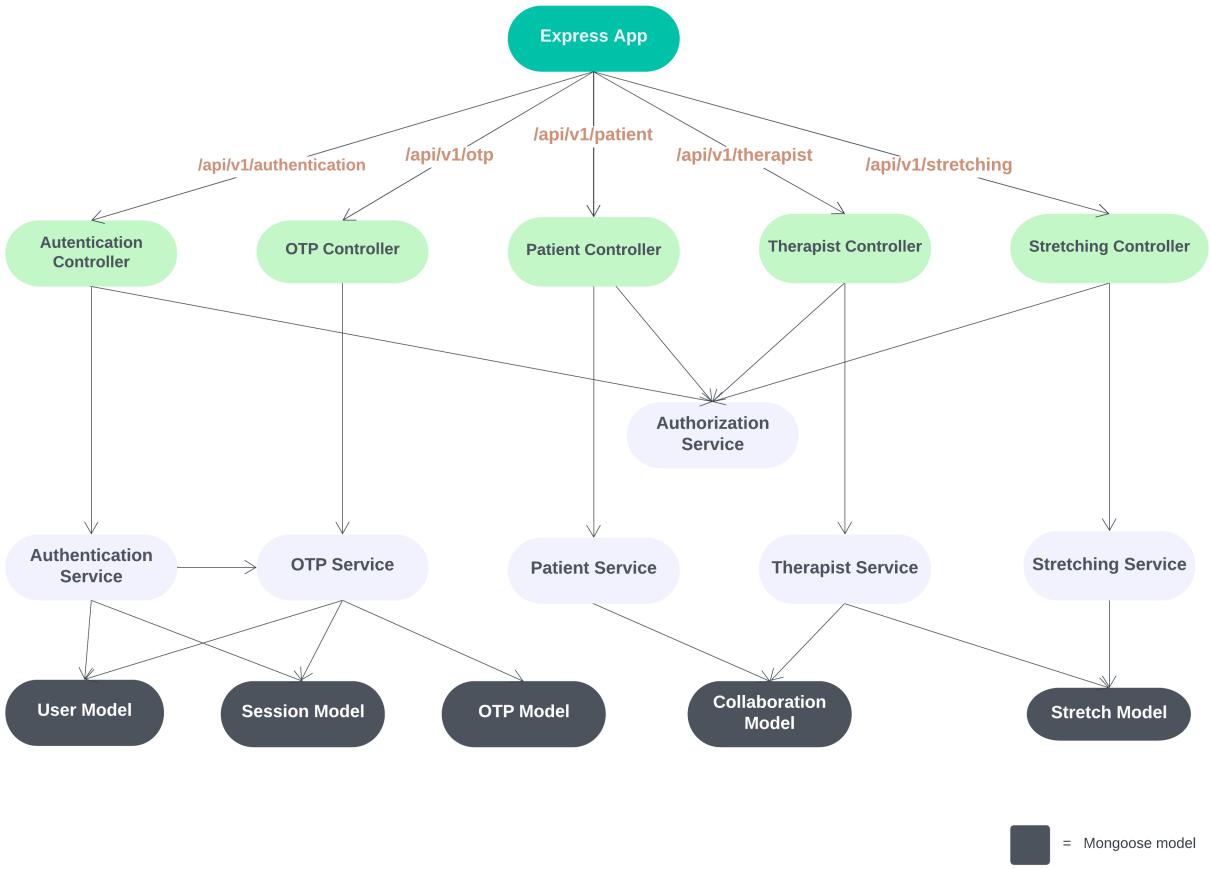


Figure 5.3: The architecture of the Node.js server

Render In order to ensure that my server is always up and running, even in periods of high traffic, I deployed it on Render, which is a cloud computing platform. I connected the Git repository which contains the Node server to the Render web service, I specified the Git branch it should track, the environment variables, the build and start command and the output was the root path of the API, which is `https://headflow.onrender.com`. The check the "health" of the server, one can access the path `https://headflow.onrender.com/healthcheck`. 5.4

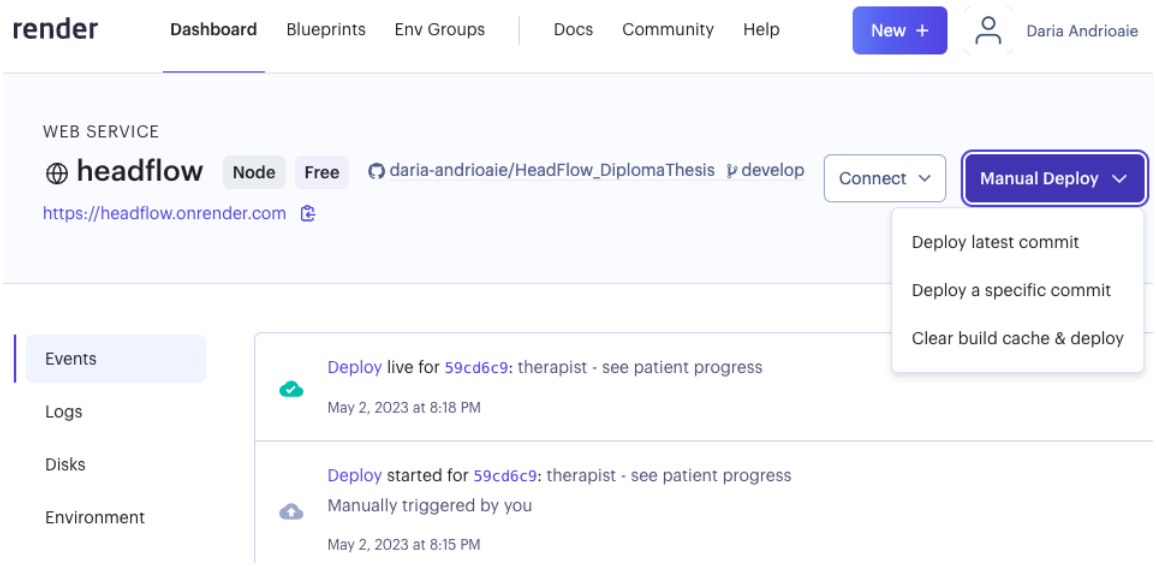


Figure 5.4: The Render console

5.1.3 Mongo DB

For the database, I opted for MongoDB, because its document-oriented data model allows to store and query data in a flexible and scalable way, which made it perfect for my changing and evolving data requirements. To interact with the database, I used **Mongoose**, a popular Object Data Modeling (ODM) library for Node.js. A Mongoose schema defines the properties of a data model, such as data types, validation rules, and default values, ensuring that data is consistent and well-structured. Once the schema is defined, a Mongoose model is created, which provides a programmatic interface for querying and manipulating the data in the actual database. Below is an example of my schema for an OTP document and the way it can be saved and updated, from the `OTPService`.

```
const mongoose = require("mongoose");

const otpSchema = new mongoose.Schema(
  {
    otp: {
      type: String,
      required: true,
      trim: true,
    },
    expireIn: {
      type: String,
      required: true,
      trim: true,
    },
    phoneNumber: {
      type: String,
      required: true,
      trim: true,
    },
    status: {
      type: String,
      required: true,
      trim: true,
    },
  },
  {
    timestamps: true,
  }
);

module.exports = mongoose.model("otp", otpSchema);
```

```
const addNewOTP = (otp,
                   expirationTime,
                   phoneNumber,
                   status) => {
  const otpModel = new OTPModel({
    otp: otp,
    expireIn: addMinutesToDate(new Date(),
                               expirationTime),
    phoneNumber: phoneNumber,
    status: status,
  });
  return otpModel.save();
};

const rejectPendingOTP = (phoneNumber) => {
  return OTPModel.updateMany(
    { phoneNumber: phoneNumber,
    status: "PENDING" },
    { $set: { status: "REJECTED" } }
  );
};
```

I chose to host the database in a cluster in MongoAtlas, which is a fully-managed cloud database service that makes it easy to set up, operate, and scale MongoDB deployments. Once I set up the cluster, I received a connection URL, and inside the server, I called `mongoose.connect(DB\_URL)`. 5.5

5.2 Authentication Flow

The entry point inside the app is represented by the authentication flow, which consists of the Register feature and the Login feature.

