M.Sc. Thesis
Master of Science in Engineering

DTU Health Tech
Department of Health Technology

Master Thesis

Real-Time ECG Monitoring Mobile App for SUDEP Risk Mitigation

Adrian Macarenco (s202415)

Kongens Lyngby 2023

DTU Health tech
Department of Health Technology
Technical University of Denmark

Ørsteds Plads, Building 345C
DK-2800 Kgs. Lyngby
healthtech-info@dtu.dk
www.healthtech.dtu.dk/

Summary

Epilepsy, a neurological disorder marked by recurrent seizures, is a global health concern affecting approximately 50 million people worldwide. One of the most severe and least understood outcomes of this condition is SUDEP (Sudden Unexpected Death in Epilepsy), which impacts 1 in every 1,000 individuals with epilepsy. Interestingly, almost half of post-mortem examinations suggest that SUDEP cases are not associated with seizures, leading to the conclusion that other aspects such as cardiac and respiratory conditions could contribute to SUDEP.

In this context, the monitoring of cardiac and respiratory functions during seizures using real-time ECG data emerges as a potential preventive strategy. The Movesense MD sensor, a wearable device that measures and records physiological signals, including ECG data, is a key tool in this approach.

This thesis presents the design and development of a mobile application aimed at enabling epilepsy patients to monitor real-time ECG data sent by the Movesense MD sensor. The application's design places a strong emphasis on usability and accessibility to ensure that all users, regardless of their abilities and technical expertise, can use it effectively. The design process follows an User Centred Design (UCD) methodology and adheres to Apple's Human Interface Guidelines, which provide design principles and recommendations for creating user interfaces that adhere to iOS design standards.

A qualitative study involving epilepsy patients will be carried out to evaluate the usability and perceived usefulness of this system. The ultimate goal is to contribute to the development of a system that can assist in better managing cardio-respiratory abnormalities in epilepsy patients and lowering their risk of Sudden Unexpected Death in Epilepsy (SUDEP).

Preface

This master thesis was prepared at the Department of Health Technology at the Technical University of Denmark in fulfillment of the requirements for acquiring a Master Science degree in Computer Science and Engineering.

Kongens Lyngby, June 30, 2023

Adrian Macarenco (s202415)

Acknowledgements

The author expresses gratitude to Jakob E. Bardram, his supervisor, for providing continuous feedback and insightful ideas throughout this project. Furthermore, the author appreciates Jakob's technical assistance when addressing the implementation of the Server Upload feature.

The author is also grateful to his co-supervisor, ying Gu, for her extraordinary support, which included assistance in procuring the required hardware. Ying's prompt actions were especially remarkable when she quickly procured a new Movesense device to resolve a significant obstacle involving Faros iOS incompatibility. In addition, the author thanks ying for her active participation as a collaborator, subject, user, and co-supervisor during the design and development phases, as evidenced by their weekly meetings.

In addition, the author wishes to express gratitude to the MonstarLab team, with special thanks to his manager Jesper Christiansen, who provided understanding and consistent support throughout the thesis process. Christian Thompson and Tobias Vogt collaborated as participants, subjects, and evaluators.

Contents

CHAPTER 1

Introduction

1.1 Background

Epilepsy is a neurological disorder characterized by recurrent seizures that affects approximately 50 million people worldwide [Fis+14]. The estimate indicates that about 3.85 percent (1 in every 26 people) will develop epilepsy in their lifetime [Thu+11]. SUDEP is sudden and unexpected death of an epilepsy person without any obvious cause of death concluded by the post-mortem examination [THF14]. Moreover, SUDEP affects 1 in every 1,000 epilepsy people [Dev11].

SUDEP is considered as associated with cardio-respiratory arrests [Nas11]. According to previous studies almost half of post-mortem examination suggest SUDEP cases are not associated with seizure [LS01]. This led to the conclusion that other aspects, such as cardiac and respiratory conditions, could contribute to SUDEP. Therefore, monitoring changes in cardiac and respiratory function during seizures will give valuable insight into the pathophysiology of SUDEP and help develop preventive strategies.

An electrocardiogram (ECG) is a non-invasive device that is used to measure the heart's electrical activity and can provide valuable information about the heart's rhythm and its electrical activity. By streaming an electrocardiogram, an epileptic patient can monitor heart activity during an attack and provide medical professionals with valuable data for her SUDEP risk assessment.

Movesense MD is a wearable sensor that can be used to measure and record physiological signals, including ECG data. The device can transmit data to a mobile device for analysis and is light and small, making it simple to wear and use. The Movesense MD device provides reliable and accurate measurements of heart activity through its built-in ECG function. The device is designed for everyday use. It is small and light, making it comfortable to wear for extended periods. Its long battery life ensures that it can continuously monitor and transmit data for prolonged durations, ensuring comprehensive tracking.

However, despite the capabilities of the Movesense MD sensor in measuring and recording physiological signals, including ECG data, there are certain limitations that warrant the need for a mobile application. Considering the uncertainties related to SUDEP prediction, an effective solution could be a mobile application that enables individuals with epilepsy to monitor ECG data from the Movesense MD sensor in real-time. In addition, the data could be stored in the cloud for further analysis,

aiming to identify patterns that may indicate SUDEP. The proposed app would serve two purposes: first, as an epilepsy management tool with a focus on live ECG data visualization; and second, as an intermediate actor that facilitates the transmission of ECG data from the wearable device to a remote server. In order for the app to be used effectively and efficiently by a wide range of users, regardless of their abilities and technical expertise, it is essential that it places a strong emphasis on usability and accessibility.

Usability and accessibility are two essential app features that can have a great impact on an app's effectiveness and user adoption. The app will be created with a focus on usability and an intuitive and user-friendly interface that walks users through the process of monitoring and analyzing their ECG data to meet the needs and goals of patients and healthcare professionals. The app will be created with a focus on accessibility so that it can meet a variety of needs, including support for assistive technologies and clear, concise, and understandable language.

With an emphasis on usability and accessibility, we will discuss the creation and assessment of the app for streaming ECG data to epilepsy patients in this thesis. As well as going over the design and development process, we'll also talk about the feedback and testing from users. Ultimately, we want to develop a tool that can assist in better managing cardio-respiratory abnormalities in epilepsy patients and lowering their risk of SUDEP.

1.2 Research Questions

- **Question 1:** What is the UX design that encourages epilepsy patients to use a mobile application that streams ECG data? What are the main features the application must have?
- **Question 2:** How can these features be integrated into a mobile app with real-time functionality?
- **Question 3:** What are the main usability and accessibility requirements for the ECG control and monitoring mobile application for epilepsy patients, and how can these be implemented into the system design?
- **Question 4:** What is the state of the art in live ECG monitoring mobile applications?

1.3 Goals and Methods

The objective of this master's thesis is to design and implement a mobile application that monitors real-time ECG data specifically for individuals with epilepsy. To achieve this objective, the following goals have been identified:

- **Goal 1:** Design a mobile application applying UX knowledge.

- **Goal 2:** Integrate a real-time ECG device with a cloud-based service for uploading ECG data.
- **Goal 3:** Determine the key usability and accessibility requirements for the ECG control and monitoring mobile application for epilepsy patients and incorporate them into the system design.
- **Goal 4:** Explore the state of the art of live ECG monitoring mobile applications.
- **Goal 5:** Conduct a user-oriented study to evaluate the usability of the mobile app.
- **Goal 6:** Conduct a data analysis study to assess the accuracy of the ECG data stored on the server.

1.4 Thesis Overview

- **Background and related work** will include SUDEP's epidemiology, risk factors, pathophysiology, impact on quality of life, prevention strategies, and the use of wearable ECG devices. subsequently concludes with a discussion of available mobile applications for ECG monitoring and the necessity of an epilepsy-specific mobile app.
- **Design process** will focus on the UCD methodology applied to the design of the ECG monitoring mobile application. It describes the principles and steps of UCD, such as comprehending user requirements through interviews and the creation of personas. The chapter then describes the design characteristics determined by user expectations. The section then addresses the iterative creation of prototypes using the Figma software.
- **Implementation** will describe the technical implementation in Swift of the EpiHeartMonitor mobile application described in Chapters 3 and 4.
- **Validation** will provides an overview of the three study setups conducted by the author in order to validate the application, with an emphasis on the tools and methodologies used. The first study was a user study in which an individual was recruited to test and evaluate the product. The second study was a data analysis conducted by the author to guarantee accurate data transmission between the wearable device and the server. Finally, an accessibility study was conducted to evaluate the application's main accessibility features.
- **Results and Discussion** will present the results of the conducted studies. It analyzes how each study result validates the application and provides an analysis of each study result. In addition, the chapter includes a discussion of the system's strengths, limitations, and potential for development based on the findings of the study.
- **Conclusion** will present a comprehensive summary of the thesis findings and responses to the thesis's research questions.

CHAPTER 2

Background and related work

The second chapter offers an overview of SUDEP, including its epidemiology, risk factors, pathophysiology, impact on quality of life, and prevention strategies. Furthermore, it examines related research on mobile applications for ECG monitoring and emphasizes the significance of user feedback in the design of effective mobile health applications.

2.1 Sudden Unexpected Death in Epilepsy

Epilepsy is a complex, enduring neurological condition marked by repetitive, spontaneous seizures. It arises due to atypical electrical activity in the brain, causing various symptoms, including convulsions, loss of consciousness, and sensory disturbances [Fis+05]. Around the world, approximately 50 million individuals suffer from epilepsy, resulting in substantial medical, psychosocial, and financial consequences [Org19].

In spite of progress in treatment, epilepsy is associated with a higher probability of untimely death. One of the most severe outcomes related to epilepsy is Sudden Unexpected Death in Epilepsy (SUDEP). This pertains to the abrupt, inexplicable passing away of an individual having epilepsy with no other reason of death detected during an autopsy [Dev11].

2.1.1 Epidemiology of SUDEP

SUDEP incidence is estimated at approximately 1 in 1,000 individuals with epilepsy each year [Hes+11]. Nonetheless, these figures might not accurately reflect the true occurrence since incidents of SUDEP are frequently overlooked or unrecorded. Significantly, the likelihood of experiencing SUDEP varies among individuals with epilepsy, and various factors that increase this risk have been recognized. These factors include the frequency of generalized tonic-clonic seizures, early epilepsy onset, poor seizure control, polytherapy with antiepileptic drugs, and nocturnal seizures [Ryv+13].

2.1.2 Risk Factors for SUDEP

Individuals with poorly controlled seizures, particularly generalized tonic-clonic seizures, are at an elevated risk of SUDEP. Early onset of epilepsy, especially childhood onset, additionally appears to correlate with multiplied risk. Complex medication regimens, which suggest inadequate management of seizures or the presence of severe forms of epilepsy, represent yet another noteworthy risk element. The occurrence of seizures during sleep or nocturnal seizures can also raise SUDEP risk due to the likely absence of assistance during a seizure occurrence [Ryv+13].

2.1.3 Pathophysiology of SUDEP

The pathophysiology underlying SUDEP remains an active area of research, with several proposed mechanisms. These generally revolve around seizure-related respiratory dysfunction, cardiac arrhythmia, and autonomic dysregulation. Post-mortem findings often suggest significant pulmonary edema and neurogenic pulmonary edema, indicating possible seizure-related respiratory dysfunction [NHM07].

Cardiac arrhythmia is another significant pathophysiological aspect of SUDEP. Convulsions have the potential to cause irregularities in the heart's rhythm, such as ictal sinus tachycardia, ictal bradycardia, and postictal asystole. Mounting evidence additionally suggests a rise in the frequency of arrhythmogenic syndromes like Long QT syndrome and Brugada syndrome in epilepsy patients [NHM07].

2.1.4 Impact on Quality of Life

The risk of SUDEP can have significant psychological impacts on patients with epilepsy and their families, often contributing to anxiety and depression. The apprehension of SUDEP has the potential to adversely impact the quality of life of patients and create difficulties for healthcare providers who strive to maintain transparency regarding the risks of SUDEP while being mindful of the possibility of causing anxiety or fear.

Understanding and addressing these challenges is an essential aspect of epilepsy care. As such, clinicians must employ a patient-centered approach, facilitating open discussions about SUDEP while providing reassurance and outlining potential risk mitigation strategies.

2.1.5 Prevention of SUDEP

SUDEP prevention strategies mainly focus on optimizing seizure control, educating patients and caregivers about SUDEP, and modifying identified risk factors. Recent initiatives aim to provide nighttime surveillance or observation for patients with a heightened risk of SUDEP, as there has been a rise in the occurrence of SUDEP while they are asleep. These may involve the usa of auditory aids or alerts that trigger upon identification of seizure occurrence.

The advent of wearable technology represents a promising frontier in SUDEP prevention. Wearable devices equipped with sensors that can detect abnormal heart rhythms offer the potential for real-time monitoring. This enables prompt intervention when irregular cardiac activity is detected, which is therefore indicative of a potential SUDEP episode. This live monitoring can be essential, as heart rhythm disorders are abrupt and demand prompt attention.

This method takes advantage of the progress in modern technology and the prevalence of portable connectivity, providing a proactive resolution that elevates the patient's security and tranquility. Unlike traditional methods that react to seizures, this approach shifts the paradigm towards prevention through continuous monitoring and timely intervention.

Additionally, wearable devices that can detect anomalous motions linked with epileptic seizures could also aid in SUDEP prevention. These devices, built upon sophisticated algorithms, decipher the movement data of the user, distinguishing between regular and seizure-related movements.

2.2 Electrocardiogram

An Electrocardiogram (ECG) is a medical test that detects heart problems by measuring the electrical activity generated by the heart as it contracts. As a non-invasive, painless test with rapid outcomes, electrocardiograms have become crucial instruments in the field of healthcare.

The cardiac muscle produces electric impulses that travel through the heart in a highly controlled and orderly manner, producing a specific sequence of contractions. These electrical signals originate in the sinoatrial node (the heart's natural pacemaker), and subsequently, they disseminate throughout the atria, arrive at the atrioventricular node, and finally enter the ventricles, causing them to contract.

An ECG tracing represents the electrical activity of the heart. Every heartbeat is represented by distinct waves (P, Q, R, S, and T) that match particular stages of the heart's electrical process.

2.2.1 Clinical Applications of ECG

ECGs are frequently used to detect various heart ailments, including arrhythmias, heart attacks, and other abnormalities in the heart's structure.

Regular electrocardiograms are also useful for observing cardiac wellbeing in patients with pre-existing heart conditions or those at risk, providing medical professionals with insights into how a patient's heart condition is progressing or their reaction to therapy.

Recent studies indicate that specific ECG irregularities may be more widespread among individuals with epilepsy, potentially offering a non-invasive approach to detecting individuals with a heightened risk of SUDEP [Ryv+13].

2.2.2 Understanding ECG Results

A typical ECG waveform consists of the P wave, QRS complex, and T wave. Each wave corresponds to a specific part of the heart's electrical cycle.

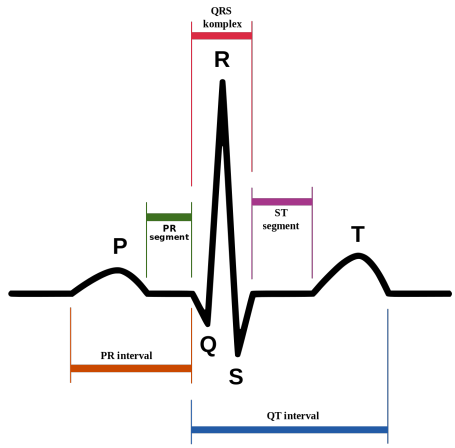


Figure 2.1: Fundamental ECG waveforms, intervals and segments [AB20].

Abnormal ECG patterns can indicate various heart conditions. For example, an ST segment elevation can indicate a myocardial infarction, while an irregular QRS complex may suggest an arrhythmia.

Arrhythmias, or abnormal heart rhythms, are typically detected through distinctive patterns on an ECG. This could involve irregularities in the P wave, QRS complex, or T wave.

2.2.3 Wearable ECG Devices and Their Importance

The latest technological progress has facilitated the development of wearable ECG devices. Contrary to traditional ECGs, these devices are portable, which gives users the ability to track their heart function in real-time, outside of clinical facilities.

Continuous electrocardiogram monitoring may assist in identifying intermittent or subtle arrhythmias that could be overlooked during a routine ECG examination. Furthermore, it can provide invaluable information regarding how heart activity varies in response to daily activities or particular triggers.

Wearable ECG devices empower individuals to take a more proactive role in controlling their health. Real-time data can assist patients in making lifestyle changes, alert them to potential problems, and help them seek medical attention promptly [Bau+16].

2.2.4 Modern ECG Monitoring Technologies

ECG technology has significantly evolved, from the first bulky, static devices to compact, wearable, and even implantable units. Modern devices often integrate with smartphone applications, providing easy access to real-time data.

While Holter monitors have traditionally been used for long-term heart rhythm monitoring, they can be bulky and uncomfortable. Modern wearable ECG devices provide a convenient and comfortable alternative, enabling continuous heart monitoring.

Artificial Intelligence (AI) is increasingly being incorporated into ECG monitoring systems, with algorithms capable of detecting subtle changes or patterns in heart rhythms that might be indicative of specific heart conditions.

2.2.5 Real-Time ECG Monitoring

Real-time ECG monitoring provides immediate feedback on heart activity, allowing for timely intervention in the event of an emergency. It is particularly useful in managing conditions such as epilepsy, where a timely response to heart rate changes could be lifesaving.

Despite its benefits, real-time ECG monitoring poses several challenges, including data privacy and security issues, as well as ensuring the accuracy and reliability of continuous data. However, ongoing technological advancements are continually addressing these challenges.

Future trends in real-time ECG monitoring include the integration of AI algorithms for enhanced diagnostic capabilities, as well as the development of even more compact, accurate, and user-friendly wearable devices.

2.2.6 Movesense MD Sensor

The Movesense MD sensor is a small, wearable device capable of recording high-quality ECG information. It's specifically created for continuous, real-time heart tracking, making it an important tool for managing conditions such as epilepsy.

The Movesense MD sensor is designed to function as a device for measuring both ECG and motion signals. It transmits the signals it records to other devices for analysis via BLE technology. Movesense MD extracts R-peaks from the recorded ECG signal with a modified Pan-Tompkins algorithm, permitting measurement of the R-R interval and subsequent calculation of heart rate [Mov20].

The Movesense MD sensor is characterized by its key features, such as its lightweight and waterproof design, long battery life, and high-quality ECG data. The device can be comfortably worn for prolonged periods, making continuous heart activity monitoring feasible.

The Movesense MD sensor pairs seamlessly with a mobile application, providing developers with accessible and easy-to-integrate Bluetooth Low Energy (BLE) connectivity. This results in platform versatility, as the technology is capable of establishing

a connection with mobile devices running on both Android and iOS operating systems.

2.3 Related work: Mobile App for ECG monitoring

According to earlier studies, ECG monitoring can improve the detection of cardiac events that could cause SUDEP and help with epilepsy diagnosis and treatment. Existing ECG monitoring devices, however, are frequently large and expensive, making them unsuitable for long-term use by patients in their daily lives, while free apps target general users or fitness users that generally do not fit epilepsy patients needs.

2.3.1 Mobile Cardiac Telemetry

Bittium's Mobile Cardiac Telemetry app is a mobile app for people with cardiac problems that allows users to continuously monitor heart function. It integrates a Bittium Faros ECG device, which allows ECG streaming and real-time arrhythmia detection. The service enables continuous remote monitoring outside of the hospital for up to 30 days. Rapid response to potential problem situations is made possible by remote monitoring. The patient can use a digital diary in the mobile application to record symptoms, activity, and sleep during an ECG recording. Using remote analysis tools, the specialist can monitor the patient's health on a daily basis and, based on the information they receive, determine how long the required recording period should last.