To register, the user must fill in his first name, last name, email, phone number and, if he is a therapist, he must provide his medical license ID. After that, his phone number will be validated by entering the one-time password received through SMS. To log in, he only has to provide his phone number, after which he will enter to OTP received as SMS. When he registers, his data will be saved in the database with the status `PENDING`, and after he enters the OTP, his status will be changed to `CONFIRMED`.

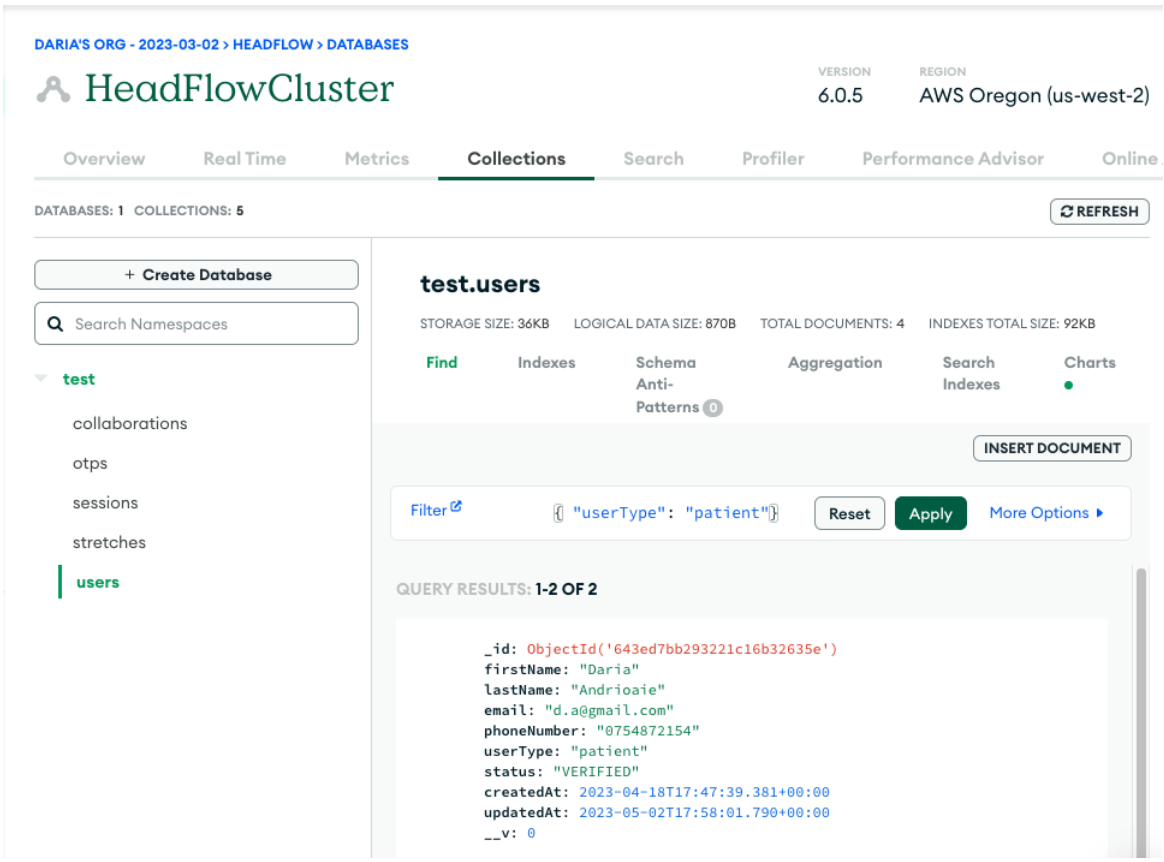


Figure 5.5: Querying the collection of users in the MongoAtlas cluster

5.2.1 One-Time-Password Sign In

To spare the user from the hassle of remembering or resetting his password, I chose to secure his account through one-time passwords received as SMS on his phone number. This way, when he registers on logs back in, he must only provide his phone number and he will receive his ”key” to access the app right away. The solution I chose for this feature is based on Courier, a popular and lightweight service used to send notifications to users through channels such as email, SMS, push and chat. The actual SMS service that it uses under the hood is Twillio, for which I created a free trial account with a phone number, and I received an API key to use on the server side and make the request to send an SMS to a specific phone number. To generate a unique, 4 digits code for each SMS, I used the package `otp-generator` in the Node server.

The Mongoose schema for the OTP contains the following fields: code, phoneNumber, expireIn, status. When the code is sent, it is saved in the database with the status `PENDING`. When it is entered by the user and verified, its status changes to `CONFIRMED`. If, for unknown reasons, the user does not receive the code in 30 seconds and chooses to resend the code, the status of the initial code is changed to `REJECTED`. This logic is implemented in the `OTPService` on the server side, and it

is shown below.

```
const courier = CourierClient({
  authorizationToken: process.env.COURIER_TOKEN,
});

const sendOTP = async (firstName, phoneNumber) => {
  const otp = otpGenerator.generate(4, {
    lowerCaseAlphabets: false,
    upperCaseAlphabets: false,
    specialChars: false,
  });

  await addNewOTP(otp, 15, phoneNumber, "PENDING");
  await sendVerificationMessage(
    {
      firstName,
      otp,
    },
    phoneNumber
  );
  return {
    success: true,
    message: "OTP sent successfully.",
  };
};

const sendVerificationMessage = (params, phoneNumber) => {
  return courier.send({
    message: {
      to: {
        data: params,
        phone_number: "+40767998715",
      },
      content: {
        title: "HeadFlow Verification",
        body: "Hi {{firstName}},
        \nYour verification code for HeadFlow is {{otp}}.",
      },
      routing: {
        method: "single",
        channels: ["sms"],
      },
    },
  });
};
```

The figure 5.6 shows the sequence diagram for the Register Patient use case.

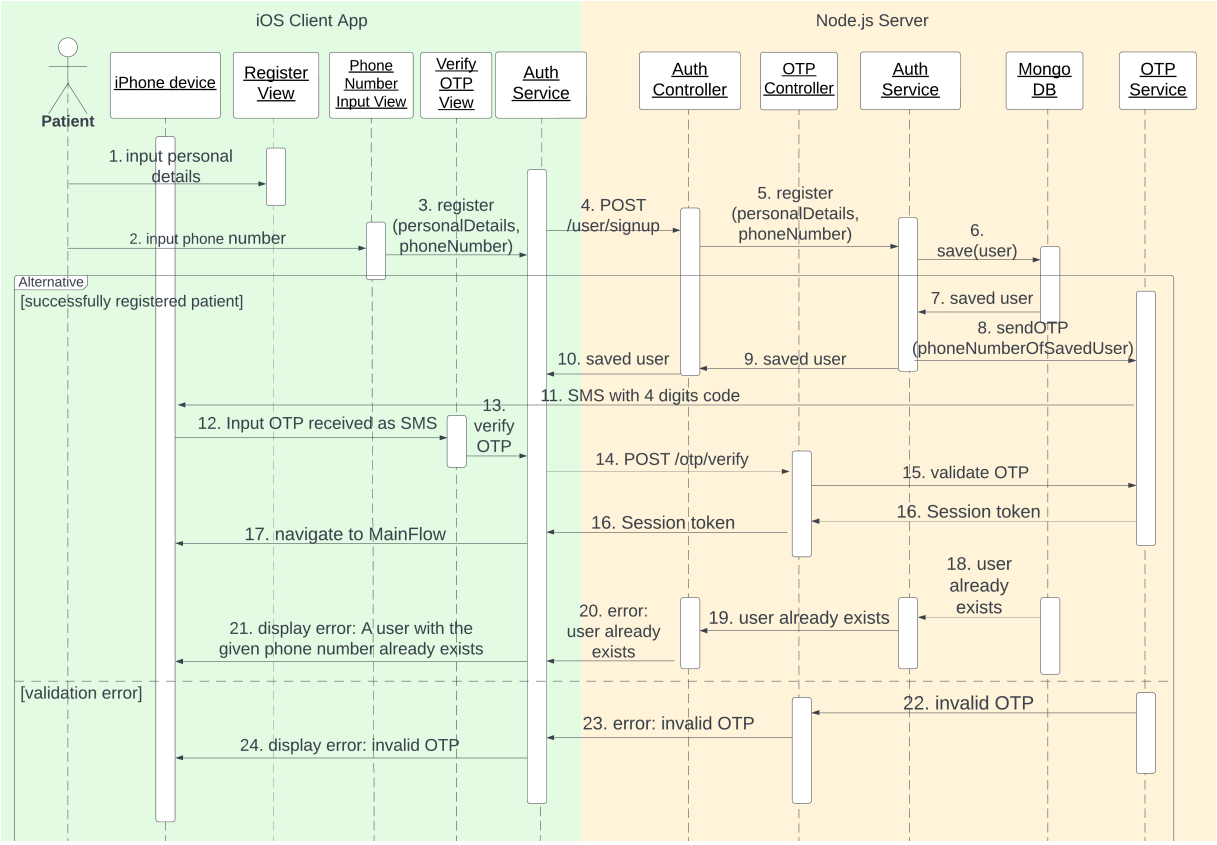


Figure 5.6: The sequence diagram for the Register feature

5.2.2 Social SignIn

The user also has the option to log in using his Google account, and his personal details will be extracted from the associated account.

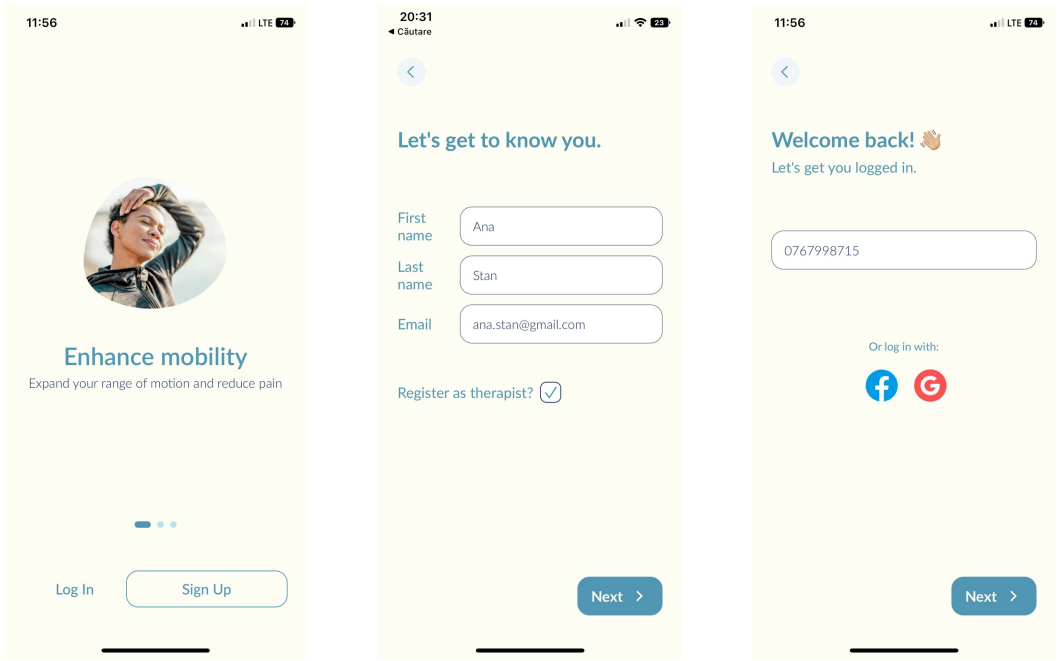
Google To achieve authentication with Google, I had to create credentials on the Google Cloud platform, both for the OAuth client (the iOS app) and the server client. Below is the function that is called when the Google icon is tapped, triggering the sign-in. The token returned after the completion of the sign-in is sent to the server, where it is decoded and a new user is created with the extracted data and saved in the database, if he wasn't registered already.

```
func signupWithGoogle() {
    guard let presentationController = presentationController else { return }
    let signInConfiguration = GIDConfiguration(clientID: Constants.GOOGLE_CLIENT_ID,
    serverClientID: Constants.GOOGLE_SERVER_CLIENT_ID)
    GIDSignIn.sharedInstance.configuration = signInConfiguration
    GIDSignIn.sharedInstance.signIn(withPresenting: presentationController)
    { [weak self] signInResult, error in
        guard error == nil else {
            print(Errors.CustomError(error.debugDescription))
            return
        }
        guard let result = signInResult else {
            self?.apiError = Errors.CustomError("There was an error signing in with Google.")
```

```
                return
            }
        guard let idToken = result.user.idToken?.tokenString else {
            self?.apiError = Errors.CustomError("There was an error signing in with Google.")
            return
        }
        self?.sendSocialSignInToken(idToken)
    }
}
```

Session Management In order to authorize the user to perform certain requests, after the verification of the OTP, the id of the user is encoded into a session token, generated using the package `jsonwebtoken` in the node.js server, which is then sent to the client app as the response of the request. The client app then saves the token locally, in the cache, so that it is still "alive" from one launch of the application to another, keeping the user logged in, and it is available for other requests that need authorization. When the user logs out, the token is discarded from the local cache and invalidated on the server side.

Figure 5.7 depicts the authentication flow in execution.



(a) Onboarding (b) Register (c) Login

Figure 5.7: Authentication flow (1)

5.3 Stretching flow

When a patient wants to perform the stretching session, the proper sequence is fetched from the server, with a `GET` request. If the patient has an active collabo-

ration with a therapist who tailored a personalized sequence, then he will have to perform that one. Otherwise, he will perform the default session, which includes all possible exercises, with a duration of 5 seconds each. Once the exercises are available, a countdown screen is presented and the patient also has the possibility to abandon the session. Once the session has started, the current exercise, a timeline and a timer with the remaining duration for the current exercise are displayed on the screen. 5.8

(a) Tilt left (b) Tilt backwards (c) Rotate right

Figure 5.8: Stretching Executor

5.3.1 CMHeadphoneMotionManager Library

To detect head motion in real-time, the AirPods Pro use a combination of sensors, that yield data such as orientation and rotation rate. Specifically, two types of sensors are used: optical sensors and motion sensors.

The optical sensors consist of inward-facing and outward-facing sensors that detect whether the AirPods Pro are in the user's ears or not. These sensors also help the AirPods Pro detect speaking, so they can activate their noise-cancellation feature.

Each AirPod is equipped with a built-in accelerometer and gyroscope, which are responsible for capturing data related to the user's head motion. The accelerometer measures linear acceleration, while the gyroscope measures angular velocity. By combining the data from these sensors, the AirPods can accurately determine the orientation and movement of the user's head.

When it comes to integrating AirPods head motion detection into a SwiftUI app, the `CMHeadphoneMotionManager` framework plays a crucial role.

`CMHeadphoneMotionManager` is part of the `CoreMotion` framework provided by Apple, which allows developers to access and utilize motion data from various sensors, including the motion sensors embedded in AirPods.

To create an instance of `CMHeadphoneMotionManager`, CoreMotion must be imported. By calling `startDeviceMotionUpdates(to:withHandler:)`, the app begins receiving motion updates from the AirPods. The withHandler parameter takes a closure that will be called each time new motion data is available. In this closure, the necessary information can be extracted from the `CMDeviceMotion` object provided, such as attitude (head orientation), rotation rate (angular velocity), gravity and user acceleration, depending on each type of stretching exercise.

```
class MotionManager: NSObject, ObservableObject {

private var motionManager = CMHeadphoneMotionManager()
@Published var motion: CMDeviceMotion?
@Published var airpodsAreDisconnected: Bool = false

override init() {
    super.init()
    checkConnection()
    startMotionUpdates()
}

private func checkConnection() {
    if !motionManager.isDeviceMotionActive {
        airpodsAreDisconnected = true
    }
}

private func startMotionUpdates() {
    motionManager.delegate = self

    motionManager.startDeviceMotionUpdates(
    to: OperationQueue()) {
        [weak self] motion, error in
        guard let self = self,
        let motion = motion else {
            return
        }

        DispatchQueue.main.async {
            [weak self] in
                self?.motion = motion
        }
    }
}

deinit {
    motionManager.stopDeviceMotionUpdates()
}

func degrees(_ radians: Double) -> Double {
    return 180 / .pi * radians
}

extension MotionManager:
    CMHeadphoneMotionManagerDelegate {

    func headphoneMotionManagerDidConnect(
    _ manager: CMHeadphoneMotionManager) {
        DispatchQueue.main.async { [weak self] in
            self?.airpodsAreDisconnected = false
        }
    }

    func headphoneMotionManagerDidDisconnect(
    _ manager: CMHeadphoneMotionManager) {
        DispatchQueue.main.async { [weak self] in
            self?.airpodsAreDisconnected = true
        }
    }
}
```

CMAttitude The `CMAttitude` object available in `CMDeviceMotion` provides information about the orientation of a device or user's head in three-dimensional space. It encapsulates the device's attitude as a combination of pitch, roll, and yaw.