2.3.2 Beurer HealthManager Pro

ECG functionality is available to users of the mobile health app HealthManager Pro. Users can easily record their ECG readings and follow them over time in the app with the help of an external ECG device. The app is a complete tool for monitoring cardiovascular health because it also has features like heart rate monitoring and blood pressure tracking. The app is not epilepsy patient-oriented; it is a general health monitoring app. The app allows users to record and save ECG readings; it does not support real-time ECG monitoring.

2.3.3 KardiaMobile

AliveCor's KardiaMobile is a smartphone app with real-time ECG monitoring features. The KardiaMobile ECG device, a compact device that snaps onto the back of a smartphone or tablet, is intended to function in tandem with the app. A medical-grade ECG reading can be obtained using the device's two metal electrodes, which the user applies to their fingers. Users can quickly record and save real-time ECG readings using the KardiaMobile app. The app also has a feature called SmartRhythm

that tracks heart rate and rhythm using artificial intelligence and alerts the user if an irregularity is found.

2.3.4 Biostrap

Users of the Biostrap app can view comprehensive ECG data, including heart rate variability, respiratory rate, and heart rate recovery. The app integrates a wearable wristband device to get data. Advanced analytics tools are also included in the app, enabling users to monitor their development over time and spot trends and patterns in their ECG data. One of the distinguishing characteristics of Biostrap is its capacity to analyze ECG data in real-time and offer individualized feedback and coaching based on the user's present fitness level and heart rate. The app also offers a variety of features for tracking workouts, such as automatic exercise recognition and in-depth performance analysis. Generally, the Biostrap app is oriented toward fitness users and does not have medical device integration.

2.3.5 EpDetect

Developed to notify caregivers when a seizure occurs, EpDetect is a mobile epilepsy seizure alert application. To identify movements that might be seizure-related, the app uses the phone's accelerometer and gyroscope sensors. The app notifies designated caregivers of seizure detection and sends them a notification with the seizure's time and location.

Since EpDetect doesn't keep track of the patient's heart rate or rhythm, it lacks ECG capabilities. Instead, it only uses the phone's sensors to identify seizure activity.

2.3.6 EpiWatch

Johns Hopkins University created the research app EpiWatch to use the Apple Watch to track epilepsy patients' seizures. The app makes use of the watch's sensors, such as the accelerometer and gyroscope, to identify unusual movements that could be seizure-related. The app prompts the user to enter seizure details, including the type and duration, when one is detected. EpiWatch doesn't have ECG capabilities because it does not integrate a heart rate sensor. It does, however, give patients a way to monitor their seizures and get in touch with their medical professionals. Along with seizure management advice, the app also provides educational materials on epilepsy.

Designing mobile health applications that are user-friendly and accessible to a variety of patients has been heavily researched in terms of usability and accessibility. There have been several studies highlighting the significance of incorporating user feedback into the design of mobile health applications to enhance usability and patient satisfaction [Har+17].

Overall, there is still a need for an extensive and practical mobile app that is especially created for streaming ECG data to epilepsy patients, despite prior attempts

to develop mobile ECG monitoring systems for patients with epilepsy and significant research on designing user-friendly and accessible mobile health applications. Movesense MD device in combination with the proposed mobile app, epilepsy patients will have access to a practical and reliable tool for real-time ECG monitoring. The device is a great option for monitoring heart activity during seizures due to its small size, long battery life, and precise recording system. By creating and analyzing a mobile app that satisfies the particular demands and needs of epilepsy patients for ECG monitoring, this thesis seeks to fill this gap in the literature.

CHAPTER 3

Design process

The creation of user-friendly and intuitive applications is covered under app design. The process entails a deliberate and methodical strategy for developing the User Interface (UI) and User Experience (UX) in order to guarantee that the application fulfills the requirements and expectations of its designated users.

Throughout the app design process, this thesis project followed the principles of UCD. The design process of UCD centers around the users, prioritizing the comprehension of their needs, actions, and preferences. This involved conducting user studies, constructing personas, and iteratively evaluating and enhancing the design based on valuable user feedback.

3.1 User Centered Design

UCD is a commonly acknowledged design methodology that forms the foundation for creating effective and user-centric applications [ND88].The core concept of UCD revolves around prioritizing the users in the design process and involving them as dynamic contributors throughout the entire design phase. The ultimate goal of User-Centered Design is to create applications that not only meet functional requirements but also offer an user interface that is shaped by user needs.

The UCD methodology involves various key steps. The first step is to establish a thorough understanding of the target users using methodologies like user interviews and observations. This empowers the creation of personas, which are fictitious representations of different user archetypes, encapsulating their characteristics, objectives, and pain points. These personas act as directing frameworks for creating informed design choices, prioritizing the requirements and preferences of the users at the forefront.

Furthermore, UCD highlights the importance of iterative design and assessment. Prototypes are created and put through usability testing and user feedback sessions. This cyclic process allows ongoing refinement and enhancement of the design, ensuring that it meets the specific needs and expectations of the users. By engaging users in the evaluation, UCD strives to discover usability problems, address the user's needs, and eventually produce a remarkably user-friendly and satisfying application.

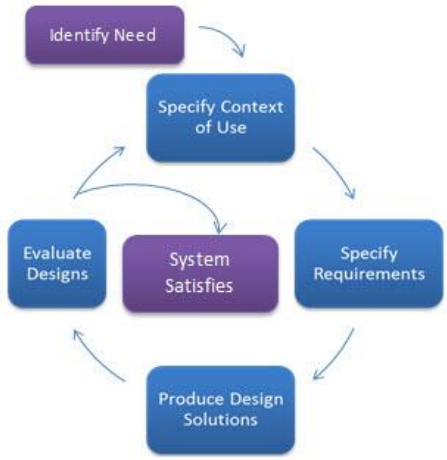


Figure 3.1: User-Centered Design Process [Usa].

3.1.1 Understanding User Needs

An intricate understanding of the needs and expectations of the end users and interested parties marks the beginning of the UCD process.

For this venture, two distinct persona types were created to represent the users of the application. The first type is represented by epilepsy people who want to visualize their live ECG data. Their primary objective is to monitor their cardiac activity in real time and gain insights into their health condition. The second type is represented by researchers who want to have ECG data on a server so that they can utilize the data for their research. The objective of these researchers is to make use of the information for their own research purposes, leveraging it to further their studies and contribute to the understanding of epilepsy and associated disorders.

Persona 1: Mike

Mike, a 35-year-old software engineer, found his life intertwined with epilepsy five years ago. As an individual who navigates the digital world with ease, Mike is familiar with the domains of mobile applications and wearable devices. Mike's aspirations center around the capability to monitor his heart activity during seizures, providing a wealth of information for his SUDEP risk assessment. His quest also drives him to seek an application that can seamlessly track his ECG data and pill intake in real-time, all within an easy-to-use mobile app.

Persona 2: Anna

Anna, a 50-year-old school teacher, has been living with epilepsy for ten years. Despite her limited experience with technology, Anna's desire to better manage her epilepsy motivates her willingness to learn. Her concerns about her risk of SUDEP have sparked a determination to be proactive in managing it. Like Mike, Anna

aims to monitor her cardiac activity during seizures to provide valuable data for the evaluation of the risk of SUDEP.

Persona 3: Mary

Mary is a research and study expert in cases related to SUDEP with a keen interest in analyzing Electrocardiogram (ECG) data. Her focus is to comprehend the trends and patterns associated with SUDEP by conducting extensive research. As someone well-versed in data analysis and interpretation, Mary needs a mobile application that can seamlessly integrate with a wearable ECG device. Her requirements include displaying real-time data in a meaningful and comprehensive way and the necessity for cloud storage to store the collected ECG data. Mary specifically requires the app to transmit data to the cloud in real-time, as she aims to analyze it promptly and utilize predictive modeling techniques to identify potential SUDEP patterns. Additionally, Mary intends to monitor medication consumption to gain a more profound understanding of its impact on ECG variations.

By exploring the requirements and anticipations of both types of users, we can ensure that the application is shaped in a way that is user-friendly and aligned with the project's objectives.

3.1.2 User Target Interview: Understanding User Expectations

By directly engaging with the target users, valuable perspectives can be obtained to guide the design and development of a mobile app. This interview aims to explore users' experiences, particular feature preferences and concerns, and the significance they attribute to the accessibility and usability aspects of the app.

- **Question 1: Could you please share your experience regarding the ECG monitoring process?**

  By understanding participants' experiences, we can acquire insights into their acquaintance with ECG surveillance and any challenges they might have confronted.

- **Question 2: What specific features or functionalities would you like to see in an application designed to assist with epilepsy management and ECG monitoring?**

  This question aims to reveal the users' preferred capabilities and features of the app, enabling us to align the application with their requirements and preferences.

- **Question 3: In your opinion, what is the biggest issue when using an app that connects to a wearable device?**

  Identifying possible challenges, we can address any issues associated with wearable device connectivity and enhance the overall user experience.

- **Question 4: On a scale from 0 to 10, where 0 represents the least importance and 10 signifies the highest importance, how important is accessibility for an ECG monitoring mobile app to you?**

  This inquiry evaluates the participants' prioritization of accessibility functionalities in the app, providing insights into their demands for inclusivity and ease of use.

- **Question 5: On the same scale, how important is usability for an ECG monitoring mobile app to you?**

  This inquiry evaluates the participants' emphasis on intuitive and user-friendly design in the app, which will assist us in giving priority to the usability aspects.

3.1.3 Designed Features

The resulting features are a reflection of user expectations and insights gathered from user target interviews. The following key features have been identified based on these considerations:

- **Live Visualization of ECG Data with Customization and Accessibility**

  One of the main features of the application is the live visualization of ECG data. Users can view their real-time ECG data through an intuitive and customizable visualization interface. The solution allows users to personalize the visualization settings according to their preferences. Accessibility is given priority, with UI components designed to be easily readable and adjustable for users with varying visual capabilities, emphasizing inclusivity and user friendliness. The incorporation of an app localization feature can be considered for the purpose of enhancing accessibility.

- **User-Friendly Bluetooth Connection Usability**

  Recognizing the importance of wearable device connectivity status, the application ensures a user-friendly Bluetooth connection experience. Users can easily access the connection status and battery level of their wearable devices within the app's interface. A lock screen widget has been designed to address this matter. This improves the overall usability and user experience, allowing users to easily link up and interact with their wearable devices.

- **Server Uploading Feature for Data Management**

  To fulfill the researchers needs for cloud storage, the application integrates a server uploading feature. This functionality enables the app to securely store their ECG data on a remote server, ensuring data integrity and availability.

- **Pill Intake Feature for Medication Tracking**

Based on the researchers requirements, a pill intake feature has to be integrated. This functionality allows researchers to correlate ECG data with specific pill intakes.

3.2 Prototype

Once the user expectations were identified and established as feature requirements, the next step of the design procedure was to create an application prototype.

To achieve this, the author of the thesis used Figma software, a platform for designing and prototyping. Figma empowers designers to create interactive and visually pleasing prototypes, which facilitate a realistic preview of the application's user interface and functionality.

The first step involved transforming the established feature prerequisites into a visual representation of the mobile interface for the user. This included designing the layout, navigation flows, and interactive components of the UI solution. Utilizing Figma's selection of design assets and cooperative features, the author was able to produce an initial prototype that mirrored the desired user experience.

3.2.1 First prototype

The first prototype included a comprehensive "Getting Started" flow that allowed users to create their accounts. The prototype further showcased a tabbar navigation with two primary tabs: "Dashboard" and "Profile."

Within the "Dashboard" tab, the prototype integrated functionalities such as device scanning and connection. Users are able to connect/disconnect and also visualize a live preview of their ECG data. Additionally, an expanded view of their ECG data is available.

To enhance the user experience, the prototype provides three customization options for ECG visualization. The first controller allows them to select the ECG intervals in seconds. The second controller is a color picker for the graph color while also addressing accessibility needs. The last is the ECG frequency selector, which served as a device control functionality, giving them control over the sampling rate or the speed at which the ECG data was captured and displayed.

The initial prototype was shared with users to collect valuable feedback, which was then utilized to iterate and refine the design. The author used a Figma prototype sharing link, allowing users to interact with the product design and functionality. This feedback-driven iteration process aimed to improve usability and align it more with users expectations.

First prototype link.

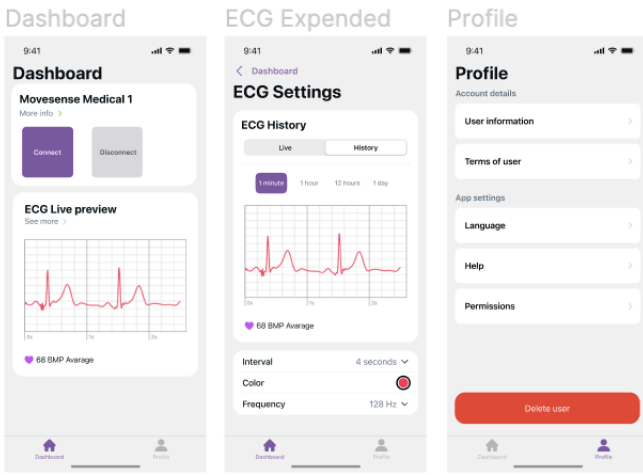


Figure 3.2: Main screens of the first prototype.

3.2.2 First prototype feedback

The input from users was collected using the comments feature in the Figma prototype. Users found it easy to use and intuitive. The Tabbar solution as the primary navigation and the use of navigation flows as navigation actions added to a native platform application feel, making the application simple, visually attractive, and familiar to users.

Users' feedback:

- The main screen's connect/disconnect button was not aligned with the overall app design, suggesting a potential design inconsistency.
- Users expressed confusion about the ECG expanded screen, specifically regarding the ECG History view and its purpose.
- Uncertainty about the purpose of the medication list, leaving users unsure of its intended functionality within the app.

Researcher's feedback:

- There was a suggestion to consider implementing a pill intake feature in the app, allowing users to track their pill intake.
- There were questions raised regarding the significance and necessity of the ECG History view within the app.

3.2.3 Second prototype

In the second prototype, the input from the users regarding the design and features of the initial prototype was considered, resulting in multiple changes and improvements. To improve the user experience, the user interface solution incorporated valuable feedback from users.

A noteworthy change made in accordance with feedback from researchers and users was the creation of a dedicated section for pill intake tracking. This resulted in a new ”Track Intake” tabview. Additionally, based on both user types feedback, the decision was made to remove the ECG History segmented controller option from the expanded ECG Settings screen.

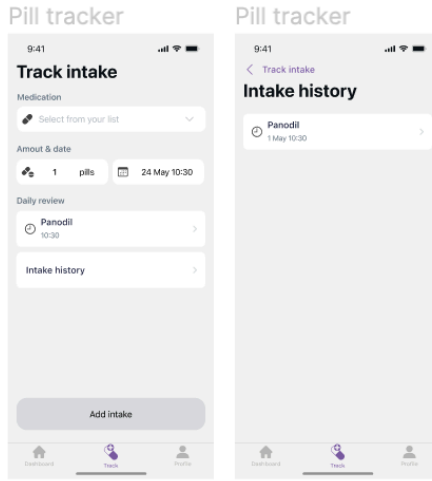


Figure 3.3: Track pill intake feature.

Similarly to the first prototype step, the second prototype was shared through a Figma sharing URL.

Second prototype link.

3.2.4 Second prototype feedback

Regarding the new ”Track intake” section feedback, it was positive, with people finding it intuitive and easily accessible. Furthermore, they considered the ”Intake history” and ”Edit intake” screens intuitive, referencing the iOS native app similarity.

Users’ feedback:

- The device connectivity is not really clear, it is just buttons that state the connection status.

Researcher's feedback:

- Consider the device battery information when the device is connected.

- Accountless app feature for users that just want to visualize ECG without providing any personal data.

After taking into account the feedback provided, the requested changes were implemented in the design. The battery level icon was added to the device cell on the Dashboard screen. A "Skip" action was added in the "Getting Started" screen so that users can skip the personal data setup and navigate directly to the main screen. Additionally, the connectivity problem was addressed.

Before discussing the solution to the connectivity problem, the final prototype can be accessed through the following Figma link:

Second prototype link.

3.2.5 Lock Screen Widget: Enhanced Connectivity and Battery Status Display

Given the Bluetooth connectivity status, the author conducted research on available iOS APIs and identified two potential solutions.

Apple recently released two new iOS features: Live Activities and the Lock Screen Widget.

Live Activites are updated based on real-time events and activities directly on the Lock Screen. It is designed to display information about events such as sports scores, food delivery, taxi updates, and more without unlocking the iPhone. This feature eliminates the need to open the app just to access the visualization of a given product's status.

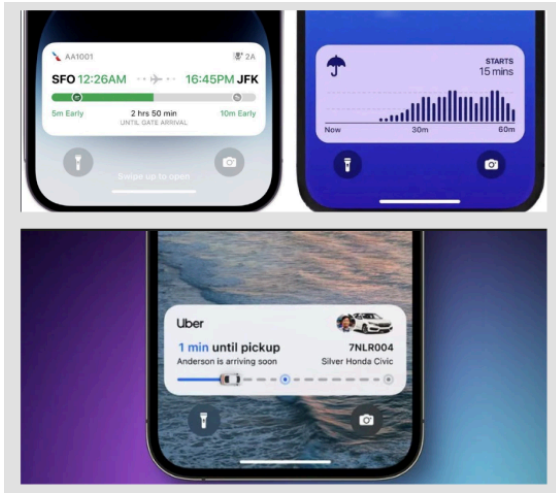


Figure 3.4: Live Activities examples [Maca].

Lock Screen Widget is similar to Live Activities; it is also displayed on the Lock Screen. As compared to the previous feature, Lock Screen Widget is especially useful for users to keep track of essential information on the go. Lock Screen Widgets in iOS have standardized sizes, offering three variants: half-width, quarter-width, and inline. These indicate that the Lock Screen Widget is not designed for complex visualization but rather for displaying essential data, ensuring that the user experience is clutter-free and smooth.

Figure 3.5: Lock Screen Widget examples [Macb].

Considering our scenario, it would be feasible to use either Live Activities or a Lock Screen Widget for displaying the connection status and the device battery level. However, it is noteworthy that Live Activity is essentially designed to represent real-time actions that have an important visualization. While our ECG data stream and server updates can be considered live activities, their visualization is not important for our users. ECG visualization is the only candidate for a Live Activity feature,

but displaying a live ECG graph on a lock screen may be distracting while also using a lot of CPU resources. On the other hand, the Lock Screen Widget offers a more elegant solution in a compact and accessible format.

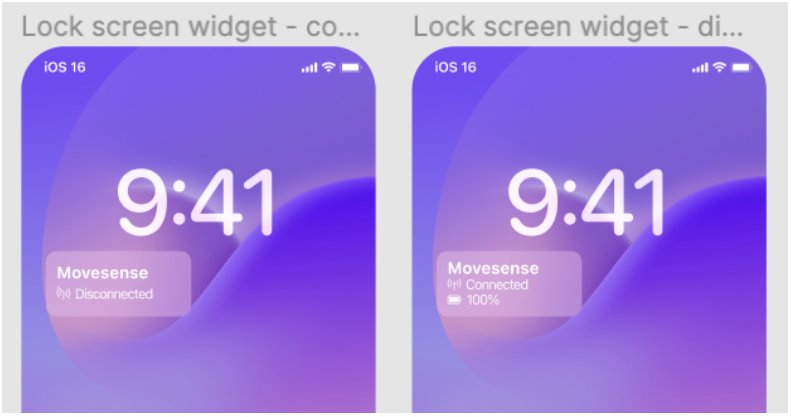


Figure 3.6: EpiHeartMonitor Lock Screen Widget.