Pitch represents the rotation around the device's lateral axis, which is an imaginary line running horizontally from the device's left to right. It indicates the up and

down movement of the device or the user's head, also known as forward flexion and backward extension.

Roll represents the rotation around the device's longitudinal axis, which is an imaginary line running from the front to the back of the device. It indicates the tilting movement of the device or the lateral flexion of the user's head from left to side.

Yaw represents the rotation around the device's vertical axis, which is an imaginary line running perpendicular to the ground. It indicates the left and right axial rotation of the device or the user's head.

The CMAttitude object provides these orientation values in the form of quaternions or radians. Quaternions are mathematical representations that are efficient for calculating complex rotations, while radians provide a more intuitive representation of pitch, roll, and yaw. To ease the calculations, they are converted to degrees using the formula: *180 / π * radians*.5.9

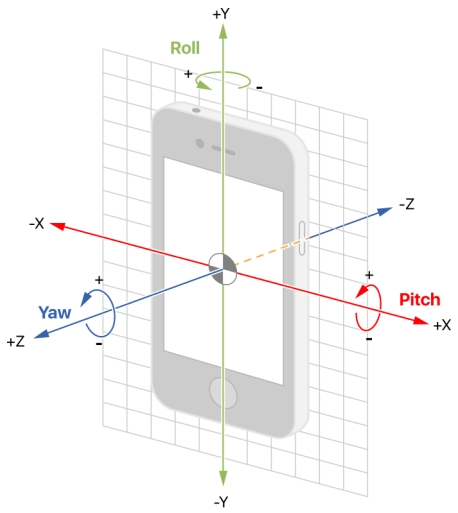


Figure 5.9: A visual representation of Yaw, Pitch and Roll available in CMAttitude, extracted from Apple Developer documentation.

5.3.2 Path Drawing in SwiftUI

The SwiftUI `DrawingView` monitors the changes published by the `MotionManager` and processes the new data and updates the drawing in real-time.

The line which describes the movement of the user's head in a particular exercise is stored as an array of `CGPoints`. A `CGPoint` id defines by its coordinates (x, y) and determines determining a point on the iPhone display. The possible values on this line are from the interval [0, 1], 0 meaning the neutral position and 1 meaning the maximum stretching position of that exercise. The points in between describe the user's progress as he performs the stretching. In order to map the degrees of the

current rotation to a point in the interval [0, 1], the value is divided by the maximum degree of rotation of the exercise, and the resulting value is added to the line, if it is in the interval [0, 1]. If the current rotation is the maximum that the user achieved up until that point, that value is also stored in the property `achievedRangeOfMotion` of the current exercise, in order to save it to his progress at the end of the stretching session. Below is an example of the approach, applied for the exercise Tilt Right, where the roll is processed. For the symmetrical exercise, Tilt Left, the roll would be negative, so the same algorithm is used, but the values are mirrored, such that the line is drawn from right to left.

```
import CoreMotion

struct Line {
    var points: [CGPoint] = []
    var color: Color = .danubeBlue
    var lineWidth: CGFloat = 10
}

struct TiltRightDrawingView: View {
    @Binding var exercise: StretchingExercise
    @ObservedObject var motionManager: MotionManager
    var isPaused: Bool
    @State private var line = Line()
    @State private var currentPosition: CGFloat = 0
    @State private var maximumX: CGFloat = 0

    var body: some View {
        Canvas { context, size in
            drawProgress(context: &context, size: size)
        }
        .onChange(of: motionManager.motion) { newValue in
            if isPaused {
                return
            }
             updateStretchingProgress(for: newValue)
        }
    }
    private func drawProgress(context: inout GraphicsContext, size: CGSize) {
        var path = Path()
        path.addLines(line.points.map({ point in
            CGPoint(x: point.x * size.width, y: size.height / 2)
        }))

        var backgroundPath = Path()
        backgroundPath.move(to: .init(x: 0, y: size.height / 2))
        backgroundPath.addLine(to: .init(x: size.width, y: size.height / 2))

        context.stroke(backgroundPath, with: .color(line.color.opacity(0.5)),
        style: StrokeStyle(lineWidth: line.lineWidth, lineCap: .round))

        context.fill(.init(roundedRect: .init(x: currentPosition * size.width - 12,
        y: size.height / 2 - 12,
        width: 24, height: 24), cornerRadius: 15), with: .color(line.color))

        context.stroke(path, with: .color(line.color),
        style: StrokeStyle(lineWidth: line.lineWidth, lineCap: .round))
```

```
    }

    private func updateStretchingProgress(for motion: CMDeviceMotion?) {
        guard let roll = motion?.attitude.roll else {
            return
        }
        let currentRoll = motionManager.degrees(roll) /
        Double(exercise.type.maximumDegrees)

        guard currentRoll > 0 && currentRoll <= 1 else {
            return
        }
        currentPosition = currentRoll

        if currentRoll > maximumX {
            exercise.achievedRangeOfMotion = currentRoll
            maximumX = currentRoll
            line.points.append(.init(x: currentRoll, y: 0))
        }
    }
}
```

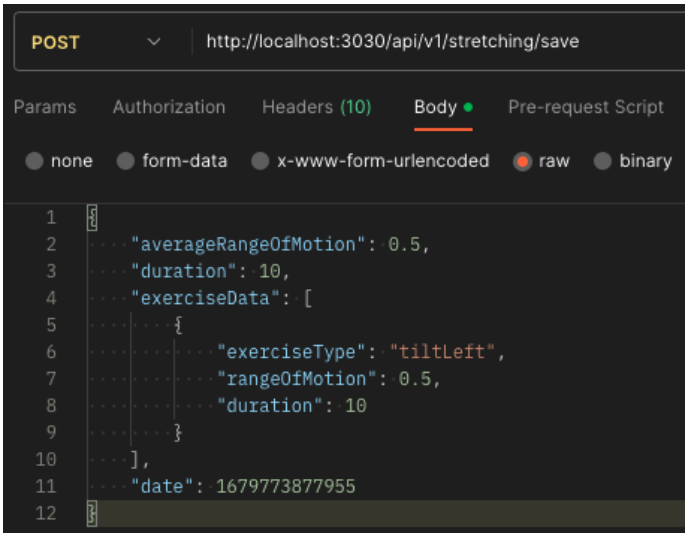
5.3.3 Saving the progress

At the completion of the stretching session, all the performed exercises, including the achieved progress for each one, together with the average range of motion and the total duration, are sent to the server, by calling `POST /stretching/save`, displayed in figure 5.10, which also depicts the screen that is shown to the user while the server request is in progress.

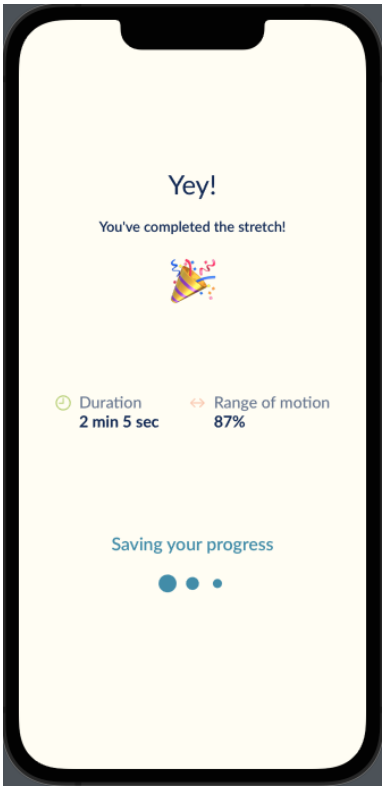
5.4 Progress Monitoring

To enable long-term progress monitoring and analysis, patients have access to a comprehensive analytics module, allowing them to track improvements over time, visualize trends, and identify areas that require further attention or modification in the rehabilitation program. By providing detailed insights and visualizations, the app facilitates data-driven decision-making and optimizes the effectiveness of the rehabilitation process.