The installation procedure for the Lock Screen is not the simplest endeavor a user may undertake. This procedure requires a series of actions that may not be immediately apparent or obvious. As a result, a comprehensive installation guide has been included in the app's profile. This article intends to provide users with detailed instructions for installing the EpiHeartMonitor Lock Screen Widget.The guide should be displayed within the app, appearing for the first time when the user activates the app after connecting a device. Screenshots illustrating the guide flow are shown in Figure 3.7

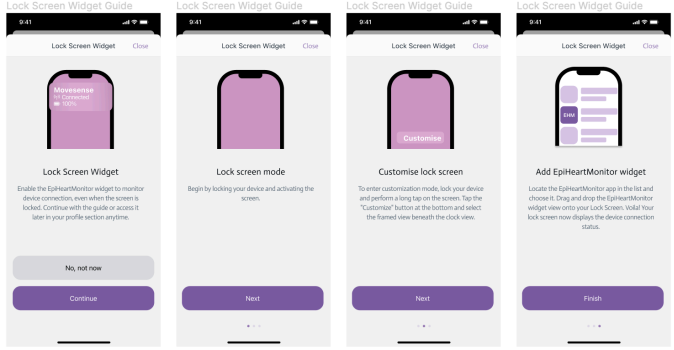


Figure 3.7: Lock Screen Installation Guide.

3.2.6 Final prototype

The final prototype was created through an iterative prototyping process while taking the feedback received from users into account. This method required multiple adjustments to features, sections, styling, and navigation based on user suggestions. The Appendix contains a collection of all the screens(see Figure B.3). The main screens of the application are illustrated in Figure 3.8.

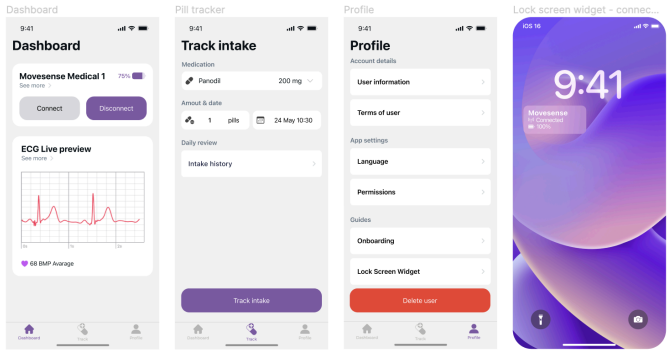


Figure 3.8: EpiHeartMonitor main screens.

CHAPTER 4

Software design

This chapter will address the solution design system, describing aspects such as software architecture and database design. In addition, the modularized approach will be discussed, along with its significance for a modern application promoting scalability, maintainability, and reusability.

4.1 System design

The ApiHeartMonitor application serves a dual role, addressing the needs of both researchers end epilepsy users who want to visualize and monitor their cardiac electroactivity and researchers or professionals seeking to analyze the stored ECG data on the server side. While the design process for the application solution was described in the previous chapter, the current chapter will tackle the technical design aspect of the plugin role of the application.

At a macro level, the technology solution comprises three core components: the wearable device, the mobile application, and the cloud server. These components function within a unidirectional data flow: the device sends data to the mobile application, and the mobile application securely stores the data on the server.

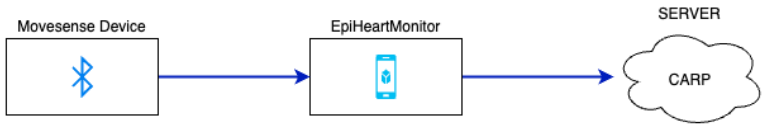


Figure 4.1: System Components Data Flow.

The mobile solution plays a pivotal plugin role, bridging device data to the cloud. It is crucial that it implement a robust server uploading policy while also guaranteeing the integrity, authenticity, and completeness of the received data.

EpiHeartMonitor integrates the CACHET Research Platform (CARP) Web Services API for server upload. CARP Web Services by Bardram is a versatile platform designed to carry out mobile health studies where data is collected from participants smartphones and wearable devices. The system provides a safe hosting infrastructure

managed by the Copenhagen Center for Health Technology (CACHET), ensuring data privacy and protection. There are two main reasons for choosing this framework. Firstly, its robust security policy offers a trustworthy environment for storing sensitive health data. Secondly, CARP Web Services offers flexibility through its file-controller API. The only prerequisite in order to use CARP's service is to have a predefined Study in its system and use the given StudyId when uploading to the server.

By leveraging CARP Web Services, the research project gains a comprehensive, scalable, and flexible platform that offers robust technical support and configuration options.

4.2 Database design

EpiHeartMonitor uploads files in the format of a SQLite database. SQLite is a relational local database system that offers a serverless and configuration-free database engine. Its purpose is to be embedded within applications, allowing the local client to store, retrieve, and manage data efficiently.

There are several reasons that led to the SQLite database solution. One key reason is its compatibility with the CARP Web Service file-controller API. The server solution we chose supports a single file upload. SQLite provides an inherent solution for supporting multi-operation locking mechanisms, dealing with the likely occurrence of race conditions when multiple operations are performed asynchronously. Moreover, considering the various client side data management requirements, such as uploading, saving, and temporary caching before server upload, SQLite is an optimal solution. It fulfills two-fold roles by serving as the file format for server uploads and as a local temporary storage solution for effective data management.

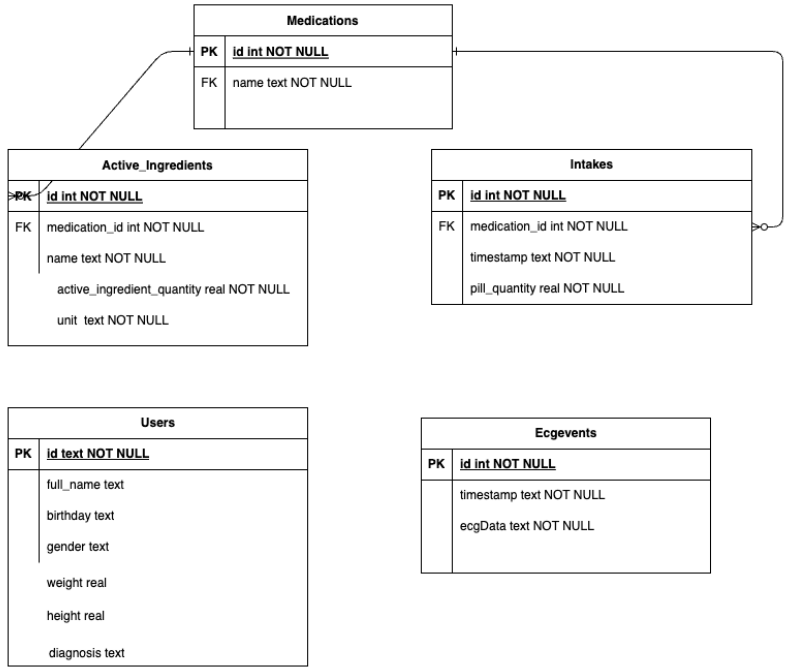


Figure 4.2: SQLite Database Tables.

In Figure 4.2 , the SQLite database scheme is presented. There are five existing tables: *Users*, *Ecgevents*, *Medications*, *ActiveIngredients* and *Intakes*. The *Users* table does not have any relationships as it solely represents the local user in the database. The *Ecgevents* table has two fields: the timestamp and the ECG comma-separated values text. A medication can have multiple active ingredients, this is why its id is a foreign key for the *ActiveIngedients* table. Medication ID is also a foreign key in the *Intakes* table that represents the pill intake records.

4.3 Software architecture

EpiHeartMonitor follows the Model-View-ViewModel software architecture. Model-View-ViewModel (MVVM) is a design pattern that divides the application into three distinct parts: the Model, The View and the ViewModel. The model represents data types that are shaped by the business logic; the View is the user interface, and the ViewModel functions as an intermediary between the Model and the View. The MVVM architecture promotes a distinct separation of responsibilities, improves code organization, and elevates testability and maintainability.

EpiHeartMonitor uses SwiftUI for developing its UI. SwiftUI is a modern UI framework provided by Apple that supports the MVVM pattern. SwiftUI is a declarative, state-driven framework that streamlines user interface development. Despite Apple having not officially provided software architecture recommendations, SwiftUI's APIs such as *@ObservedObject* and *@State*, indicate the adoption of a state-driven architecture.

4.4 Modularization

Within an application, there are multiple functional features and their supporting features such as Bluetooth client and API client. These features, whether functional or supportive, frequently require reuse across different modules, applications, and even platforms to prevent redundancy. To accomplish this, developers aim to modularize these components. However, in the *Swift* environment, available tools, such as static frameworks, target frameworks, and Cocoa Pods, often pose challenges of complexity in configuration, debugging, and lack of intuitiveness.

Fortunately, Apple launched Swift Package Manager in 2018. Swift Package Manager (SPM) is their first-party solution for dependency management that offers a robust system for modularizing project components. SPM streamlines the process by using a *Package.swift* file, where the modules are configured, providing transparency in managing and maintaining its contents. This method ensures a clean and efficient way to handle dependencies. Moreover, with Xcode support, every single component can be built and tested in isolation, further promoting clean code, maintainability and improving code reusability and flexibility.

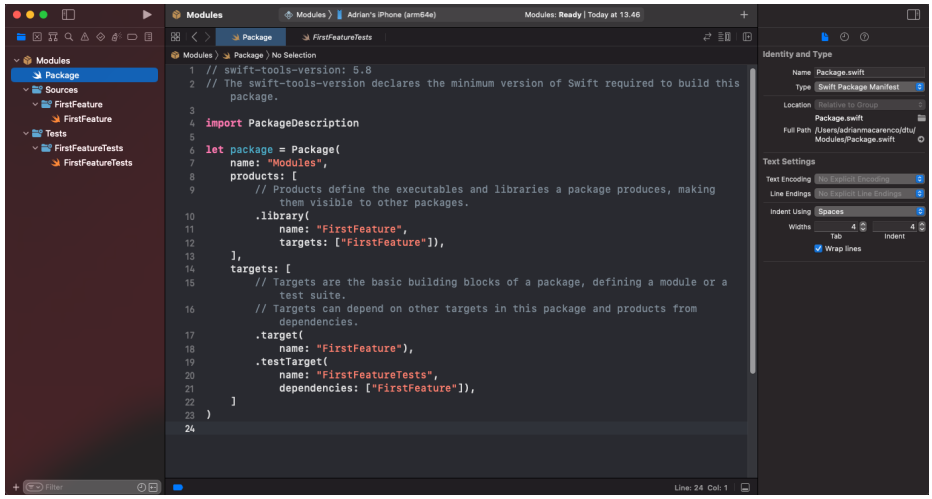


Figure 4.3: Package.swift example.

Furthermore, SPM is also great for external dependency management. If a module requires an external dependency, it can be easily added to that individual module. This offers granular control over modules.

EpiHeartMonitor uses SPM to modularize its components and manage their external dependencies. Figure 4.4 presents its modules, grouping them by their functionality.

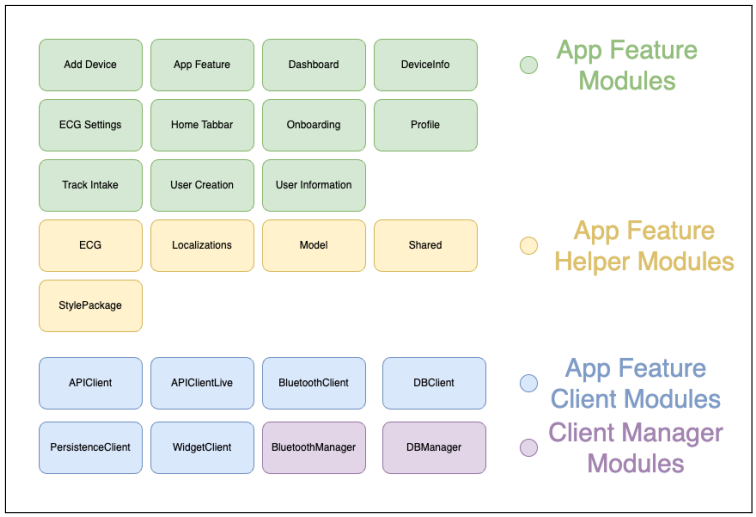


Figure 4.4: EpiHeartMonitor Modules.

CHAPTER 5
Implementation

This chapter describes the technical implementation of the EpiHeartMonitor mobile application described in Chapter 3 and 4. The author will start by explaining the process of integrating the Movesense MD sensor. Subsequently, a detailed description of the SQLite implementation. This will be followed by the CARP integration and then the Lock Screen Widget implementation.

5.1 Movesense API integration

The Movesense MD sensor can be seamlessly integrated into applications by utilizing the Movesense API. The API documentation [Mov] offers detailed information on how to utilize these functionalities effectively, making it an essential reference for developers looking to incorporate Movesense capabilities into their applications. The device offers a wide range of features, including sensor data collection, device configuration, and real-time monitoring.

The device supports BLE technology; therefore, its integration requires the use of Apple's Core Bluetooth framework [Appc]. Core Bluetooth is an iOS Framework provided by Apple to build Bluetooth Low Energy applications that communicate with hardware devices.

5.1.1 MovesenseApi-iOS

Mikko Eronen [Mik] has developed a Swift API that provides an Event-based Swift API for communicating with Movesense sensors through BLE. His implementation utilizes the Core Bluetooth framework [Appc] and provides an extensive range of functionality. The EpiHeartMonitor app requires device scanning, device connectivity, HR and ECG subscriptions, battery level, and system settings. Mikko's implementation fulfills all of the author's needs. The primary advantage of utilizing Mikko's API [Mik] is its support for SPM integration. This allows for easy integration, as discussed in Chapter 4, without the need for a dependency wrapper or other workaround.

Listing 5.1 highlights how is MovesenseApi-iOS [Mik] added as a dependency in the Package.swift SPM file configuration.

```
.target(
    name: "Model",
    dependencies: [
        .product(name: "MovesenseApi", package: "MovesenseApi-iOS")
    ]
)
```

Listing 5.1: MovesenseApi Dependency.

5.1.2 Bluetooth client

Section 4.3 explores the concept of the ViewModel layer, which functions as the holder of the view state and enables interaction with services to modify the state. In this specific scenario, the author designed the bluetooth service interface, named BluetoothClient, to act as a client. This client is injected as a dependency into all ViewModels that communicate with the Movesense sensor. Given the current context, the author designed the bluetooth service interface as a client: *BluetoothClient*. *BluetoothClient* is injected as a dependency into all ViewModels that interact with the Movesense sensor.

Interacting with BLE devices means that the client has to perform work asynchronously. In Swift, there are several approaches to accomplishing this, including the use of completion handlers, observers, and async/await. For the purpose of this project, the author has chosen the async/await API due to its readability and simplicity.

```
public struct BluetoothClient {
    public var isDeviceConnected: () -> Bool
    public var getDeviceBattery: (DeviceWrapper) async throws -> Int
    public var scanDevices: () -> Void
    public var stopScanningDevices: () -> Void
    public var getDevice:(DeviceNameSerial) -> DeviceWrapper?
    public var getDiscoveredDevices: () -> [DeviceWrapper]
    public var getDeviceBatteryPercentage: (DeviceWrapper) async throws
        ↪ -> Int
    public var getDeviceEcgInfo: (DeviceWrapper) async throws ->
        ↪ MovesenseEcgInfo
    public var discoveredDevicesStream: () -> AsyncStream<DeviceWrapper>
    public var connectToDevice: (DeviceWrapper) async throws ->
        ↪ DeviceWrapper
    public var disconnectDevice: (DeviceWrapper) async throws ->
        ↪ DeviceWrapper
    public var disconnectionStream: () -> AsyncStream<MovesenseDevice>
    public var subscribeToEcg: (DeviceWrapper, Int) -> Void
    public var unsubscribeEcg: (DeviceWrapper) -> Void
```

```
16    public var subscribeToHr: (DeviceWrapper) -> Void
17    public var unsubscribeHr: (DeviceWrapper) -> Void
18    public var hrStream: () -> AsyncStream<MovesenseHeartRate>
19    public var dashboardEcgPacketsStream: () -> AsyncStream<MovesenseEcg>
20    public var settingsEcgPacketsStream: () -> AsyncStream<MovesenseEcg>
21    }
```

Listing 5.2: BluetoothClient Interface.

Encapsulating itself within its own module, it ensures that it can be built and tested in isolation. To effectively handle dependencies such as the BluetoothClient, the author used Point-Free's Dependencies API [Poi], which allows the provision of live, failing, and test values based on the context. This is achieved by conforming to the DependencyKey protocol and providing at least the *liveValue*.

```
1  extension BluetoothClient: DependencyKey {
2      public static var liveValue: BluetoothClient {
3          let bluetoothManager = BluetoothManager()
4
5          return .init(
6              isDeviceConnected: { bluetoothManager.isDeviceConnected },
7              getDeviceBattery: { try await bluetoothManager.
                   ↪ getDeviceBattery($0) },
8              // inititializer implementation
9              settingsEcgPacketsStream: { bluetoothManager.
                   ↪ settingsEcgPacketsStream }
10         )
11     }
12 }
```

Listing 5.3: BluetoothClient Services.

5.1.3 Bluetooth manager

When *liveValue* is initialized, *BluetoothClient* extension uses an instance of the *BluetoothManager* class. *BluetoothManager* is a class that interacts directly with MovesenseApi-iOS [Mik]. The manager keeps track of the state of the connected device and executes device commands.

By conforming to the *Observer* protocol defined in MovesenseApi-iOS [Mik], it can capture the Movesense responses that can be grouped in 3 types of responses: **Movesense API Event**, **Movesense Device Event** and **Movesense Operation Event**. Movesense API Event is related to device connection management, such as

discovered, connected, and disconnected. Movesense Device Event is similar to the previous type, except that this event is not caused by a command sent to the device, for example if the connection is lost. Movesense Operation Event is related to subscriptions such as ECG, Heart Rate, Accelerometer, Gyro. Another operation *BluetoothManager* is tasked with is to bridge the completion handler and delegate callback asynchronous work to the new async/await world. It uses async/await continuation [Appb] and async stream continuation [Appa] every time an MovesenseApi-iOS event is received.

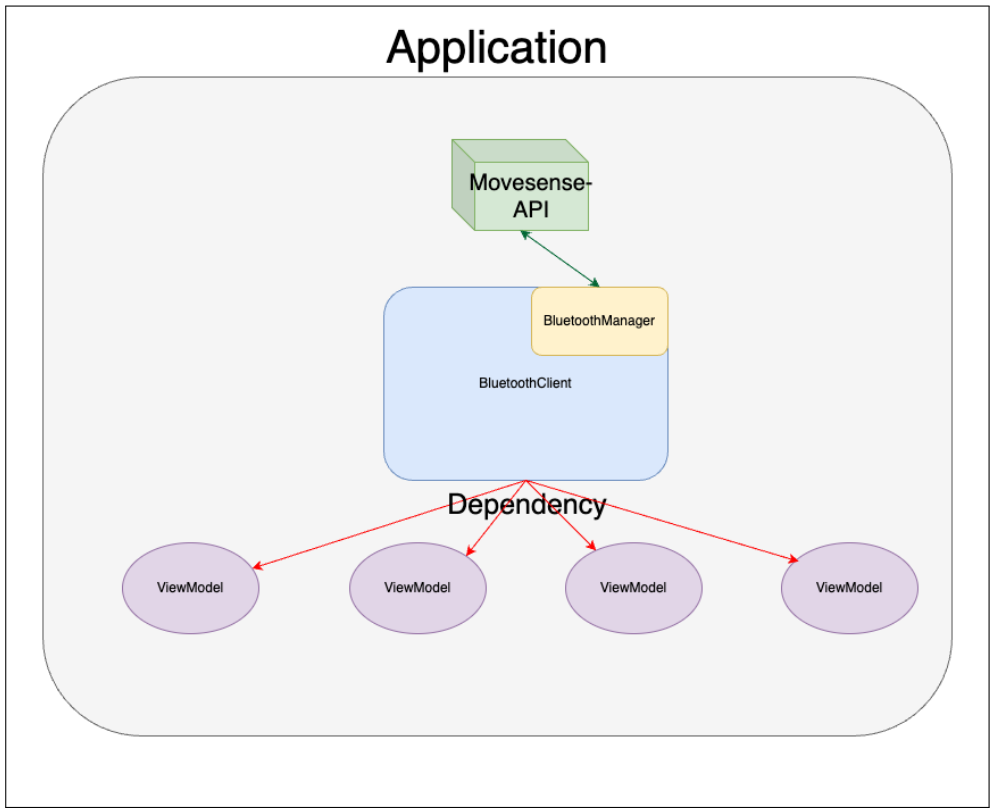


Figure 5.1: Movesense Integration Overview.

Figure 5.1 illustrates end-to-end MovesenseApi-iOS API [Mik] integration. ViewModel communicates whit the BluetoothClient dependency that is injected into it, which subsequently uses BluetoothManager functionality that interacts with MovesenseApi-iOS API.

5.2 Data management

Data management is an important aspect of mobile app development, as it requires stable and accurate handling of sensitive input like personal data and authentication credentials, as well as application-specific data. In the case of the EpiHeartMonitor app, the data can be classified into two distinct categories.

The first category pertains to the research analysis data; it has a deterministic save-upload-clear policy and should be uploaded as a bundle.

The second category consists of data that is primarily related to the app features implementation, including ECG configuration, information about previously connected device, authentication tokens, and more.

To meet these requirements, two different persistent data approaches were implemented. As discussed in Section 4.2, the first solution involves SQLite database, while the second is a custom Persistent Client that saves information in JSON files on the disk.

5.2.1 SQLite implementation

Apple provides an in-built SQLite database implementation for iOS. However, integrating it demands following various steps, as demonstrated in the **Kodeco SQLite tutorial** [Kod]. These steps imply a C API interactor and tedious callbacks, which are not straight-forward and time-consuming for a simple use case such as EpiHeartMonitor. This led the author to explore alternative solutions that could provide a swift implementation suitable for the thesis context.

Several options were available, including FMDB[ccg], Realm[Mon], SQLite.swift[Steb] and more. The author chose the SQLite.swift solution for several reasons. As illustrated in Listing 5.4, it provides a pure Swift interface, ensuring type safety while also supporting optional-aware SQL expression building. It also comes with automatically-typed data access mapping, mapping Swift types directly to their SQLite counterparts.

```
1 private let users = Table("users")
2 private let userId = Expression<String>("id")
3 private let fullName = Expression<String?>("full_name")
```

Listing 5.4: SQLite Swift Interface.

Ultimately, SQLite.swift[Steb] offers developers a versatile solution that has been extensively tested and has a proactive community, ensuring continuous support and development.

The selected library offers a modern dependency integration alternative with SPM. Hence, incorporating it in the EpiHeartMonitor project requires adding just two lines of code in the *Package.swift* file (lines 2 and 9 in Listing 5.5).

```
1    dependencies: [
2        .package(url: "https://github.com/stephencelis/SQLite.swift.git
             ↪ ", from: "0.14.1"),
3    ],
4    targets: [
5        .target(
6            name: "DBManager",
7            dependencies: [
8                "Model",
9                .product(name: "SQLite", package: "SQLite.swift")
10           ]
11       )
12   ]
```

Listing 5.5: SQLite.Swift SPM Integration.

Generally, the SQLite integration overview is analogous to the integration of Movesense presented in figure 5.1. Each one involves identical components: firstly, a client that is injected into the ViewModel and a manager that provides functionality to the client. The manager directly interacts with the dependency library. Client and manager have their own module, this offers a clear separation of responsibilities and enables building and testing in isolation. The author has created a *DBClient* that is injected as a dependency into the ViewModels. ViewModels can use the client functionality to perform database operations, as highlighted in Listing 5.6. The client performs asynchronous work using *DBManager*.

```
1  public class AddMedicationViewModel: ObservableObject {
2      @Dependency (\.dbClient) var dbClient
3      // Implementation
4      func addMedication() {
5          // Functionality logic
6          let medication = try await dbClient.createMedication(
               ↪ medicationName, validIngredients)
7      }
8  }
```

Listing 5.6: DBClient Esage Example.

The DBManager class has a database connection reference that is stored locally. The path of the database file is provided as an environment variable, which has been set in such a way that it is backed up by iCloud, ensuring that users' data is safeguarded and avoiding any potential loss of data.

```
dbBasePath: NSSearchPathForDirectoriesInDomains(
                .documentDirectory, .userDomainMask, true
            ).first!
```

Listing 5.7: Database Environment Path URL Variable.

DBManager behavior is very similar to the REST API. It implements functions that interact with the database using the database connection. It implements operations such as get, create, update, and delete for each of the existing database entities. Listing 5.8 shows create user operation the manager executes.

```
public func addUser(userId: String, fullName: String?, birthday: Date?,
    ↪ gender: String?, weight: Double?, height: Double?, diagnosis:
    ↪ String?) async throws -> User {
    return try await withCheckedThrowingContinuation { cont in
        do {
            let insert = users.insert(self.userId <- userId, self.fullName
                ↪  <- fullName, self.birthday <- birthday, self.gender <-
                ↪  gender, self.weight <- weight, self.height <- height,
                ↪ self.diagnosis <- diagnosis)
            _ = try dbConnection.run(insert)
            cont.resume(returning: .init(
                id: userId,
                fullName: fullName,
                birthday: birthday,
                gender: gender,
                weight: weight,
                height: height,
                diagnosis: diagnosis
            ))
        } catch {
            cont.resume(throwing: error)
        }
    }
}
```

Listing 5.8: Add User DBManager Function.

The function uses **withCheckedThrowingContinuation** closure to bridge the asynchronous work from the completion handler to the async/await world. Then, inside the closure, it tries to execute the insert query using the *dbConnection* instance. This task has to be encapsulated into a *do-try-catch* Swift statement for proper exception handling. Upon successful execution of the operation, the async/await continuation returns a *Void* response. However, if an error occurs during the operation, the continuation throws the corresponding error.

The same approach is applied for the remaining functions related to *users*, *activeIngredients*, *intakes*, *ecgEvents*, *medications* tables.

5.2.2 Persistence Client

Persistence Client is designed the same way BluetoothClient and DBClient were designed. They follow the client pattern, having it's own SPM module. *Persistence Client* is responsible for saving data persistently on the disk. It creates, reads, and writes to specific JSON format files. To fulfill these requirements, a *FileClient* helper type is used. *FileClient* implements the load and save functionalities of a generic *Value* type that conforms to the Codable protocol.

```
public struct FileClient<Value> {

    public var load: () -> Value?
    public var save: (Value?) -> Void

    public init(load: @escaping () -> Value?, save: @escaping (Value?)
        ↪  -> Void) {
        self.load = load
        self.save = save
    }
}
```

Listing 5.9: FileClient Type.

By conforming to this protocol, the client gains the capability to encode and decode a specific object of type *Value* to and from data.

```
private func saveValue<Value>(
    _ value: Value?, forKey key: String,
    in directory: FileManager.SearchPathDirectory = .documentDirectory
) where Value: Codable {
    let url = getDirectoryURL(directory).appendingPathComponent(key)

```

```
    guard let value = value else {
        try? FileManager.default.removeItem(at: url)
        return
    }

    do {
        let data = try JSONEncoder().encode(value)
        try data.write(to: url, options: .atomic)
    } catch {
        print("did not write to url \(url.absoluteString) due to: \(error
            ↪ )")
    }
}

private func loadValue<Value>(
    forKey key: String, in directory: FileManager.SearchPathDirectory = .
        ↪ documentDirectory
) -> Value? where Value: Codable {
    let path = getDirectoryURL(directory).appendingPathComponent(key)
    do {
        let data = try Data(contentsOf: path)
        return try JSONDecoder().decode(Value.self, from: data)
    } catch {
        print("\(#function) No value decoded for key: \(key) error: \(
            ↪ error.localizedDescription)")
        return nil
    }
}
```

Listing 5.10: Load and Save File Client Implementations.

Each function of *Persistence Client* uses a *File Client* implementation, providing a key of type String that is appended to the file's path. This approach ensures that each type of saved information has its own dedicated file. Figure 5.11 showcases the live value of the Persistence Client and the corresponding file name for each implemented type.

```
extension PersistenceClient {
    public static func live(keyPrefix: String) -> Self {
        .init(
            user: .live(key: keyPrefix + ".savedUser"),
            deviceNameSerial: .live(key: keyPrefix + ".
                ↪ savedConnectedNameSerialDevice"),
            deviceConfigurations: .live(key: keyPrefix + ".
                ↪ savedConnectedDeviceConfigurations"),
```

```
            ecgConfiguration: .live(key: keyPrefix + ".
                ↪ savedEcgViewConfiguration"),
            medications: .live(key: keyPrefix + ".savedMedications"),
            medicationIntakes: .live(key: keyPrefix + ".
                ↪ savedMedicationIntakes"),
            apiTokenWrapper: .live(key: keyPrefix + ".apiTokenWrapper"),
            prevEcgUploadingDate: .live(key: keyPrefix + ".
                ↪ prevEcgUploadingDate")
        )
    }
}
```

Listing 5.11: Persistence Client Live Value.

Persiste Client is shared and utilized across View Models in a similar manner as the previous dependency clients. By leveraging Point-Free's [Poi] @Dependency annotation, the same **liveValue** instance can be accessed by the application's ViewModels.

5.3 Lock Screen Widget

Lock Screen Widgets, as their name implies, are small applications that offer easily accessible content that remains visible on the lock screen of the mobile device.

Lock Screen Widgets are available on iOS 16+ and they are developed using the Apple WidgetKit framework [Appg]. WidgetKit is a toolkit that is used to build home screen widgets, Live Activities Widget and Lock Screen for multiple platforms, including iOS, watchOS, iPadOS and macOS.

To specify the widget type, Apple has implemented an enumeration type (Enum) that defines widget's size and shape. The available homescreen types are *systemSmall*, *systemMedium*, *systemLarge*, and *systemExtraLarge*.

The Lock Screen Widget features add three more cases that are associated with this specific widget.

- **accessoryRectangular**

  This type can showcase multiple lines of text or small visualizations. It can be used to display a wide range of information, such as calendar events, their time and location, or a list of items. It has a width that spans half of the screen's width, while the height is fixed. These dimensions make this type the largest variant available.

- **accessoryInline**

  The second case provides a single row of text and an optional image. It is centered on top of the OS time view. Some available implementations are, for example, weather with an appropriate image or the number of unread emails.

- **accessoryCircular**

  The last Lock Screen variant displays its data in a circular view. It's usually used to display metrics, progress views, or gauges. It can be used to display wearable device battery, fitness metrics, or temperatures. On a lock screen, there can be a maximum of four accessoryCircular Lock Screen Widgets.

According to subsection 3.2.6 the author implemented an accessoryRectangular Lock Screen Widget. To add a Widget to the project, a new target called WidgetExtension was added. A widget extension represents a lightweight application that is linked to the main app. It can execute its own code and display the content accordingly. As a result, a new **EpiHeartMonitorWidgetsExtension** widget target was added to the project. Additionally, two files are automatically generated by Xcode: *EpiHeartMonitorWidgetsBundle.swift* and *EpiHeartMonitorWidgets.swift*. By conforming to the WidgetBundle protocol, the first file specifies the view that is used to create the WidgetView. Listing 5.12 highlights its implementation.

```
1 import WidgetKit
2 import SwiftUI
3
4 @main
5 struct EpiHeartMonitorWidgetsBundle: WidgetBundle {
6     var body: some Widget {
7         EpiHeartMonitorWidgets()
8     }
9 }
```

Listing 5.12: EpiHeartMonitor Widget Bundle.

While the bundle file implementation was relatively straightforward, the View file is slightly more complex and requires specific implementations. Prior to delving into its explanation, it is important to present a summary of the widget update cycle and how the EpiHeartMonitor application enables it.

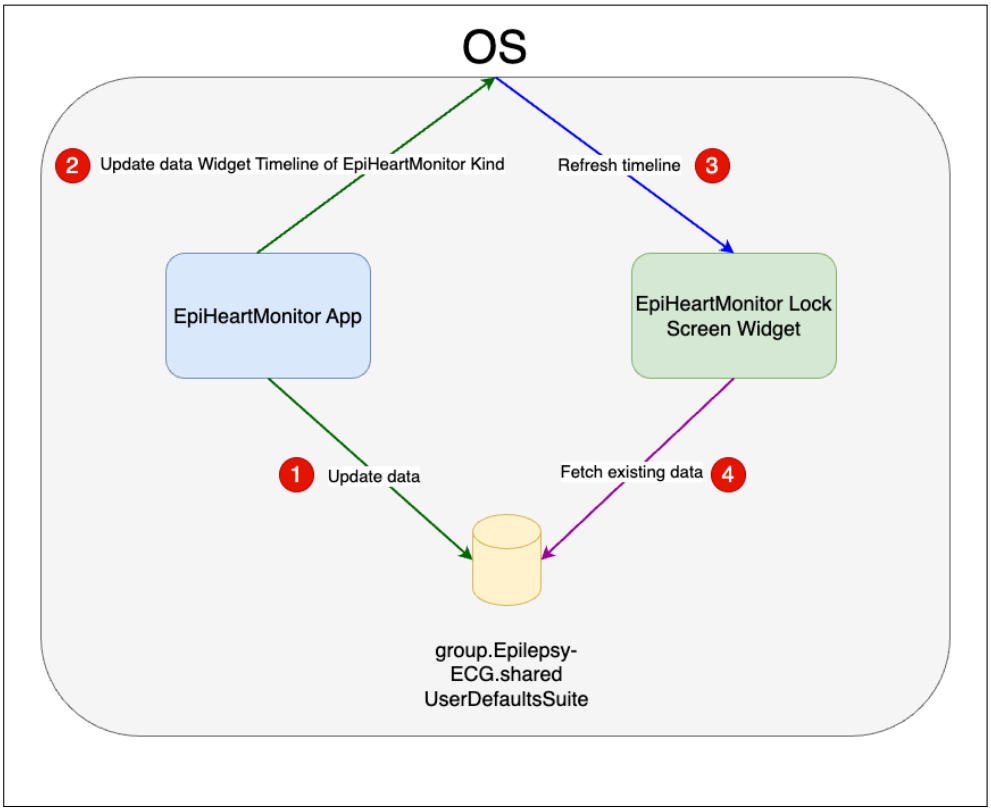


Figure 5.2: Lock Screen Widget Update Flow.

It is important to mention that direct communication between the mobile app and its Lock Screen Widget is not possible. The communication is mediated by the operating system for multiple reasons beyond the scope of this implementation. To establish this indirect communication, a designated target capability named **App Groups** had to be assigned to both application and widget targets. By enabling App Groups with the same *suiteName*, the operating system grants multiple apps developed by the same team access to a shared container. This container provides User Defaults persistence memory access for apps within the same app group. User Defaults is a database where key-value pairs can be persistently stored across the app group.

Figure 5.2 illustrates the update flow of a lock screen widget. Considering a scenario where the app is in the background and has ongoing code execution to handle a device disconnection event callback. Upon receiving the *didDisconnect* event, it is necessary to mirror this state in the Widget view on the Lock Screen. The following

steps outline the flow presented in Figure 5.2:

1. Firstly, the app updated the value of the ”isConnected” key of the UserDefaults suite with the common **suiteName**.
2. Next, it notifies the operating system to update the timeline of the Widget of a given *kind*.
3. Subsequently, the operating system communicates to the specific widget that it should refresh.
4. Finally, the widget fetches the data from the common UserDefaults suite and updates its views accordingly.

For managing the updating and fetching of the connection status and battery level, a *WidgetClient* has been created. *WidgetClient* follows the client pattern the author presented in the previous sections. Having its own modules enables sharing it across the main app target and the widget target. It owns the kind, suiteName, connection, and battery level keys as static properties.

```
public struct WidgetClient {
    static let suiteName = "group.Epilepsy-ECG.shared"
    static let isConnectedKey = "isDeviceConnected"
    static let deviceBatteryPercentageKey = "deviceBatteryPercentage"
    static public let kind = "EpiHeartMonitorWidgets"

    public var updateConnectionStatus: (_ isConnect: Bool) -> Void
    public var updateBatteryPercentage: (_ newValue: Int) -> Void
    public var fetchConnectionStatus: () -> Bool?
    public var fetchBatteryPercentage: () -> Int?
}
```

Listing 5.13: Widget Client Implementation.

To highlight the first and second steps from Figure 5.2, the following calls are needed:

```
widgetClient.updateConnectionStatus(isDeviceConnected)
WidgetCenter.shared.reloadTimelines(ofKind: WidgetClient.kind)
```

Listing 5.14: Update WidgetClient data and reload Widget timeline.

Moving to step number three, the system executes the command by invoking WidgetCenter which reloads the timelines for all configured widgets belonging to the given *kind*. Ultimately, to ensure the widget view is refreshed accordingly, the timeline callback is handled within the *Provider* structure type, which has been automatically generated by Xcode in the EpiHeartMonitorWidgets.swift file.

The Provider delegate function callback that is called when the system reloads the timeline is called **getTimeline**. This function returns the state of WidgetView, specifically, in the thesis project, it is referred to as **EpiHeartMonitorWidgetsEntryViewModel**. The following listing demonstrates the EpiHeartMonitor timeline implementation.

```
1  func getTimeline(for configuration: EpiHeartMonitorStatusIntent, in
       ↪ context: Context, completion: @escaping
2  (Timeline<EpiHeartMonitorWidgetsEntryViewModel>) -> ()) {
3      guard
4          let isConnected = widgetClient.fetchConnectionStatus(),
5          let batteryPercentage = widgetClient.fetchBatteryPercentage()
               ↪ else {
6          let vm = EpiHeartMonitorWidgetsEntryViewModel(isConnected: false,
               ↪  batteryPercentage: -1)
7          completion(.init(entries: [vm], policy: .atEnd))
8          return
9      }
10
11     let vm = EpiHeartMonitorWidgetsEntryViewModel(isConnected:
           ↪ isConnected, batteryPercentage: batteryPercentage)
12
13     let timeline = Timeline(entries: [vm], policy: .atEnd)
14     completion(timeline)
15 }
```

Listing 5.15: EpiHeartMonitor Timeline Implementation.

Whenever this function callback is received, it attempts to fetch the connection status and the battery percentage values (code lines 4 and 5 in Listing 5.16). If either of them is missing (nil), it indicates that they were never set and the default view state should be displayed. Otherwise, the view model is initialized using the existing values and triggers the WidgetView refresh.

5.4 Carp integration

As presented in Section 4.1, the EpiHeartMonitor app integrates the *file-controller* API [CAC] provided by CARP Web Services. According to the API documentation, this endpoint offers flexibility, allowing the authenticated client to upload files in various formats. The request body must contain the file in a multipart/form-data type.

To handle authentication, the author first implemented an AuthenticationHandler that uses CARP system credentials to request an access token. If the token is missing or expired, the solution integrates the PersistenceClient feature introduced

in Section 5.2.2. AuthenticationHandler is an actor type, which is a Swift reference type and is similar to a class; additionally, it offers concurrency safety. To ensure this feature, actors methods that access their data have to be implemented using the new async/await API. The reason AuthenticationHandler type has to be concurrently safe is that the ApiToken data cannot be edited and read at the same time, thus avoiding data races.