Users can access a screen that displays the history of all the stretching sessions, together with a chart that visually depicts their progress over time. For this, the array of sessions is fetched using the endpoint `GET /stretching/allSessions`, with the bearer token as the authorization method. If the user is running on iOS 16 or higher, a chart is also drawn on the screen, using native charts from SwiftUI. The X-axis represents time, while the Y-axis represents the average range of motion obtained in that session. By tapping the session cards, a bottom sheet is presented,



(a) Save summary endpoint



(b) Summary screen

Figure 5.10: Saving the summary of the completed stretching session

that displays more detailed information and compares the measurements achieved for each type of movement in that session. 5.11

The summary can also be exported as a PDF and be saved to files or shared with a doctor, via various social apps. In order to achieve that in SwiftUI, `ShareLink` is used. To generate the URL, `pdfRendering()` creates an `ImageRenderer` object and generates a path for a PDF based on patient information and the stretching session date, relative to the Documents directory of the device. The function then uses the `renderer.render()` method to render the content into a PDF context created with the URL and returns the URL.

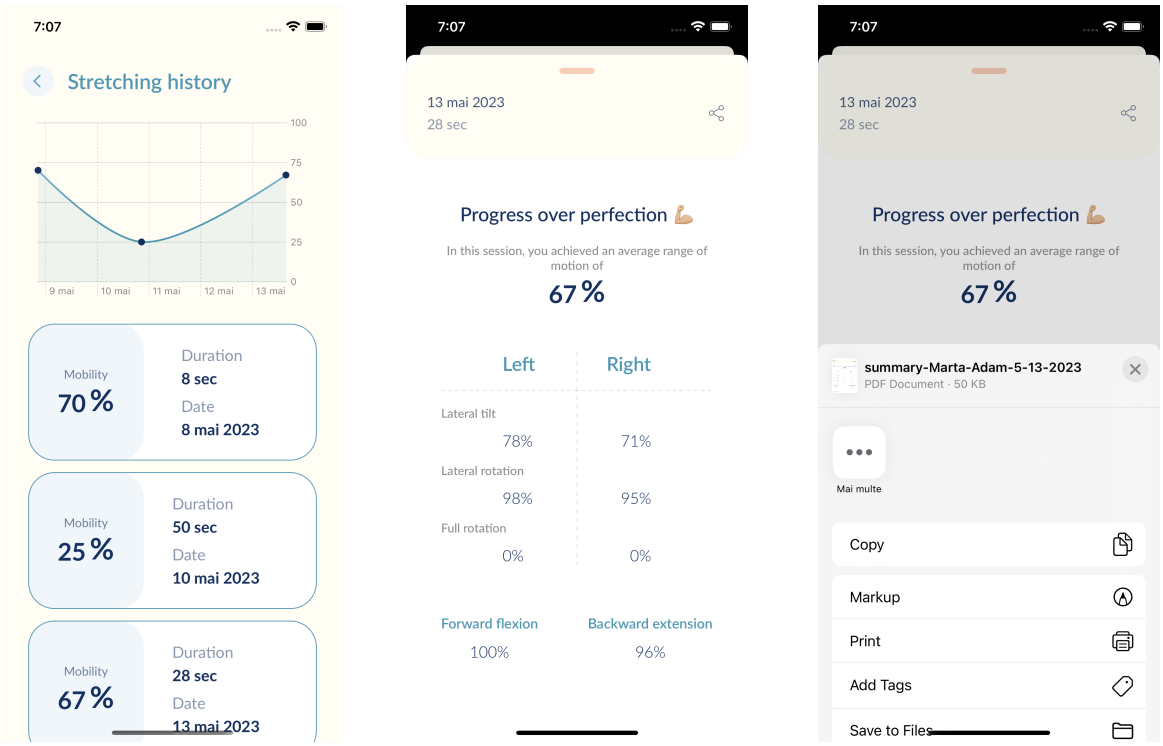
```
@MainActor
func pdfRendering() -> URL {
    if #available(iOS 16.0, *) {
        let renderer = ImageRenderer(content:
            DetailedStretchingInfo.PDFSummaryView(
                patient: patient,
                stretchingSession: stretchingSession)
            )
        let date = stretchingSession.date.toCalendarDate(.numeric).replacing("/", with: "-")
        let path = "summary-" + patient.firstName + "-" + patient.lastName + "-" + date + ".pdf"
        let url = URL.documentsDirectory.appending(path: path)
        renderer.render { size, context in
            var box = CGRect(origin: .zero, size: size)

            guard let pdf = CGContext(url as CFURL, mediaBox: &box, nil) else {
```

```
                return
            }

            pdf.beginPDFPage(nil)
            context(pdf)
            pdf.endPage()
            pdf.closePDF()
        }

        return url
    } else {
        return URL(string: "https://www.google.com/")!
    }
}
```



(a) Stretching history (b) Detailed stretching info (c) Export summary as PDF

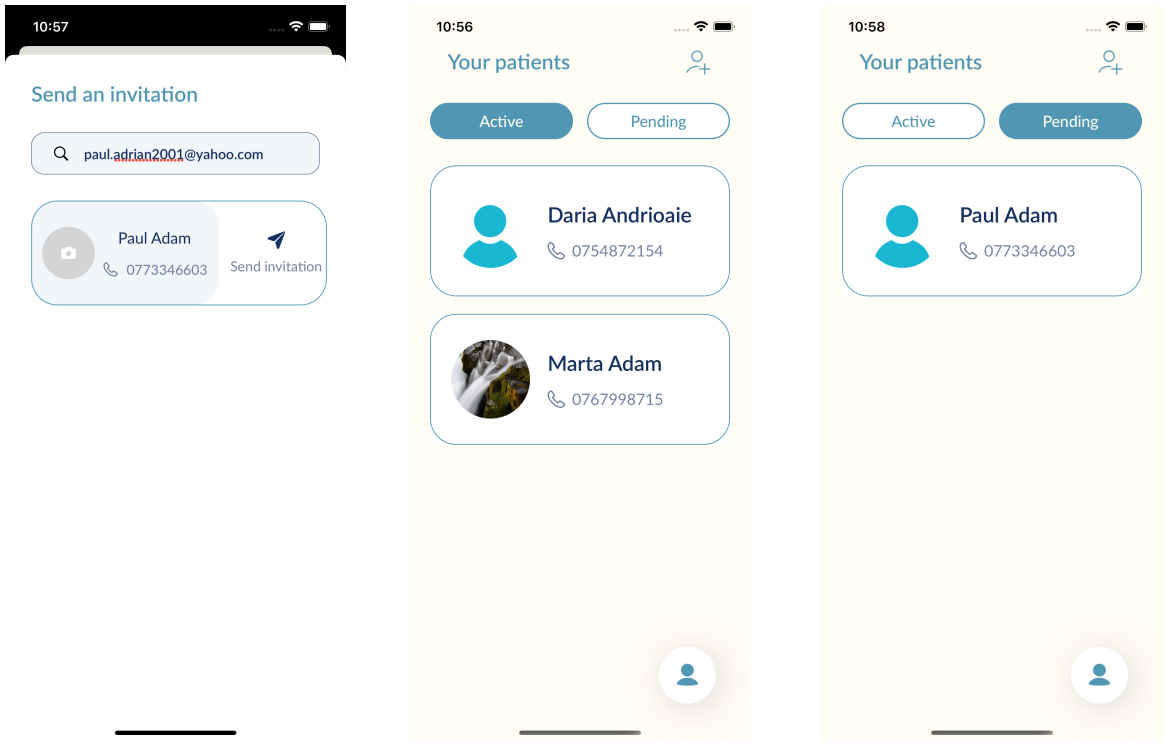
Figure 5.11: Stretching history

5.5 Therapist-Patient Collaboration

HeadFlow allows therapists to create personalized stretching programs for individual patients based on their specific needs and conditions. Based on the stretching history of the patient, therapists can evaluate their range of motion and flexibility to tailor the stretching exercises accordingly. The therapist also has the possibility to provide feedback for each session of his patients. By providing customized programs and feedback, HeadFlow optimizes the rehabilitation process and promotes more targeted and effective outcomes.

5.5.1 How They Connect

To connect with a patient, a therapist will search by his email address. The endpoint `POST /therapist/searchPatient` only returns a patient if the email address is valid and the patient is unassigned to another therapist. If a valid patient entity is returned, the therapist has the possibility to send him an invitation to collaborate, which will appear in the list of pending invitations. 5.12



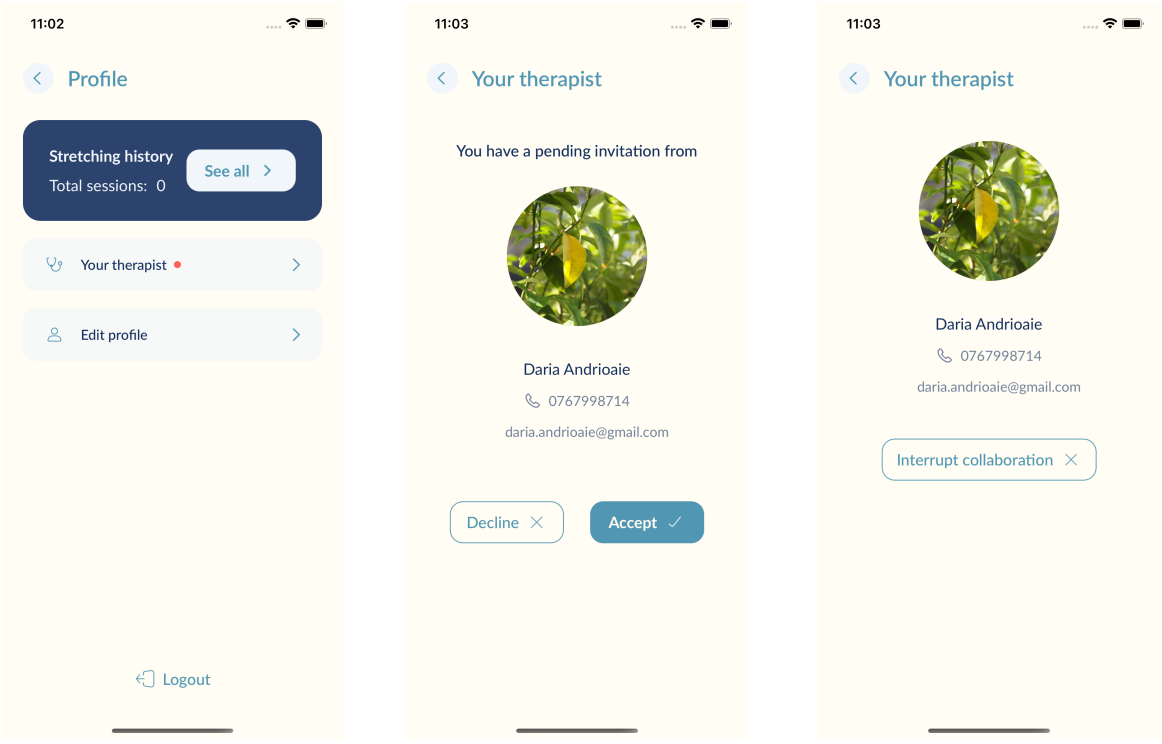
(a) Search patient by email (b) Active list of patients (c) Pending list of patients

Figure 5.12: List of patients

The endpoint used to send the invitation is `POST /therapist/sendInvitation`, which will create a new Collaboration document in the database, with the status "pending". The endpoint that returns all the collaborations, both active and pending, is `GET /therapist/allCollaborations`.

When a patient receives a collaboration invitation, a new section appears in the profile section, called "Your therapist". There, he sees the details of the therapist that sent the invitation and has the possibility to either accept or decline it. The endpoint called is `POST /patient/respondToInvitation`. Once he accepts it, the status of the collaboration in the database is changed to `active`, and he will always have the possibility to interrupt the collaboration, through a button that triggers a call to the endpoint `POST /patient/interruptCollaboration`. 5.13

The Collaboration schema is intended to act as the intermediary 'join table' between the Therapist and Patient collections, in order to replicate the many-to-many



(a) Therapist notification

(b) Acccept/decline invitation

(c) Interrupt collaboration

Figure 5.13: Handle collaboration with therapist

relationship from Relational Databases. To achieve this relationship in Mongoose, the 'foreign key' fields of the schema that reference a document in other collections are defined as ObjectId and annotated with the option `Ref`. When querying the collection for a document, if there is a requirement to retrieve the associated data from the referenced document through an inner join, the `populate()` function is used.

Below is the Collaboration schema and an example of how `populate()` was used on the server side to retrieve the data of both the patients and the therapist, from the collaborations of a given therapist Id.

```
const collaborationSchema = new mongoose.Schema(
  {
    therapist: {
        type: mongoose.Schema.Types.ObjectId,
        ref: 'user',
        required: true
    },
    patient: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'user',
      required: true
    },
    status: {
        type: String,
        required: true,
    }
  }
```

```
);

const allCollaborations = async(therapistId) => {
    const collaborations = await CollaborationModel
                            .find({ therapist: therapistId })
                            .populate(['patient', 'therapist']);
    return collaborations;
}
```

5.5.2 Session Tailoring

The therapist has access to the stretching history of all the patients that accepted their collaboration invitation.

After evaluating the patient's history, the therapist can alter the current stretching sequence, by adding, removing, reordering or editing the parameters of each exercise, such as the duration (in seconds) and goal degree of movement (since a therapist may want his patient to only perform the movement up to a certain degree, not to the maximum). These features can be seen in figure 5.14. The current sequence of exercises of a patient is retrieved as an array, from the endpoint `POST /therapist/plannedSession`. For saving the modified sequence, the endpoint `POST therapist/saveSession` is called. Unless the therapist has already tailored the sequence, the patient will perform the default stretching sequence, which includes all the exercises (axial rotation, lateral flexion, forward flexion, backward extension), for a duration of 5 seconds, with the maximum possible ranges, listed in section 4.2.

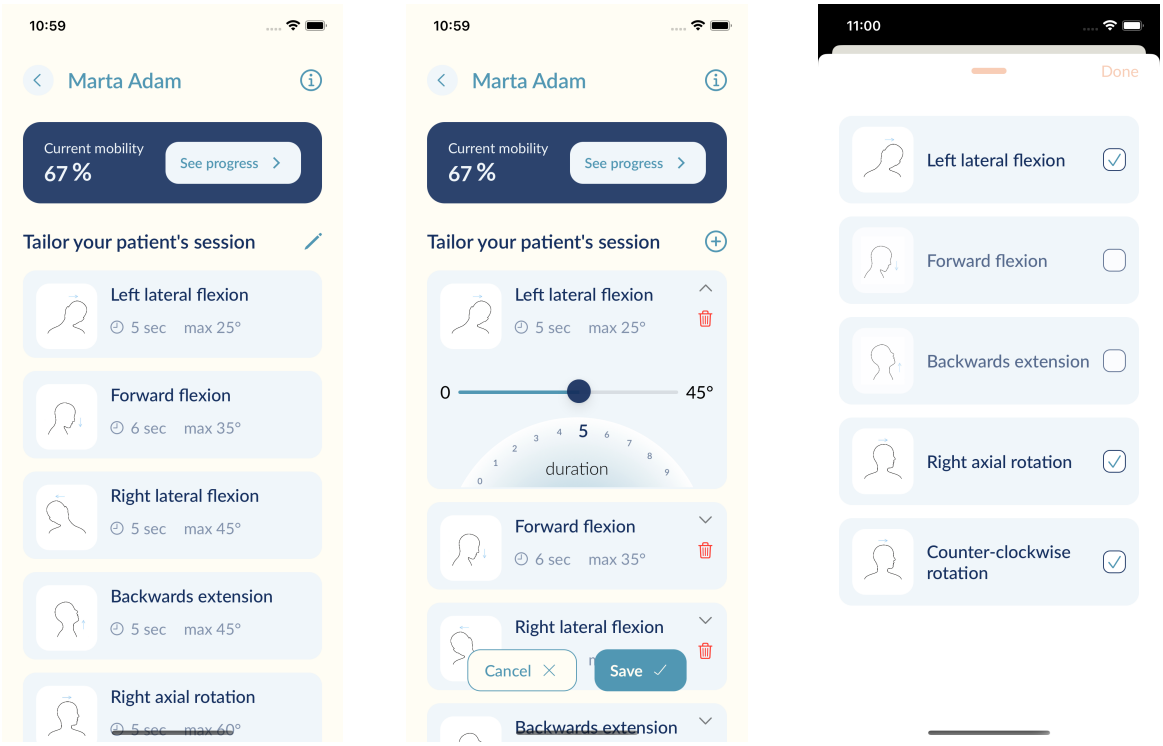
5.5.3 Session Feedback

In order to provide the possibility for coaching similar to that offered in clinics, but from the comfort of personal homes, therapists can leave a feedback message for a session that was completed, or edit that one that previously existed. The endpoint called to fetch the existing feedback of a session is `POST /feedback/getFeedback`, and the one that saves the new feedback is `POST /feedback/`. 5.15

5.6 Local Notifications

Each morning, at 9 AM, if the logged-in user is a patient and he gave his permission to receive notifications, he will receive a reminder to practice.