The implementation includes a static function named **performAuthenticatedUploadRequest**, which takes an URL request and file path URL as parameters. The function verifies if a valid access token is available. If a valid token exists, it performs the authenticated upload request. However, if the response indicates a 401 error code, it requests a new access token.

```
public func performAuthenticatedUploadRequest(_ request: URLRequest, _
    ↪ fromFile: URL) async throws -> (Data, URLResponse) {
    do {
        let tokens = try await validTokens()
        do {
            let response = try await performAuthenticatedUploadRequest(
                ↪ request, fromFile, accessToken: tokens.accessToken)
            if
                let code = (response.1 as? HTTPURLResponse)?.statusCode,
                    ↪ code == 401 {
                throw URLError(.userAuthenticationRequired)
            }
            return response
        } catch let error as URLError where error.code == .
            ↪ userAuthenticationRequired {
            let newTokens = try await Self.getNewAccessToken(
                getTokenUrl: getTokenUrl,
                carpUsername: self.carpUsername,
                carpPassword: self.carpPassword,
                networkRequest: self.networkRequest)
            self.apiTokens = newTokens
            let response = try await performAuthenticatedUploadRequest(
                ↪ request, fromFile, accessToken: newTokens.accessToken)
            return response
        }
    } catch {
        apiTokens = nil
        throw error
    }
}
```

Listing 5.16: EpiHeartMonitor timeline implementation.

In the absence of a valid token, the authentication handler requests a new access token using its saved credentials. The newly received token is then stored in the Persistence Client and used to perform the upload request.

At the time of implementing the authentication handler, there was no existing API for using the refresh token. As a future improvement, it is recommended to integrate a refresh token mechanism. This would involve getting a refresh token and using it to retrieve subsequent access tokens, resulting in a more optimized process. Additionally, the current system injects static CARP credentials as environment variables. It is recommended to consider alternative methods and potentially integrate CARP's Study APIs.

5.5 ECG Data Transmission Policies

The app was designed with policies for updating the local database and uploading to the server to ensure accurate data transmission between the device and the CARP system. Both of these operations utilize an asynchronous timer-driven cycle. When the user establishes a connection with the device, the cycles begin.

The local database is updated every five seconds by appending the contents of a local array variable. Upon successful completion of the local database update, the local array's elements are removed, and it begins to buffer new elements. This task's implementation can be found in Listing 5.17.

```
// Saves ecg data to the local db every 5 seconds
private func startLocalUpdateTimerTask() {
    dbUpdateTask = Task(priority: .background, operation: { [weak self,
        ↪ clock, localDbUpdateInterval] in
        for await _ in clock.timer(interval: .seconds(
            ↪ localDbUpdateInterval)) {
            do {
                try await self?.updateLocalDbData()
            }
        }
    })
}
```

Listing 5.17: Local database update task.

The data gathered from the ECG device comprises a timestamp and an array of ECG values. However, the provided timestamp is not universal but relative to the device's internal clock. This internal device clock is initialized either during the installation of the device's software or when the device is reset. However, the initial time of the device clock cannot be retrieved, making the timestamp of the device impracticable. To address this issue, a solution was implemented: whenever data is received from the Movesense sensor, an Universal Time Coordinated (UTC)

timestamp is generated, appended to the ECG data, and stored in the local database. The application therefore utilizes its own absolute time for the ECG timestamp.

The task of uploading to the server occurs every five minutes or when the application is opened, if necessary. Once an upload completes, ECG information in the local database is deleted. This strategy of uploading data to the server upon app launch is predominantly implemented due to the fact that users frequently close the app within a five-minute interval. Therefore, the data stored locally must be uploaded the following time the application is launched. It is important to avoid updating the application when it is closed, as the uploading process can take a significant amount of time on slow Internet connections and therefore may fail due to app closing time constraints. Starting the upload when the app launches ensures that the local data will be uploaded successfully. This task's execution is depicted in Listing 5.18.

```
1 private func startServerDbUploadTimerTask() {
2     serverUploadTask = Task(priority: .background, operation: { [weak
          ↪ self, clock, serverUploadInterval] in
3         for await _ in clock.timer(interval: .seconds(
              ↪ serverUploadInterval)) {
4             do {
5                 try await self?.uploadDbToServerIfNecessary()
6             }
7         }
8     })
9 }
```

Listing 5.18: Server upload task.

CHAPTER 6

Validation

6.1 User study

A user study was undertaken to examine and assess the usability and accessibility of EpiHeartMonitor. In this instance, the app was utilized by an individual with epilepsy for a duration of one day, and their feedback was gathered to evaluate the performance of the app.

System Usability Scale (SUS) was utilized to evaluate the application's usability. This scale, comprised of ten straightforward items, provides a comprehensive view of personal usability evaluation.

Due to the difficulty of recruiting individuals with epilepsy who are willing to use the application for an extended period, the user evaluation was limited to a single participant. In addition to presenting the SUS scale, an interview was conducted with the participant. The interview focused on app-specific details rather than the broader evaluation addressed by SUS queries. The user assessed the usability of features such as the ECG streaming visualization, Lock Screen Widget, and Pill Tracker functionality. In addition, the participant was invited to provide enhancement suggestions.

6.1.1 Study setup

TestFlight facilitated the distribution of the application, and the user received a mobile device along with the Movesense MD device. The user was provided with detailed instructions on how to mount and operate the wearable device.

6.2 Data analysis study

A study focusing on data analysis was carried out to assess whether the data uploaded by the app to the CARP system maintains the integrity, authenticity, and completeness of the received data.

6.2.1 Study setup

For this study, the Xcode IDE was utilized with debugging prints, the Proxyman app was employed to monitor the phone network traffic, and the Postman application was utilized to check the data sent to the server.

In order to analyze the ECG data, the author added debug print commands to the project's codebase to observe the raw data transmitted by the Movesense device. Each time the EpiHeartMonitor app receives data from the device, a timestamp is allocated, and the data is stored in a SQLite database. Later, this information is uploaded to the server. The additional print statements were helpful for contrasting the raw data received from the ECG device with the data stored in the SQLite database file transmitted to the server.

```
1  private func subscribeToEcgStream() {
2      for await ecgData in bluetoothClient.dashboardEcgPacketsStream()
           ↪ {
3          let newRecording = (ecgData, Date())
4          print("  \(newRecording.1.description) \(newRecording.0.
               ↪ samples.commaSeparatedString())")
5          ecgDataEvents.append(newRecording)
6      }
7  }
```

Listing 6.1: ECG raw data print.

In addition, print statements were incorporated to monitor the local data updates and the completion of the server upload task. The addition of these print statements to Xcode made it easier to comprehend the values and timing of particular actions, such as local data storage and server update procedures.

```
1  private func updateLocalDbData() async throws {
2      let localData = ecgDataEvents.map { ($0.1, $0.0.samples.
           ↪ commaSeparatedString()) }
3      try await self.dbClient.addEcg(localData)
4      print("Update local data  \(Date().description)")
5  }
6
7  private func uploadDbToServer() async throws {
8      print("Update server started  \(Date().description)")
9      try await apiClient.uploadDbFile()
10     try await dbClient.clearEcgEvents()
11     print("Update server completed  \(Date().description)")
12
13 }
```

Listing 6.2: Local update and server update tasks prints.

As described in the Implementation section, once the ECG data is transmitted, both the local ECG storage and server uploading processes occur in a recurring cycle. The server uploads occur every five minutes, while the local updates occur every five seconds. Nevertheless, for the purposes of this validation, the author modified the upload assignment interval. It was considered impracticable to verify a cycling action by waiting for such a lengthy duration. Therefore, the server upload interval was changed to fifteen seconds. Consequently, local database modifications occur every five seconds during the validation procedure, while server uploads occur every fifteen seconds.

The following tool was employed: Proxyman, an effective proxy application that facilitates the monitoring of mobile internet traffic. Proxyman enables the monitoring and analysis of network requests. In order to validate the project, it was necessary to examine the files sent from the app to the server. To facilitate this, the proxy configuration was configured on both the mobile device and the computer in accordance with the documentation provided by Proxyman[Pro].

Finally, the author utilized Postman. Postman is used in this context to facilitate various API testing and request handling. Specifically, it was used to verify the existence of uploaded data on the server and ensure that it corresponds to the identity created by the application during the upload process. This validation phase was required for every upload to ensure data consistency and accuracy.

6.3 Accessibility study

The author assessed the app's accessibility to specific UI properties, Voice Over and font size, using the app itself. In addition, a custom-designed accessibility testing app was employed. Stark, an application that can be incorporated as a plugin in Figma, was utilized to evaluate the accessibility scores of UI elements. The findings from this study will be presented in a subsequent chapter.

6.3.1 Voice over

Voice Over, an essential step in reviewing mobile applications accessibility [Har], was thoroughly tested by the author. This accessibility feature can be enabled by accessing Settings -> Accessibility -> Voice Over -> Switch on Voice Over in the Settings application of the phone.

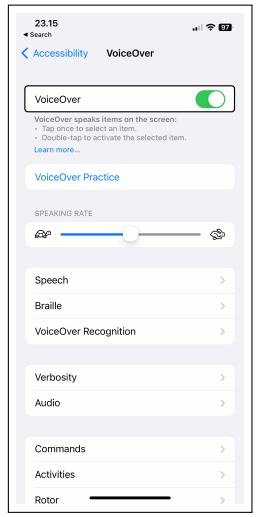


Figure 6.1: Voice Over Setting.

Enabling this setting automatically activates the Voice Over feature for all UI elements that have implemented this feature. A validation of the EpiHeartMonitor app has been done, and the actions have been recorded. The resulting video showcasing the Voice Over testing can be found at the **following address**.

6.3.2 Dynamic font type

The Dynamic Type feature on iOS is an important accessibility function that allows users to customize the font size of the user interface according to their preferences. Similar to Voice Over, the Dynamic Type options can be set through the device's Settings app. The font size can be adjusted at the system level by following the path: Settings -> Accessibility -> Display Text Size -> Larger Text Size, or on a specific app level by changing, Settings -> Accessibility -> Per-App Settings -> Add App -> Larger Text.

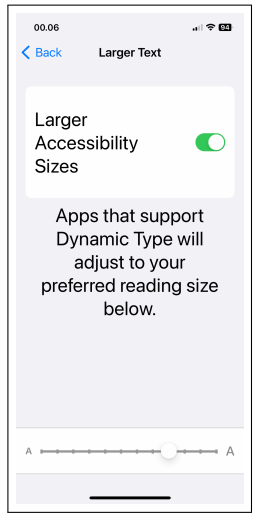


Figure 6.2: Dynamic Type Setting.

Activating this function automatically adjusts the font dimensions for applications that support it. The author conducted a validation of the EpiHeartMonitor application, capturing screenshots of its main sections.

6.3.3 UI Controllers' Contrast

Stark is a tool for performing design product accessibility audits. It evaluates various accessibility components, with particular focus on the application's contrast, which is essential for ensuring accessibility, ethics, and inclusivity. With Stark integrated as a Figma plugin, designers can easily select and evaluate particular UI elements for accessibility.

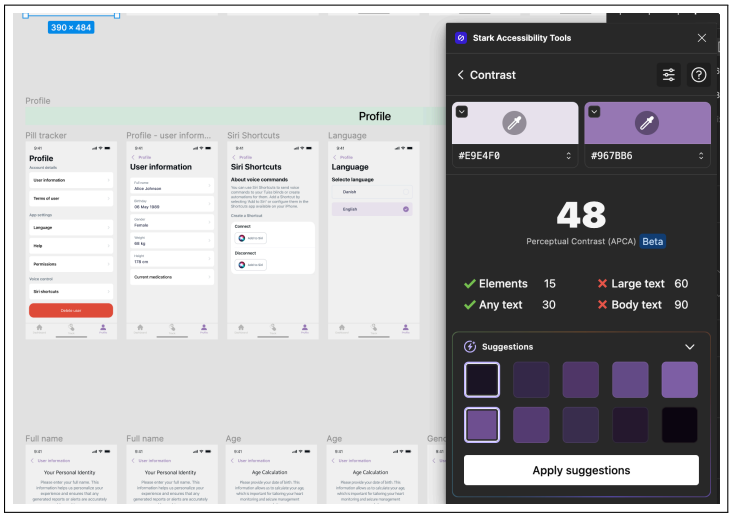


Figure 6.3: Stark Figma Plugin.

The author conducted validation of UI controllers, including buttons, selectors, and pickers, to assess their contrast. The resulting Stark scores were registered and will be addressed in Chapter 7.

CHAPTER 7

Results and Discussion

In this chapter, the outcomes of the studies outlined in Chapter 6 will be presented. Additionally, a comprehensive discussion of these findings, along with the system's strengths, limitations, and potential future improvements, will be addressed.

7.1 Analysis of Studies

7.1.1 User study

It was requested that the participant completes the SUS questionnaire. Using the following formula, an overall rating was calculated based on the scores the user assigned to the application.

$$\text{SUS Score} = \left(\sum_{i=1}^{10} \text{Score Contribution} \right) * 2.5 \tag{7.1}$$

where

$$\text{Score Contribution} = \begin{cases} \text{User response value} - 1 & \text{for odd-numbered items} \\ 5 - \text{User response value} & \text{for even-numbered items} \end{cases}$$

By applying the formula to the questionnaire responses presented in B.4, the score assigned to the application by the participant is 75 points.

Grade	SUS	Percentile range	Adjective	Acceptable	NPS
A+	84.1-100	96-100	Best Imaginable	Acceptable	Promoter
A	80.8-84.0	90-95	Excellent	Acceptable	Promoter
A-	78.9-80.7	85-89		Acceptable	Promoter
B+	77.2-78.8	80-84		Acceptable	Passive
B	74.1 – 77.1	70 – 79		Acceptable	Passive
B-	72.6 – 74.0	65 – 69		Acceptable	Passive
C+	71.1 – 72.5	60 – 64	Good	Acceptable	Passive
C	65.0 – 71.0	41 – 59		Marginal	Passive
C-	62.7 – 64.9	35 – 40		Marginal	Passive
D	51.7 – 62.6	15 – 34	OK	Marginal	Detractor

Table 7.1: SUS Grading Scale [Jef].

According to the data presented in Table 7.1, the EpiHeartMonitor app's usability score lies within the range of a B grade, which corresponds to a score between 74.1 and 77.1. To be noted here, is the fact that the average SUS score is 68, and that scores above 80 are considered as exceptional [Mas]. Consequently, the application's usability score is above average and close to an exceptional rating, signifying a highly favorable evaluation.

In addition to the SUS evaluation, the author conducted a comprehensive interview in which the user was asked more specific questions regarding the EpiHeartMonitor app. This interview's full transcript is presented in Section B.2.1.

The general app usability feedback was very positive, the user found the app easy to use and intuitive, despite being an Android user. The feedback regarding the app's usability was very positive. The user found the app to be intuitive and simple to navigate, despite being an Android user.

The user commended the Lock Screen Widget, describing it as highly practical and useful for recording ECG data in the background. He particularly liked the design's clarity and simplicity.

Additionally, he had positive feedback about the ECG visualization, appreciating the lucidity it provided in monitoring his pulse. However, he proposed an improvement: a continuous visualization in which the signal graph always appears at the end, arguing that this would be easier to follow than the current model, which resumes from the beginning whenever it reaches the end of the graph. He also called attention to the ECG's sensitivity to noise, implying that frequent movement introduced some disturbance to the readings.

He found the pill intake feature to be beneficial, especially from an epilepsy management standpoint, acknowledging the fact that he frequently forgets whether or not he has taken his medication.

Nonetheless, he emphasized the usefulness of the app's guides. Specifically, he was able to install the Lock Screen Widget by following its instructions, a notable accomplishment given the inherent complexity of the implementation process.

7.1.2 Data study

The author validated the data using the instruments described in Section 6.2. The study's conclusions are presented in this subsection.

In order to validate this task, there are three fundamental data properties that must be observed:

- **Data authenticity** - the values of the data received from the Movesense Device remained unaltered.

- **Data integrity** - there is no missing data, which could have been caused by numerous operations or other concurrent asynchronous database tasks.

- **Continuity Across Multiple App Usage Cycles** - similar to data integrity, but it extends beyond a single recording session per app opening. Locally preserved data, which was not updated to the server due to app termination, is effectively sent to the server upon app relaunch.

The author utilized the Xcode outputs, which contained raw ECG data values, and compared them to the data stored in the database file, which was then transmitted to the server. If the server responded with a successful transfer confirmation, we could assume that the file sent with the request was effectively stored on the server.

The data analysis followed the session lifecycle of an application. Opening the application, establishing a connection with the device, and initiating ECG data streaming. Following two cycles of data uploading to the server and local data saving tasks, the author terminated the application. The app was then restarted and evaluated to determine whether the data, which had been saved locally prior to app termination, was effectively uploaded to the server.

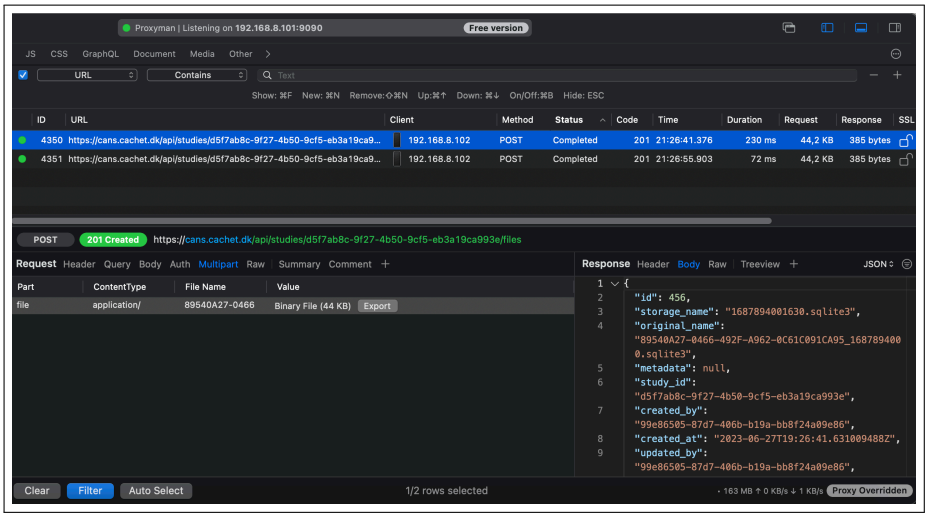


Figure 7.1: Proxyman EpiHeartMonitor file upload CARP requests..

As appears in Listing B.1, two successful requests were sent to the CARP Web Services API, which corresponds with the Xcode server upload task logs.

The author extracted the values from the files included in each request and opened them in a text editor application. The author began by locating the very first values from the Xcode records, which should correspond to the corresponding data in the database file. Both the timestamp and ECG values were discovered to be identical, validating the data's authenticity. The upper window of Figure 7.2 displays the Xcode logs, while the lower window displays the file sent to the CARP system.

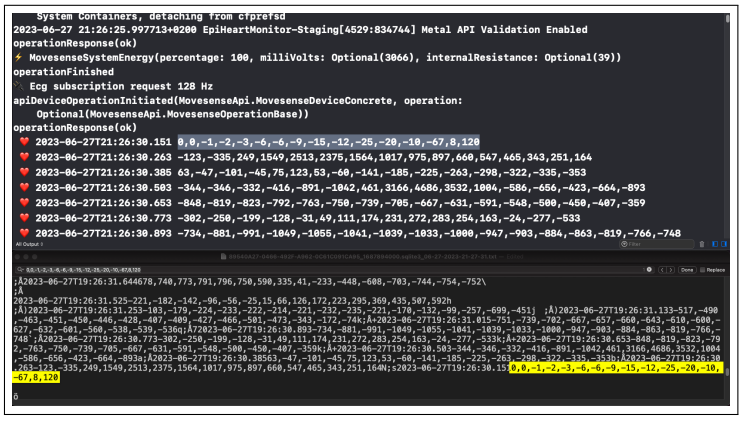


Figure 7.2: Data authenticity validation..