Because the notifications permissions are global to the whole app, and not relative to each user, I wanted to find a solution to let upcoming users know what is the current status of the notifications permissions and inform them that they can change



(a) Current session

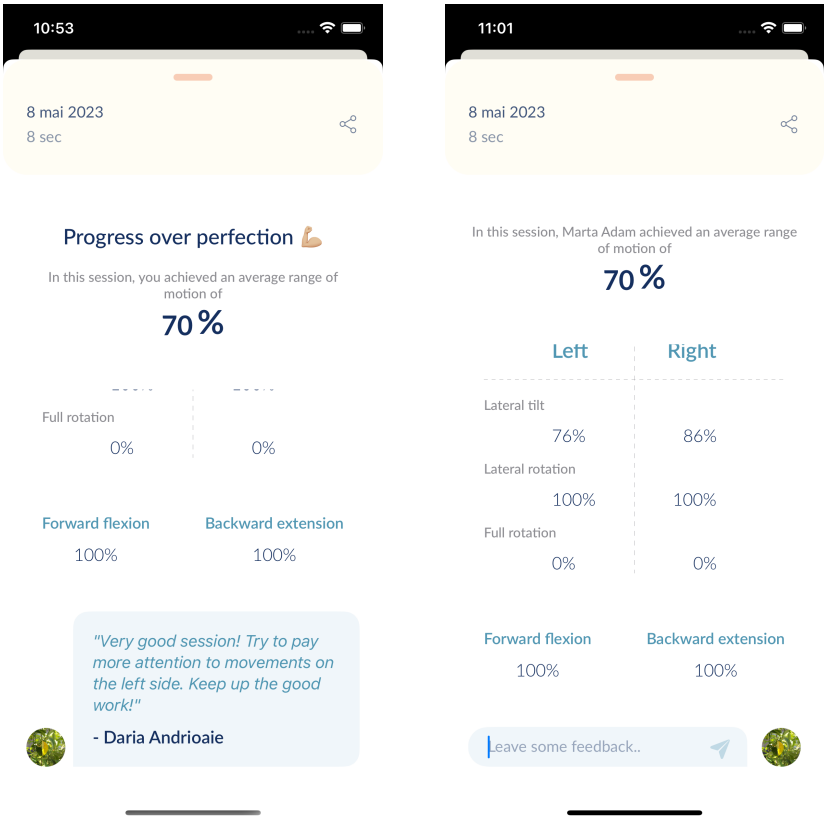
(b) Edit duration and goal degree

(c) Add missing exercises

Figure 5.14: Tailoring the stretching session of a patient

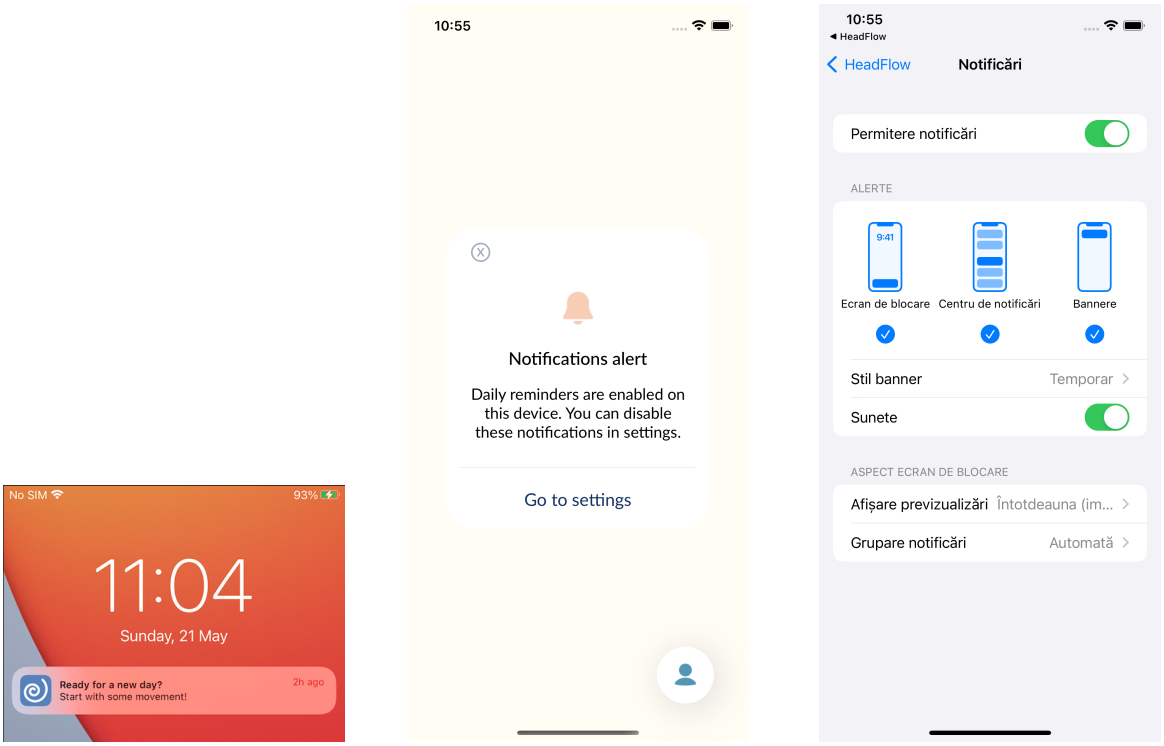
these permissions from the Settings app, but only give them this information once, at the very first entry in the app. To implement this feature, I set up a 'mini' local database, using Realm, that stores objects of the type `NotificationsPermissions`, composed from a `userID` and a boolean `notificationsStatusPresented`. For each new user that logs in on the same device, a new entry in the local DB is created, for the current userId and value false for the `notificationsStatusPresented`, because he hasn't yet seen the information of the current status.

When the user enters the Home Screen, the `NotificationsViewModel` determines the current notifications permissions, using the current (singleton) instance of `UNUserNotificationCenter`. If the authorization status is not determined, the app requests authorization for displaying alerts and sounds, and updates the `notificationsStatusPresented` value for the current user to true, in the local database. If permissions are either authorized or denied, it checks whether the notification notice has been presented before for the current user. If not, it presents an alert, notifying the user that notifications are either enabled or disabled, together with a button `Go to settings`, that redirects the user to the permissions settings. Afterward, the `notificationsStatusPresented` value for the current user is set to true, in the local database. The notice regarding current permissions, together with the daily notifications, is presented in figure 5.16.



(a) Patient point of view (b) Therapist point of view

Figure 5.15: Session feedback, from both perspectives



(a) Daily notification (b) Notifications status alert (c) Redirection to permissions settings

Figure 5.16: Local notifications and permissions status

After handling the authorization status, the notification content is created using an instance of `UNMutableNotificationContent`, which sets the title, body, sound, and badge properties of the notification. Next, to trigger the notification at a specific time each day at 9 AM, a `UNCalendarNotificationTrigger` is used. The final step is to create a `UNNotificationRequest` with the identifier `daily-cta`, the previously defined notification content, and the trigger.

Codebase The entire source code of the application is available at `https://github.com/daria-andrioaie/HeadFlow_DiplomaThesis`.

Chapter 6

Conclusions

In conclusion, this thesis has explored the potential of wearable computing in rehabilitation, specifically focusing on earable technology, in the context of assisted rehabilitation for cervical spine issues. Through the development and implementation of HeadFlow, this research has demonstrated the practical application of earphones as a tool to map sensor data to concepts of kinesiology and provide meaningful reports to medical professionals about the overall mobility of their patients.