The author then proceeded to validate the integrity of the data by inspecting the data received by the app during the uploading task's operation. Accessing or modifying a file for uploading should not result in the loss of data, so it was crucial to analyze this particular data. In order to investigate this, the author centered on the initial data uploading task and localized the received data. This information should not have been present in the initial database file, as it was already being sent to the server. However, it should not be absent in the second file. This was determined by searching for ECG value records between the "upload task started" and "upload task completed" logs. Figure 7.3 displays Xcode logs in the top window, with the first uploading task file in the bottom left corner and the second uploading task file in the bottom right corner.

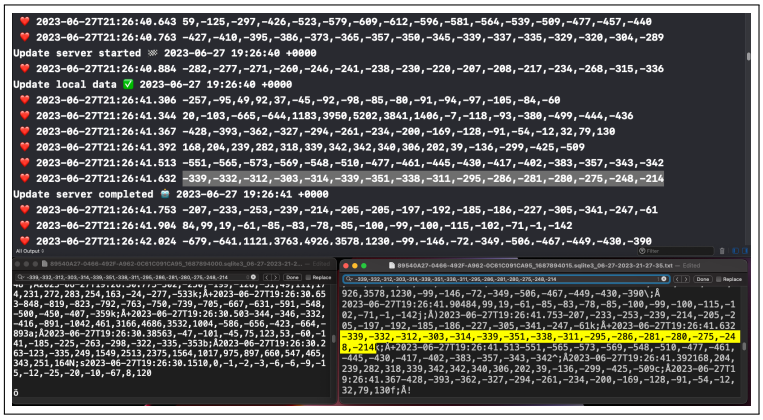


Figure 7.3: Data integrity validation..

The author then tested whether locally stored data was sent to the server upon reopening the application. Only the values that weren't saved locally should have been absent from the sent file when the application was reopened. In order to conduct this test, the application was reopened, which triggered a new API call, which revealed that locally stored data had not been synced with the server. The file that was sent when the application was accessed was then investigated using a text editor. It was discovered that the data stored locally but not sent to the server, as indicated in the Xcode records, could be found in the cloud-sent database file. In addition, the data that was printed in the logs but not preserved locally was indeed absent, further validating the test's success.

Figure 7.4: Data continuity validation.

The upper window of Figure 7.4 displays the Xcode logs, while the lower window displays the third file sent to the CARP system.

The saved file data was then tested using Postman and a GET endpoint designed to retrieve the stored data. Unfortunately, this endpoint did not return the actual stored data; rather, it returned only the generic information saved during the upload request. Despite this, the author was able to effectively validate and verify each of the three upload task requests.

This study ultimately enabled the author to validate the desired data properties, namely authenticity, integrity, and continuity, across multiple app usage cycles. It is essential to note, however, that one aspect of the validation procedure involved reducing the time between uploading jobs from five minutes to fifteen seconds.

7.1.3 Accessibility study

7.1.3.1 Voice Over

As presented in Section 6.3, the author conducted validation tests on the application's accessibility features. The first test focused on evaluating the Voice Over app's capabilities. A screen recording demonstrating the application behavior throughout this test can be accessed at the following **link**.

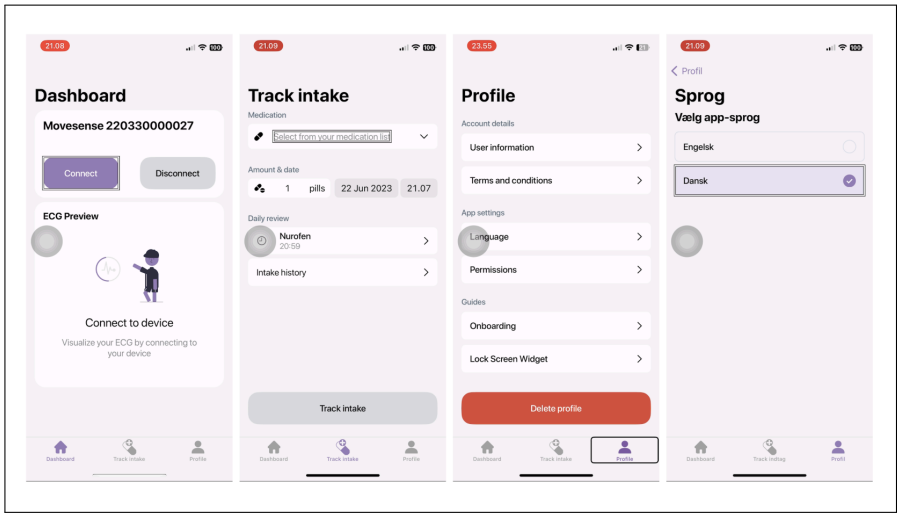


Figure 7.5: Voice Over EpiHeartMonitor main screens.

EpiHeartMonitor integrates the Voice Over functionality for every button and label, ensuring accessibility for users with visual impairments. Furthermore, it provides audio feedback for activities related to device connection and disconnection.

Once the device is connected, the battery level is assessed, and the battery percentage is announced to the user. Similarly, the app communicates the average heart rate measurement through sound. These features are vital in providing essential information to the user. The EpiHeartMonitor feature is active only if Voice Over is enabled in the Settings app on the phone.

7.1.3.2 Dynamic Type font

In terms of Dynamic Type font sizes, it is essential to consider the recommendations given by Apple's Human Interface Guidelines [Appd]. According to these guidelines, it is recommended to support a minimum font size of 17 pt for the body text. Research indicates that more than 40% of iOS users adjust their font size to better suit their requirements[Kri]. Noteworthy applications including AirBnb [Noa] and PDF Viewer [Stea] have analyzed their individual statistics, which reveal that around 30% of their users prefer a personalized font size. These findings emphasize the significance of diverse font size support to ensure optimal accessibility for all users.

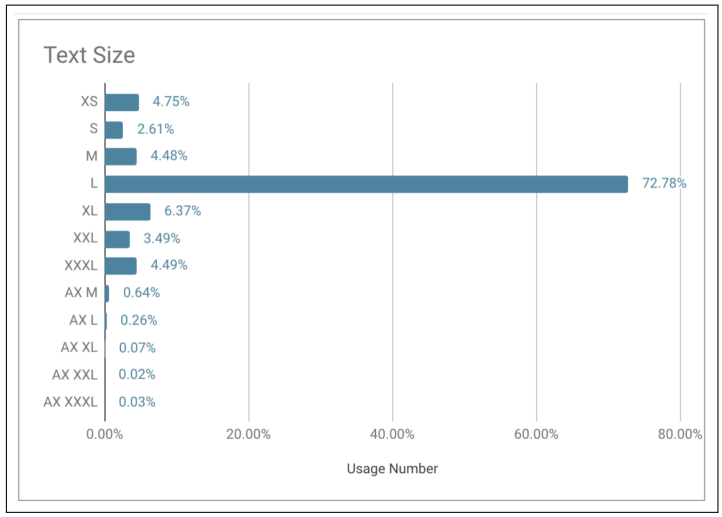


Figure 7.6: User Dynamic Type adoption of PDF Viewer [Stea].

As illustrated in Figure 7.6 PDF Viewer app users supporting up to Extra Extra Extra Large Content Font (XXXL) size covers 98,88% of their users preferences. Raising support to Accessibility Medium (AX M) would increase the percentage by 0,64 making a total of 99.52%.

Considering PDF Viewer user adoption, EpiHeartMonitor has been tested for AX M font.

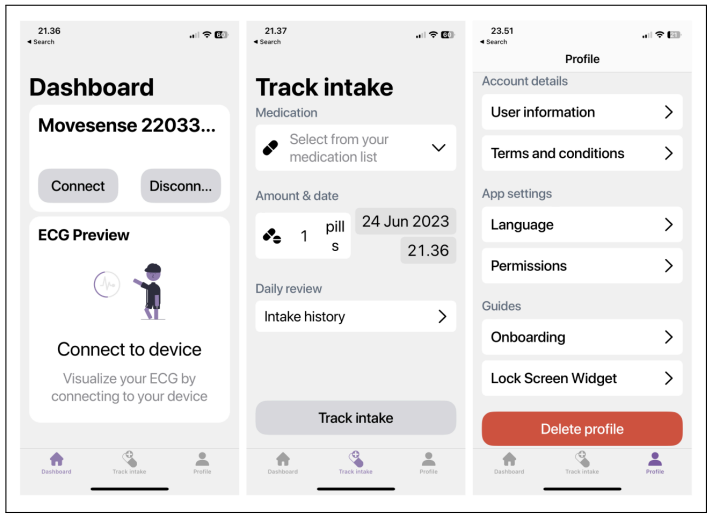


Figure 7.7: EpiHeartMonitor Accessibility Medium Dynamic Type.

As shown in Figure 7.7, the EpiHeartMonitor labels demonstrate clarity and readability, meeting the accessibility requirements of around 99% of users, according to Figure 7.6. This result signifies a positive outcome for the EpiHeartMonitor accessibility features.

Furthermore, by enlarging the font size options to include two additional size units, reaching up to Accessibility Extra Large (AX XL), the coverage can be extended to cover around 99.95% of users, as evidenced by the findings from PDF Viewer users. The outcomes are illustrated in Figure 7.8,where some labels may appear slightly compressed but remain easy to read for users.

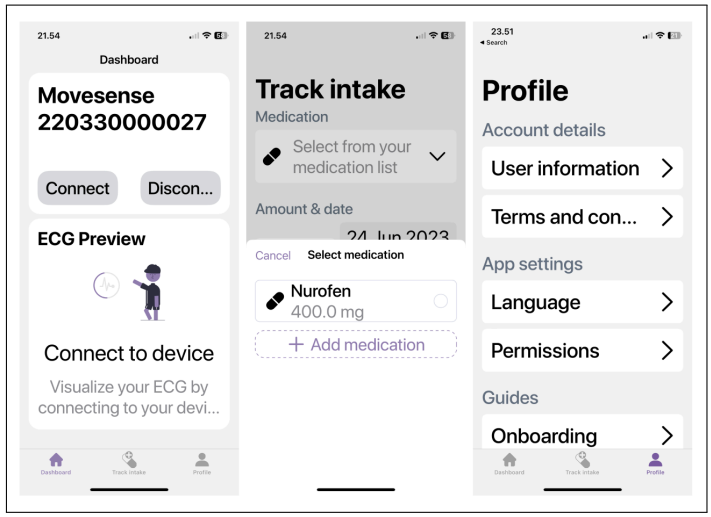


Figure 7.8: EpiHeartMonitor Accessibility Extra Large Dynamic Type.

7.1.3.3 UI Controllers' Contrast

To evaluate the accessibility contrast of the application, the writer utilized the AA and AAA compliance levels according to the Web Content Accessibility Guidelines (WCAG) 2.0 standards. These standards establish the fundamental requirements for contrast ratios between background colors and text to ensure a great experience for all users, including those with colorblindness and cognitive or neurological disorders.

The levels of priority for AA and AAA are as follows [Sta]:

- Single A is the minimum level of requirement that all digital products must implement. This is essential while simultaneously expresses the minimal level of attention an author could dedicate to this aspect.

- Double A (AA) requires a contrast ratio of at least 4.5:1 for normal text and 3:1 for large text. This is an acceptable level for most of digital products.

- Triple A (AAA) necessitates a contrast ratio of at least 7:1 for normal text and 4.5:1 for large text. AAA is a gold standard level of the contrast accessibility. It marks the edge between a good and an excellent inclusiveness experience.

Large Text is defined as text that is at least 18 pt or 14 pt and bold for regular text.

Two key UI components, namely the primary and secondary buttons, were selected for the evaluation test. These components were picked based on their frequent appearance and significance as fundamental user interactions within the application.

The primary button component of the EpiHeartMonitor app has a purple background with a hex color value of #7C5AA4. The button text, displayed in 17 regular SF Pro font in white color, is categorized as Normal text. To evaluate the accessibility of this button, the Stark plugin was utilized, and the results are depicted in Figure 7.9.

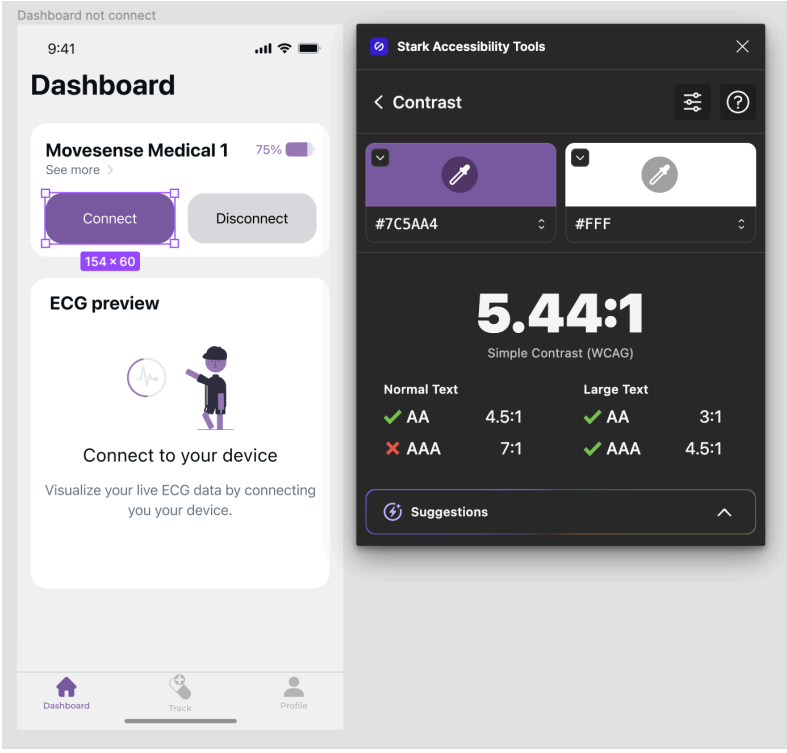


Figure 7.9: Primary Button Contrast Result.

The result highlights its conformance to the AA of WCAG standard. This level of compliance is commendable and aligns with the standards observed in many modern

digital applications. The text size is nearly meeting the criteria for being labeled as Large Text (18 pt), falling just below the AAA standard.

The secondary button component of the EpiHeartMonitor app has a gray background with a hex color value of #D8DADC. The button text, displayed in 17 size regular SF Pro font in black, is categorized as Normal text. The following 7.10 Figure showcases its contrast results.

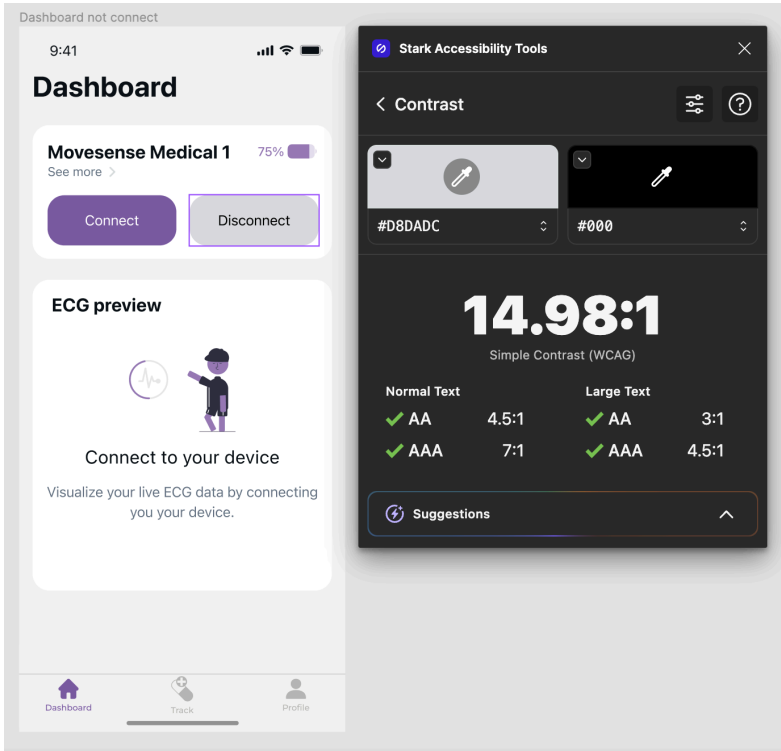


Figure 7.10: Secondary Button Contrast Result.

The second evaluation result is better, as it complies with the AAA WCAG standard. This level is considered the gold standard in terms of accessibility.

7.2 Discussion

In this section, a thorough discussion on utilization will be presented, emphasizing the project's strengths, limitations, and areas for improvement. The objective is to critically assess the findings and insights gathered throughout the studies, highlighting various aspects of the application's effectiveness and performance.

7.2.1 Strengths

The strengths of the application are discussed in this subsection. These are mainly obtained from the findings of the studies presented in Section 7.1 .

- In the context of the EpiHeartMonitor application, one of its main strengths is the inclusion of the Lock Screen Widget, which reflects the device connection status. User feedback from User Study 6.1 emphasized the importance of this feature, stating that it is easily accessible and practical for supervising the connection status.

  Notably, while the feedback originated from an user from epilepsy management target group, the potential benefits of this feature extend to a wider user range, including researchers using the app for their investigations. The importance of providing research participants with real-time feedback regarding device connectivity holds significant value.

- The user study highlighted another advantage of the app: its clarity, with well-separated app sections. The ECG feature, which is the app's main feature, stands out for its accessibility and user-friendly adjustments. It presents a convenient preview on the Dashboard screen, ensuring effortless accessibility for users.

- In the second analysis, the application showcased its aptitude to respect data integrity, authenticity, and completeness. As shown in Data Study (Section6.2), the challenge of acquiring data at 128 Hz or 256 Hz frequencies while persistently saving it on both the device and the server was successfully handled by the application.

- The EpiHeartMonitor application demonstrated a versatile adoption of accessibility features, as shown in Section 6.3. It aptly incorporates the Voice Over feature, which enables users with visual deficiencies to browse and engage with buttons and labels. Furthermore, it supports Dynamic Type font sizes, accommodating users who favor customized font modifications. These functionalities collectively improve the accessibility and user-inclusiveness of the application, ensuring a great experience for a diverse user base.

7.2.2 Limitations

There are several limitations in terms of development and evaluation that need to be acknowledged and may result in concerns or unpredictable outcomes:

- Throughout the User Study phase (Section 6.1), the application was tested by a single user as a result of difficulties in recruiting participants with epilepsy. This constraint of having only one user for testing may introduce potential biases and not provide a comprehensive representation of the user base, leading to potentially misleading conclusions.

- One more limitation of the app is its exclusive support for the iOS platform. As a result of this limitation, the testing and usage of the app were restricted to iOS users only. This limitation can pose difficulties when taking into account both epilepsy and researchers users.

- The present Lock Widget implementation has a limitation. While conducting tests, the author came across a bug related to the widget's state when the app is closed by the system. While the widget correctly reflects the connectivity status when the app is terminated by the user, there is a rare situation where the application can be stopped by the system. This termination can occur unexpectedly and is influenced by different factors like system memory and CPU usage. In such a scenario, the widget fails to indicate that the connection has been lost.

  Unfortunately, there is no straightforward way to handle this event because the application does not have a callback, particularly for this system-initiated termination.

- Currently, the system is restricted in that information is sent to the server at a predetermined and unchangeable rate. This is a limitation for researchers who need greater flexibility in receiving data pace according to their research findings. The current rigid updating system calls an upgraded solution to tackle this constraint.

7.2.3 Improvements

There are several areas within the application that could benefit from enhancements. The subsequent considerations should be addressed:

- To overcome the limitation of not being able to control the server upload rate, the implementation of remote control integration for server upload rate is proposed as a solution. This solution eliminates the previous restriction that prevented researchers from controlling the rate at which data was uploaded to the server.

  Utilizing Firebase Remote Configuration is a viable implementation option for this solution. This method retrieves the current configured uploading interval from the remote configuration when the application initiates. When a group of researchers collaborating on a project can coordinate to achieve the intended value, this solution is effective. However, if multiple researchers require varying upload timeframes, the system may need to be modified to accommodate multiple configurations for various research requirements.

- A potential workaround can be implemented to solve the problem of the Lock Screen Widget being terminated by the system. Despite the fact that this solution is nondeterministic, it provides a method to address the existing issue.

By enabling remote fetch capabilities, an application can execute a section of code even after it has been terminated. A validation can be conducted within this code block to verify any connected devices. If BluetoothManager detects no connected devices, the Lock Screen connection status can be changed to signify disconnection.

The proposed workaround has its own limitations. The developer can specify a minimum interval for the execution of the code sequence during configuration. However, setting a minimum interval does not guarantee that the code will execute precisely at that time. The operating system determines the actual execution time based on app usage, resources available, and other variables. This method offers a potential solution to the termination problem and maintains accurate connection status information on the Lock Screen Widget.

- A potential uploading capability enhancement is mutually exclusive with the previously suggested improvement 7.2.3. Using a dedicated streaming service to transmit the data in real-time without predetermined time intervals is another alternative. However, it is essential to consider whether live transmission is required to meet the requirements of the researchers. In addition, the implementation of this feature may have a significant influence on battery life, and the CPU utilization should be studied thoroughly. Before implementing this feature, a comprehensive evaluation of the prospective benefits and drawbacks is required.

- Improving data transmission security is essential. The application currently transfers a SQLite database file to the CARP system. It is important to note that SQLite files can be readily interpreted using a code editor, as depicted in Section 6.2. This exposes a security flaw, as the data is not presently sufficiently secured. Encrypting the data prior to its transmission to the server could be a potential solution to this problem, ensuring its confidentiality throughout the process.

- The Movesense API provides a variety of data endpoints, including ECG, accelerometer, gyroscope, and temperature. While the project's primary focus is on ECG data, it is reasonable to consider incorporating accelerometer data for future research analysis. It is essential to note, however, that this integration may affect the app's resource optimization. Before implementing this feature, a comprehensive analysis of its resource impact should be conducted.

- The application's structure is extremely modular, allowing its modules to be built and tested independently. This architecture provides ample opportunity for additional unit testing. This can substantially improve code quality and code stability.

CHAPTER 8

Conclusion

To conclude, an examination of the original research questions that formed the basis of our premise will be presented.

- **What is the UX design that encourages epilepsy people to use a mobile application that streams ECG data? What are the main needed features the application must have?**

  The UCD iterative design methodology was used to develop the UX design of the application. Continuous user feedback influenced the design process, enabling the app's refinement and development. The two distinct user groups, epilepsy patients and researchers, contributed important insights that shaped the application's common features. The customization options for ECG and the server uploading functionality were influenced by the feedback of epilepsy patients, whereas the medication intake tracking was influenced by the feedback of researchers. The Lock screen widget was identified as a requirement shared by both user categories, thereby enhancing the user experience overall.

- **How can these features be integrated into a mobile app with real-time functionality?**

  Using modern Swift APIs, the aforementioned features were integrated into an iOS application. As a result of using async/await to handle asynchronous duties, ECG data could be updated locally and on the server with greater efficiency. The Lock Screen Widget functionality was implemented using the SwiftUI framework, which provided a streamlined and intuitive user interface. In addition, the project's modularization strategy enabled flawless functionality sharing between the main app target and the Lock Screen Module, thereby ensuring efficient code organization and maintenance.

- **What are the main usability and accessibility requirements for ECG control and monitoring mobile application for epilepsy patients and how can these be implemented into the system design?**

  Following user interviews and iterative design, the essential usability and accessibility requirements for the ECG control and monitoring mobile application for epilepsy patients were identified. Accessibility features such as Voice Over [Appe] and Dynamic Type [Appf] were recognized alongside usability features such as the Lock Screen Widget and app localization. Moreover, ensuring app

intuitiveness, simplicity, and usability emerged as important considerations for the intended user base.

- **What is the state of art of live ECG monitoring mobile applications?**

  In related work, the state of the art in live ECG monitoring mobile applications for epilepsy patients has been investigated. Existing solutions, including Bittium's Mobile Cardiac Telemetry, Beurer HealthManager Pro, KardiaMobile by AliveCor, and Biostrap, provide ECG monitoring capabilities but are predominantly intended for cardiac patients or general health monitoring. These applications provide features such as ECG monitoring in real-time, heart rate tracking, and data analysis. However, they lack features tailored to the requirements of patients with epilepsy. Using the Movesense MD sensor and the EpiHeartMonitor application, this thesis provides epilepsy patients with a reliable ECG monitoring tool in real-time that can satisfy their needs.

Application Validation

Three validation studies were conducted on the final product, each focusing on a distinct aspect: usability, data transmission analysis, and accessibility.

An app user with epilepsy completed a SUS questionnaire based on-one day app usage as part of the initial study. The collected data was used to evaluate the usability of the EpiHeartMonitor app, which received a score of 75, which is above average and close to the exceptional threshold, indicating a highly positive evaluation. In addition, an interview was conducted in which notable app features such as ECG visualization, the Lock Screen Widget, and the Pill Intake Tracker were highlighted.

The author conducted the second study, which involved validating the application's functionality by analyzing the data transmitted by it. This required comparing the data sent to the CARP server with the data received from the Movesense device. The author was able to successfully validate this aspect of the investigation.

The final study, also conducted by the author, focused on accessibility. The study examined the design requirements for Voice Over, Dynamic Type fonts, and Color Contrast. The author effectively validated these aspects of accessibility, resulting in a positive evaluation.

Multiple potential enhancement areas have been identified. Priority number one is improving security, as the current method for uploading SQLite database files lacks sufficient security. A potential improvement could entail the use of an alternative uploading service, which could mitigate the SQLite-intrinsic security vulnerability. Incorporating remote settings could further improve support for research users if implemented.

APPENDIX A

Acronyms

PDR Project Definition Report

ILO intended learning objectives

IPR Intellectual Property Rights

SUDEP Sudden Unexpected Death in Epilepsy

ECG Electrocardiogram

BLE Bluetooth Low Energy

CARP CACHET Research Platform

CACHET Copenhagen Center for Health Technology

SPM Swift Package Manager

MVVM Model-View-ViewModel

BLE Bluetooth Low Energy

XXXL Extra Extra Extra Large Content Font

AX M Accessibility Medium

AX XL Accessibility Extra Large

WCAG Web Content Accessibility Guidelines

UCD User Centred Design

AI Artificial Intelligence

SUS System Usability Scale

UTC Universal Time Coordinated

UI User Interface

UX User Experience

APPENDIX B

Appendix

In the Appendix B documentation regards the design process and Faros implementation will be found.

B.1 User Centered Design

The author has conducted two interviews to understand potential users needs and requirements. The outcome of the interviews is presented below.

Participant 1: Christian (epilepsy person, middle-aged adult)

Question 1: Could you please share your experience regarding the ECG monitoring process?

Christian's Answer: I haven't done that on any wearable. I worked with a wearable for an app, but haven't really interacted with such a device.

Question 2: What specific features or functionalities would you like to see in an application designed to assist with epilepsy management and ECG monitoring?

Christian's Answer: Monitoring the heart activity in case of a concerning pattern.

Question 3: In your opinion, what is the biggest issue when using an app that connects to a wearable device?

Christian's Answer: I wanted it connected all the time. Having to click a connect button and waiting is hard. The connection itself is the most annoying thing. The feel, more direct feel to it - have feedback of the connection.

Question 4: On a scale from 0 to 10, where 0 represents the least importance and 10 signifies the highest importance, how important is accessibility for an ECG monitoring mobile app to you?

Christian's Answer: UX accessibility is very important. It should be clear. But it is hard to achieve that. It should be clear and simple for the connection status, so I would rate it as 9. For secondary sections, I would rate it around 5-6.

Question 5: On the same scale, how important is usability for an ECG monitoring mobile app to you?

Christian's Answer: I would rate usability as 8. I would like to have more options and be more involved in it, so it is not only a background task that runs but also has some evolving features.

Participant 2: Tobias (youth person with cardiac problems)

Question 1: Could you please share your experience regarding the ECG monitoring process?

Tobias's Answer: I've tried 15 times, holter monitoring. It monitors the heart rhythm. The annoying part is that if I feel symptoms, I can't get the feedback instantly. The process is very time-consuming. I actually want to see what my heart is doing, but I can't see the visualization. Around 80% of people with heart problems need coaching about anxiety. It is so stressful not knowing what's going on.

Question 2: What specific features or functionalities would you like to see in an application designed to assist with epilepsy management and ECG monitoring?

Tobias's Answer: Two things: visualization of the ECG and a feature about irregular heartbeats, similar to the Apple Health app.

Question 3: In your opinion, what is the biggest issue when using an app that connects to a wearable device?

Tobias's Answer: Connection issues and how accurate it is. Connection status.

Question 4: On a scale from 0 to 10, where 0 represents the least importance and 10 signifies the highest importance, how important is accessibility for an ECG monitoring mobile app to you?

Tobias's Answer: Intuitively, I would rate it as 10. Overall UX accessibility, contrast is not that important. I think it would be great to have a localized application that offers multi language support.

Question 5: On the same scale, how important is usability for an ECG monitoring mobile app to you?

Tobias's Answer: I would rate usability as 10. It should be super easy to use this kind of app. It should only show you the info that you need to know.

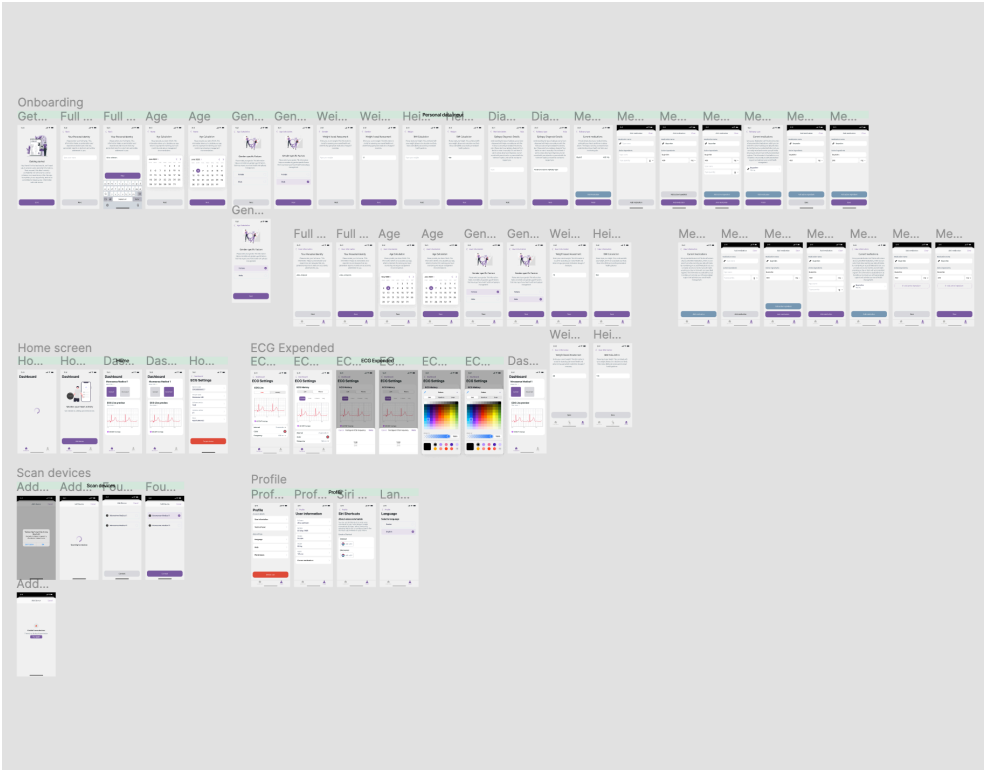


Figure B.1: First prototype screens collection.

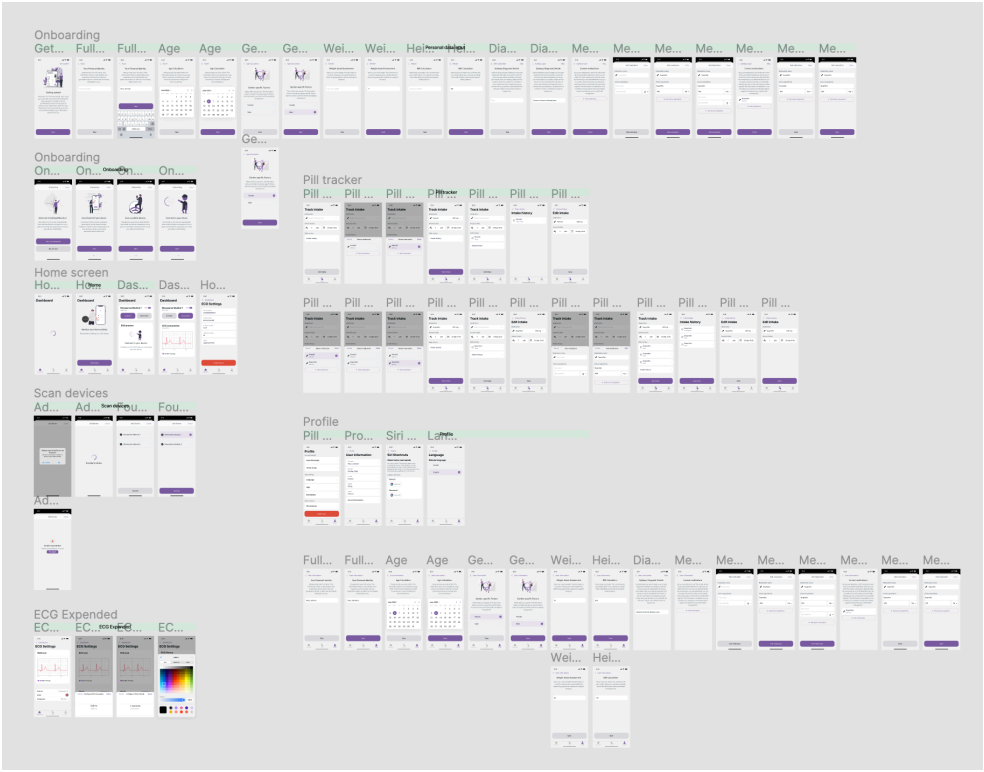


Figure B.2: Second prototype screens collection.

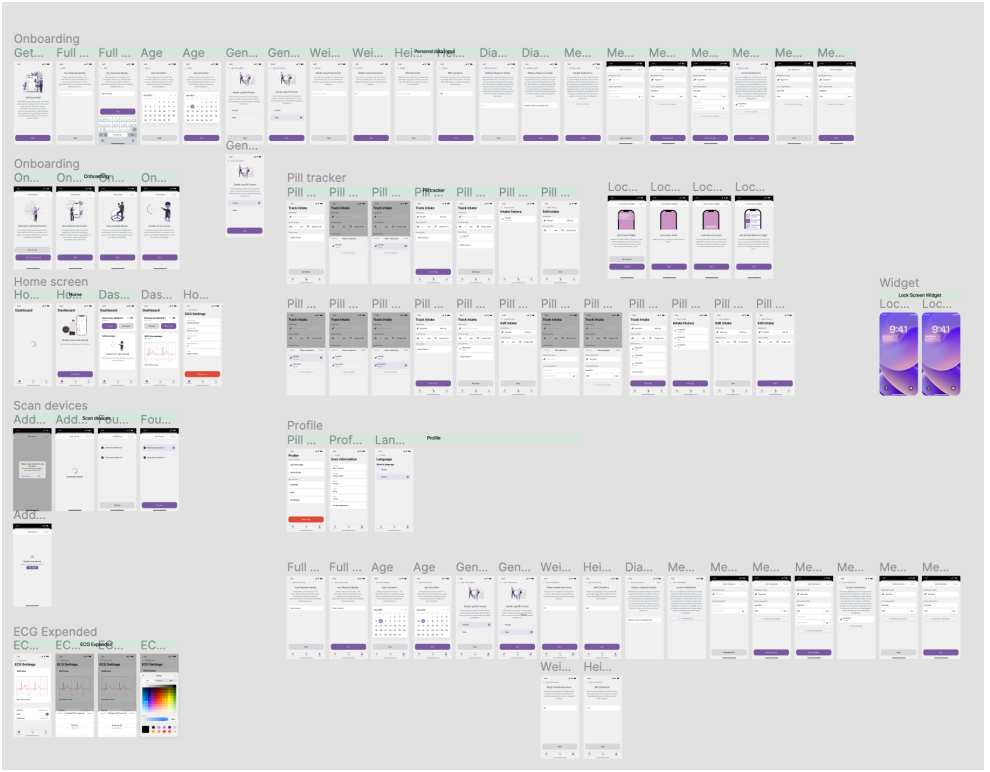
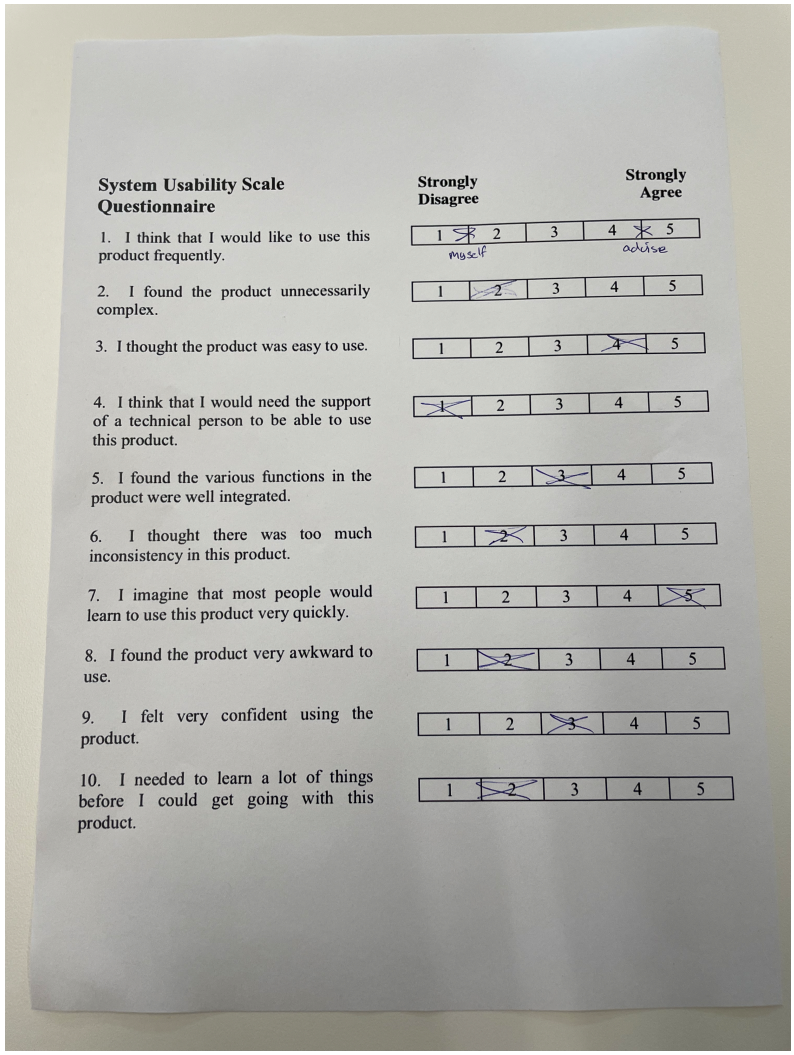


Figure B.3: Final prototype screens collection.