By bridging the gap between technology and rehabilitation, this thesis contributes to the field of wearable computing by showcasing the potential of earable technology and providing a comprehensive understanding of the development and implementation of HeadFlow for cervical spine rehabilitation. The application's features, including progress monitoring, tailored stretching sessions, and collaboration between patients and therapists, offer a promising avenue for improving patient outcomes and enhancing the efficiency of rehabilitation processes.

Moving forward, further research and development in the field of earable technology hold immense potential for revolutionizing the realm of assisted rehabilitation. As technology continues to evolve, integrating advanced sensor capabilities and data analysis techniques, there is an opportunity to enhance the accuracy, effectiveness, and accessibility of rehabilitation programs. By leveraging the power of wearable computing, such as earphones, in conjunction with innovative software applications, future advancements in this field have the potential to positively impact the lives of individuals with cervical spine issues and other rehabilitation needs.

6.1 Future work

To make a bigger contribution to assisted rehabilitation in the field of earable computing, here are a few potential improvements that can be considered for HeadFlow:

Real-Time Feedback and Correction The capabilities of the app could be enhanced to provide real-time feedback and correction during the stretching exercises. By integrating advanced algorithms and machine learning techniques to analyze the user's movement patterns, posture, and alignment, HeadFlow can offer instant feedback and guidance to users, helping them improve their technique and ensure optimal execution of the stretching exercises.

Gamification and Engagement The incorporation of gamification elements and interactive features into HeadFlow could increase user engagement and motivation. By introducing challenges, achievements, and progress-tracking systems, users would be rewarded for consistent adherence to their rehabilitation routine. Additionally, integrating interactive games or immersive virtual reality experiences can make the stretching exercises more enjoyable and engaging, potentially improving adherence and long-term compliance.

Integration with other Wearable Devices Another possibility would be to integrate HeadFlow with other wearable devices, such as smartwatches or fitness trackers, to gather additional physiological data during the rehabilitation process. By combining data from multiple sources, including heart rate, sleep patterns, and activity levels, HeadFlow can provide a more comprehensive understanding of the patient's overall well-being and tailor the rehabilitation program accordingly, extending the use cases to the whole body of the patient, not just their cervical spine.

Integration with existing Tele-Rehabilitation Platforms The integration opportunities with telerehabilitation platforms could be explored, enabling seamless communication and remote monitoring between patients and therapists. This could include features like video consultations, instant messaging, and real-time access of the therapist to the stretching session that the patients perform.

By incorporating these improvements, HeadFlow can further advance the field of earable computing in assisted rehabilitation by offering a more personalized, interactive, and data-driven approach to cervical spine rehabilitation. These enhancements have the potential to improve patient outcomes, increase engagement and adherence to rehabilitation programs, and empower both patients and therapists with valuable insights and tools for effective rehabilitation management.

Bibliography

[AR18] Caulfield B. Argent R, Daly A. Patient involvement with home-based exercise programs: Can connected health interventions influence adherence? *JMIR Mhealth Uhealth*, 47(6(3)), 2018.

[Cor22] Motion Analysis Corporation. Cortex - our most powerful motion capture software yet, Apr 2022.

[DAP+92] Jiří Dvořák, James Antinnes, Manohar M. Panjabi, Daniel Loustalot, and Marco Bonomo. Age and gender related normal motion of the cervical spine. *Spine*, 17:S393–S398, 1992.

[ea90] A Dwyer et al. Cervical zygapophyseal joint pain patterns. i: A study in normal volunteers. *Spine*, 1990.

[ea91a] Kamwendo et al. Neck and shoulder disorders in medical secretaries. part i. pain prevalence and risk factors. *Scand J Rehabil Med*, 3(23), 1991.

[ea91b] Turk et al. Neglected topics in the treatment of chronic pain patients — relapse, noncompliance, and adherence enhancement. *Pain*, 44:5–28, 1991.

[ea96] Skov et al. Psychosocial and physical risk factors for musculoskeletal disorders of the neck, shoulders, and lower back in salespeople. *Occupational and Environmental Medicine*, 5(53):351–356, 1996.

[ea01] Ariëns et al. Are neck flexion, neck rotation, and sitting at work risk factors for neck pain? results of a prospective cohort study. *Occupational and Environmental Medicine*, 3(58):200–207, 2001.

[ea03a] Ortiz-Hernandez et al. Computer use increases the risk of musculoskeletal disorders among newspaper office worker. *Arch Med Res*, 2003.

[ea03b] Talbot et al. A home-based pedometer-driven walking program to increase physical activity in older adults with osteoarthritis of the knee: a preliminary study. *J Am Geriatr Soc*, 51, 2003.

[ea07a] Cagnie et al. Individual and work related risk factors for neck pain among office workers: a cross sectional study. *Eur Spine J.*, 16(5), 2007.

[ea07b] Demyttenaere K et al. Mental disorders among persons with chronic back or neck pain: results from the world mental health surveys. *Pain*, 2007.

[ea10] Pisters et al. Exercise adherence improving long-term patient outcome in patients with osteoarthritis of the hip and/or knee. *Arthritis Care Res (Hoboken)*, 62, 2010.

[ea14] Goto et al. Self-monitoring has potential for home exercise programmes in patients with haemophilia. *Haemophilia*, 20, 2014.

[ea17] Sá S et al. Repositioning error, pressure pain threshold, catastrophizing and anxiety in adolescents with chronic idiopathic neck pain. *Musculoskelet Sci Pract*, 2017.

[ea18] Liu F et al. Association of depression/anxiety symptoms with neck pain: a systematic review and meta-analysis of literature in china. *Pain Res Manag*, 2018.

[ea20a] Andias R et al. Psychosocial variables and sleep associated with neck pain in adolescents: a systematic review. *Phys Occup Ther Pediatr*, 2020.

[ea20b] Martinez-Calderon et al. Which psychological factors are involved in the onset and/or persistence of musculoskeletal pain? an umbrella review of systematic reviews and meta-analyses of prospective cohort studies. *Clin J Pain*, pages 626–637, 2020.

[ea20c] Mork R et al. Discomfort glare and psychological stress during computer work: subjective responses and associations between neck pain and trapezius muscle blood flow. *Int Arch Occup Environ Health*, pages 29–42, 2020.

[ea20d] Xie Y et al. Comparing central pain processing in individuals with non-traumatic neck pain and healthy individuals: a systematic review and meta-analysis. *J Pain*, 2020.

[Kaz22] Somaye et al Kazeminasab. Neck pain: global epidemiology, trends and risk factors. *BMC Musculoskelet Disord*, 23(1), 2022.

[LA08] Clough PJ Levy AR, Polman RC. Adherence to sport injury rehabilitation programs: an integrated psycho-social approach. *Scand J Med Sci Sports*, 18, 2008.

[MB] Melanie Martin and Lindsay Bearne. Telehealth.

[MTU+17] I M Moldovan, L Tric, R Ursu, A Podar, A D Călin, A C Cantea, L A Dascălu, and C A Mihaiu. Virtual rehabilitation programme using the mira platform, kinect and leap motion sensors in an 81 years old patient with ischemic stroke. In *2017 E-Health and Bioengineering Conference (EHB)*, pages 325–328, 2017.

[Reha] Evolv Rehab.

[Rehb] Mira Rehab. Play your way to recovery.

[Saf20] Saeid et al. Safiri. Global, regional, and national burden of neck pain in the general population, 1990-2017: systematic analysis of the global burden of disease study 2017. *BMJ (Clinical research ed.)*, 368, 2020.

[Tun16] Kuptniratsaikul Tunwattanapong, Kongkasuwan. The effectiveness of a neck and shoulder stretching exercise program among office workers with neck pain: a randomized controlled trial. *Clinical Rehabilitation*, 30(1), 2016.

[Vic] Life sciences — vicon motion capture for biomechanics. `https://www.vicon.com/applications/life-sciences/#components`.

[You16] Kwang Young. Clinical effects of deep cervical flexor muscle activation in patients with chronic neck pain. *The Journal of Physical Therapy Science*, 28:269–273, 2016.

[Yük18] Burcu Yüksel. Intervertebral movement analysis of the cervical spine : Prediction and categorising of cervical joint motion. 2018.