B.2 Study results

B.2.1 User study

During the user study, Christian, the participant evaluated the EpiHeartMonitor application using SUS. Figure B.4 depicts his evaluation.



System Usability Scale Questionnaire

	Strongly Disagree				Strongly Agree
1. I think that I would like to use this product frequently.	1	2	3	4	5
2. I found the product unnecessarily complex.	1	2	3	4	5
3. I thought the product was easy to use.	1	2	3	4	5
4. I think that I would need the support of a technical person to be able to use this product.	1	2	3	4	5
5. I found the various functions in the product were well integrated.	1	2	3	4	5
6. I thought there was too much inconsistency in this product.	1	2	3	4	5
7. I imagine that most people would learn to use this product very quickly.	1	2	3	4	5
8. I found the product very awkward to use.	1	2	3	4	5
9. I felt very confident using the product.	1	2	3	4	5
10. I needed to learn a lot of things before I could get going with this product.	1	2	3	4	5

myself

advise

Figure B.4: SUS Questionnaire.

In addition, the author conducted an interview in which specific questions were addressed.

Question 1: How easy or difficult was it to set up and start using the app?

Christian's Answer: Very easy.

Question 2: Were there any functions or features that were hard to find or use?

Christian's Answer: No, they were easily accessible.

Question 3: Was there anything in the app you found confusing or hard to understand?

Christian's Answer: ECG Preview called in both preview.

Question 4: Did the ECG live capabilities work as you expected them to?

Christian's Answer: Almost, I would like to see it like a "snake" instead of a "worm". Generally yes.

Question 5: Did the pill intake tracker function correctly?

Christian's Answer: Generally yes. I found a bug in time of the pill, it does not set the pill to now date once the first pill was tracked.

Question 6: Did the device connection status widget work well, and was it easy to understand?

Christian's Answer: I think it was a good idea. It requires it use the device. It is very convent. The guide was useful to install a Lock Screen, it worked even for me, despite not being an IOS user.

Question 7: What did you like the most about the app?

Christian's Answer: I like the ECG, you could see the two phases of the heartbeat, high enough resolution to do that. I think it is a good feature to have the pill tacker. I think the Lock Screen widget is a very good thing, it is simple and it shows the device connection and batter statuses, that is useful and easy to access. I used the widget while I was testing the app.

Question 8: What did you like the least?

Christian's Answer: ECG noise sensibility. But I suspect it is a hardware problem.

Question 9: Were you satisfied with the app and its features?

Christian's Answer: Yes.

Question 10: Are there any features that you think should be added to the app?

Christian's Answer: Epilepsy type should offer a preset list and possible to add a custom.

Question 11: How could the app be improved to better meet your needs?

Christian's Answer: Rolling display, it clears it instead of starting from the beginning.

Additionally, Christian provided some notes he took while testing and how he thinks the ECG view can be improved.

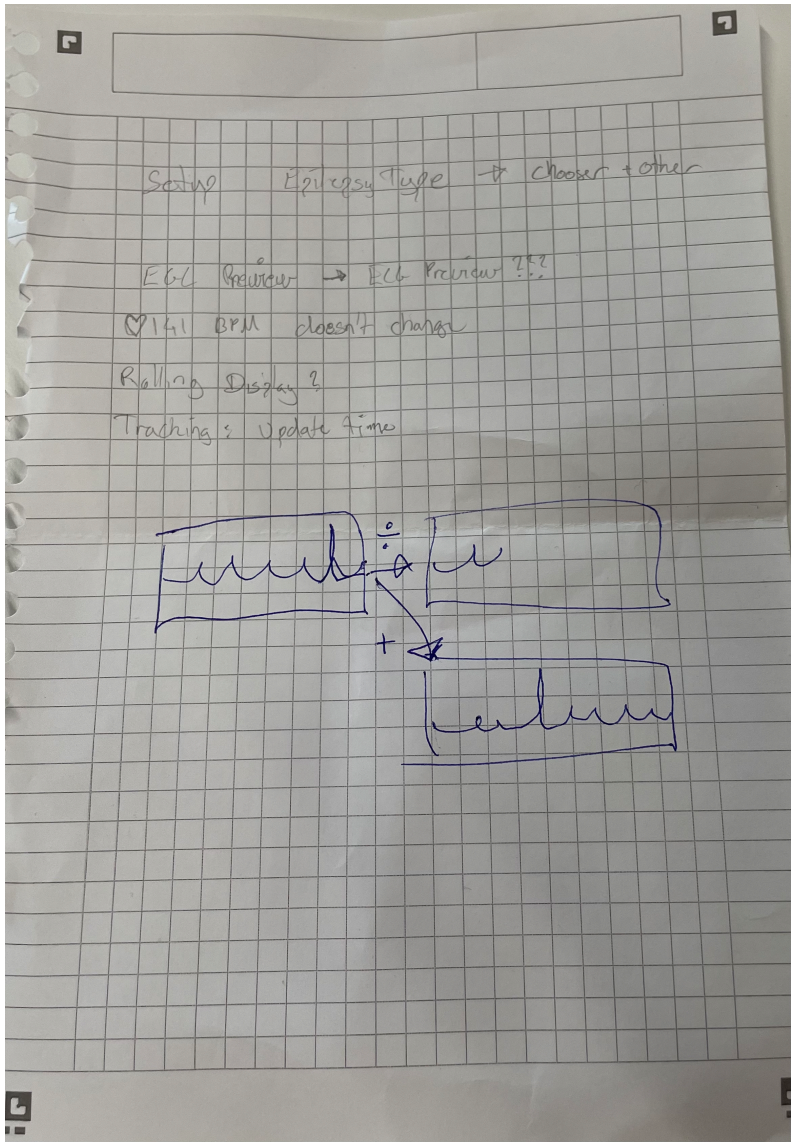


Figure B.5: Christian's testing notes.

B.2.2 Data study

The data analysis study outlined in Section 6.2 necessitated the use of ECG values print logs. The following lines were generated using Xcode, with the addition of a

vertical bar character to create line breaks:

```
1
2  2023-06-27T21:26:30.151
       ↪ 0,0,-1,-2,-3,-6,-6,-9,-15,-12,-25,-20,-10,-67,8,120
3  2023-06-27T21:26:30.263 -123,-335,249,1549,2513,2375, |
       ↪ 1564,1017,975,897,660,547,465,343,251,164
4  2023-06-27T21:26:30.385 63,-47,-101,-45,75,123,53,-60,|
       ↪ -141,-185,-225,-263,-298,-322,-335,-353
5  2023-06-27T21:26:30.503 -344,-346,-332,-416,-891,-1042, |
       ↪ 461,3166,4686,3532,1004,-586,-656,-423,-664,-893
6  2023-06-27T21:26:30.653 -848,-819,-823,-792,-763,-750,|
       ↪ -739,-705,-667,-631,-591,-548,-500,-450,-407,-359
7  2023-06-27T21:26:30.773 -302,-250,-199,-128,-31,49,111,|
       ↪ 174,231,272,283,254,163,-24,-277,-533
8  2023-06-27T21:26:30.893 -734,-881,-991,-1049,-1055,-1041,-1039,|
       ↪ -1033,-1000,-947,-903,-884,-863,-819,-766,-748
9  2023-06-27T21:26:31.015 -751,-739,-702,-667,-657,-660,-643,-610,-600,|
       ↪ -627,-632,-601,-560,-538,-539,-536
10 2023-06-27T21:26:31.133 -517,-490,-463,-451,-450,-446,-428,-407,|
       ↪ -409,-427,-466,-501,-473,-343,-172,-74
11 2023-06-27T21:26:31.253 -103,-179,-224,-233,-222,-214,-221,-232,|
       ↪ -235,-221,-170,-132,-99,-257,-699,-451
12 2023-06-27T21:26:31.404 1456,3919,4568,2739,348,-602,-235,17,-280,|
       ↪ -472,-400,-347,-336,-317,-294,-264
13 2023-06-27T21:26:31.525 -221,-182,-142,-96,-56,-25,15,66,126,172,223,|
       ↪ 295,369,435,507,592
14 2023-06-27T21:26:31.644 678,740,773,791,796,750,590,335,41,-233,|
       ↪ -448,-608,-703,-744,-754,-752
15 2023-06-27T21:26:31.763 -735,-706,-664,-626,-598,-565,-529,-493,-466,|
       ↪ -454,-454,-452,-437,-428,-417,-402
16 2023-06-27T21:26:31.883
       ↪ -398,-392,-384,-380,-379,-369,-348,-340,-349,-342,|
       ↪ -308,-271,-267,-284,-284,-268
17 2023-06-27T21:26:32.003 -260,-276,-316,-363,-364,-253,-77,52,60,-25,-103,|
       ↪  -131,-131,-120,-113,-99
18 2023-06-27T21:26:32.156 -76,-79,-62,-46,-11,-64,-538,-763,644,3377,|
       ↪ 5092,4131,1635,-58,-217,19
19 2023-06-27T21:26:32.304 -218,-467,-427,-399,-409,-381,-341,-298,|
       ↪ -268,-237,-210,-175,-138,-95,-43,4
20 2023-06-27T21:26:32.393 32,54,84,142,203,261,322,387,449,495,514,|
       ↪ 526,532,506,421,246
21 2023-06-27T21:26:32.514 19,-203,-374,-486,-561,-610,-621,-613,-588,|
       ↪ -562,-533,-499,-468,-435,-413,-401
22 2023-06-27T21:26:32.633 -388,-376,-356,-334,-318,-312,-312,-304,|
       ↪ -296,-295,-300,-304,-289,-277,-273,-268
23 2023-06-27T21:26:32.753 -257,-245,-229,-230,-224,-217,-206,-205,|
       ↪ -211,-223,-247,-303,-345,-291,-132,28
24 2023-06-27T21:26:32.904 79,27,-52,-96,-101,-101,-98,-102,-104,|
       ↪ -110,-111,-85,-49,-7,-247,-764
25 2023-06-27T21:26:33.023 -359,1800,4324,4871,3019,721,-243,-122,|
       ↪ -119,-416,-501,-432,-425,-410,-383,-360
26 2023-06-27T21:26:33.144 -325,-285,-250,-223,-192,-157,-124,-84,|
       ↪ -42,10,67,110,155,209,268,334
27 2023-06-27T21:26:33.265 391,439,473,492,510,512,461,330,130,-89,|
```

```
      ↪ -293,-457,-570,-628,-647,-648
  2023-06-27T21:26:33.414 -648,-630,-602,-566,-541,-527,-509,-486,|
      ↪ -463,-444,-426,-408,-392,-389,-395,-399
  2023-06-27T21:26:33.503 -394,-377,-367,-363,-363,-355,-335,-320,|
      ↪ -317,-319,-319,-303,-287,-271,-262,-253
  2023-06-27T21:26:33.654 -238,-234,-248,-286,-345,-368,-298,-144,|
      ↪ 6,60,6,-83,-134,-142,-143,-143
  2023-06-27T21:26:33.773 -151,-151,-146,-145,-115,-81,-42,-249,|
      ↪ -745,-433,1568,4022,4627,2855,580,-374
  2023-06-27T21:26:33.893 -176,-82,-375,-500,-435,-423,-414,-386,|
      ↪ -360,-327,-297,-267,-237,-201,-163,-116
  2023-06-27T21:26:34.013 -72,-33,5,57,105,156,207,262,325,394,|
      ↪ 452,489,508,523,516,470
  2023-06-27T21:26:34.133 345,145,-87,-293,-451,-556,-633,-676,|
      ↪ -683,-667,-642,-620,-595,-565,-537,-510
  2023-06-27T21:26:34.253 -493,-484,-472,-458,-440,-427,-416,-411,|
      ↪ -414,-409,-410,-408,-389,-368,-362,-365
  2023-06-27T21:26:34.405 -360,-343,-331,-325,-318,-304,-295,-283,|
      ↪ -263,-247,-242,-234,-223,-222,-235,-267
  2023-06-27T21:26:34.523 -303,-339,-335,-240,-61,89,114,22,-97,-162,|
      ↪ -157,-138,-144,-154,-132,-119
  2023-06-27T21:26:34.644 -121,-121,-105,-60,-314,-815,-282,2008,4480,|
      ↪ 4812,2815,559,-289,-131,-181,-486
  2023-06-27T21:26:34.764 -539,-467,-460,-441,-403,-371,-344,-319,-278,|
      ↪ -226,-187,-161,-121,-66,-18,18
  2023-06-27T21:26:34.883 49,105,167,225,272,323,378,425,463,477,488,|
      ↪ 493,456,339,159,-60
  2023-06-27T21:26:35.003 -259,-411,-511,-561,-583,-596,-597,-579,-550,|
      ↪ -525,-506,-491,-473,-451,-433,-417
  2023-06-27T21:26:35.154 -403,-400,-393,-386,-374,-364,-361,-360,-358,|
      ↪ -361,-353,-334,-308,-291,-291,-288
  2023-06-27T21:26:35.281 -279,-262,-247,-247,-252,-249,-242,-244,|
      ↪ -276,-316,-311,-224,-77,40,70,16
  2023-06-27T21:26:35.410 -55,-94,-103,-100,-92,-95,-87,-91,-96,-65,|
      ↪ -27,-4,-255,-675,-101,2057
  2023-06-27T21:26:35.515 4257,4321,2239,149,-412,-6,61,-267,-377,|
      ↪ -308,-298,-280,-244,-208,-176,-144
  2023-06-27T21:26:35.633 -110,-74,-34,5,35,71,115,169,223,273,331,|
      ↪ 389,456,526,590,648
  2023-06-27T21:26:35.754 687,722,751,742,665,491,239,-35,-282,-469,|
      ↪ -586,-639,-648,-644,-637,-617
Update local data  2023-06-27 19:26:35 +0000
  2023-06-27T21:26:35.904 -592,-563,-545,-520,-490,-457,-437,-432,|
      ↪ -426,-404,-386,-381,-389,-391,-376,-363
  2023-06-27T21:26:36.023 -349,-348,-351,-346,-338,-325,-315,-311,|
      ↪ -299,-289,-281,-284,-301,-347,-397,-375
  2023-06-27T21:26:36.143 -242,-71,35,28,-46,-111,-141,-146,-126,|
      ↪ -120,-116,-117,-131,-116,-89,-35
  2023-06-27T21:26:36.263 -64,-510,-775,511,3152,4909,4102,1716,9,|
      ↪ -207,2,-232,-483,-449,-407,-404
  2023-06-27T21:26:36.383 -376,-345,-316,-281,-250,-226,-196,-167,|
      ↪ -124,-78,-44,0,43,92,141,182
  2023-06-27T21:26:36.503 234,298,352,396,426,453,474,479,442,335,|
      ↪ 155,-58,-260,-412,-521,-585
  2023-06-27T21:26:36.655 -616,-614,-584,-553,-542,-536,-524,-495,|
```

```
    ↪ -458,-431,-417,-407,-392,-382,-375,-365
2023-06-27T21:26:36.773 -357,-343,-338,-342,-338,-328,-323,-319,|
    ↪ -310,-297,-291,-286,-275,-260,-247,-242
2023-06-27T21:26:36.893 -243,-246,-268,-322,-370,-337,-192,-24,|
    ↪ 57,29,-45,-99,-121,-129,-126,-128
2023-06-27T21:26:37.014 -128,-132,-145,-125,-93,-34,-81,-559,-797,|
    ↪ 550,3219,4967,4161,1788,58,-237
2023-06-27T21:26:37.134 -94,-316,-524,-488,-457,-451,-421,-391,-362,|
    ↪ -333,-302,-269,-235,-197,-161,-122
2023-06-27T21:26:37.253 -84,-39,15,65,110,145,193,243,292,330,355,|
    ↪ 370,382,382,352,263
2023-06-27T21:26:37.404 109,-80,-261,-404,-496,-551,-575,-585,-581,|
    ↪ -569,-546,-523,-502,-483,-460,-443
2023-06-27T21:26:37.523 -425,-407,-389,-378,-372,-373,-364,-357,|
    ↪ -348,-343,-339,-329,-325,-316,-305,-297
2023-06-27T21:26:37.643 -290,-287,-274,-263,-256,-247,-242,-236,|
    ↪ -237,-256,-290,-347,-362,-275,-112,33
2023-06-27T21:26:37.765 73,16,-59,-110,-123,-124,-123,-125,-124,|
    ↪ -125,-128,-107,-78,-21,-189,-713
2023-06-27T21:26:37.883 -520,1417,4011,4914,3358,1032,-145,-133,|
    ↪ -108,-395,-502,-436,-429,-426,-405,-381
2023-06-27T21:26:38.003 -349,-310,-272,-236,-211,-181,-148,-105,|
    ↪ -60,-16,15,56,94,146,196,244
2023-06-27T21:26:38.154 285,316,338,350,359,355,305,191,16,-184,|
    ↪ -352,-475,-561,-614,-641,-642
2023-06-27T21:26:38.273 -629,-605,-569,-545,-525,-510,-487,-461,|
    ↪ -446,-432,-416,-410,-407,-403,-393,-378
2023-06-27T21:26:38.393 -369,-362,-365,-367,-362,-346,-325,-313,|
    ↪ -312,-304,-293,-275,-261,-249,-235,-231
2023-06-27T21:26:38.513 -229,-234,-244,-261,-299,-342,-324,-205,|
    ↪ -30,74,52,-37,-109,-126,-121,-115
2023-06-27T21:26:38.633 -113,-117,-113,-132,-130,-94,-45,-31,|
    ↪ -402,-819,105,2616,4711,4365,2072,125
2023-06-27T21:26:38.753 -281,-6,-143,-458,-474,-409,-395,-358,|
    ↪ -330,-315,-294,-263,-216,-174,-136,-107
2023-06-27T21:26:38.905 -75,-28,25,74,111,148,204,269,327,369,|
    ↪ 403,446,485,507,523,507
2023-06-27T21:26:39.023 417,238,10,-197,-362,-487,-579,-628,-633,|
    ↪ -617,-588,-553,-529,-507,-491,-472
2023-06-27T21:26:39.145 -449,-434,-422,-404,-388,-378,-372,-357,|
    ↪ -334,-325,-336,-352,-349,-333,-318,-303
2023-06-27T21:26:39.263 -294,-286,-276,-272,-259,-243,-231,-228,|
    ↪ -238,-254,-282,-330,-343,-259,-108,24
2023-06-27T21:26:39.383 62,16,-61,-107,-121,-114,-106,-114,-112,|
    ↪ -106,-121,-105,-77,-22,-138,-629
2023-06-27T21:26:39.503 -579,1138,3681,4689,3193,843,-305,-130,|
    ↪ 59,-257,-457,-390,-347,-340,-313,-281
2023-06-27T21:26:39.654 -248,-215,-186,-152,-110,-68,-30,3,41,|
    ↪ 85,132,182,235,286,334,390
2023-06-27T21:26:39.773 449,503,548,575,596,598,535,383,149,|
    ↪ -106,-332,-503,-611,-659,-667,-662
2023-06-27T21:26:39.893 -653,-628,-590,-557,-529,-513,-493,-468,|
    ↪ -444,-427,-420,-410,-399,-384,-381,-380
2023-06-27T21:26:40.015 -373,-366,-364,-361,-357,-343,-329,-327,|
    ↪ -319,-305,-284,-275,-279,-275,-255,-243
```

```
  2023-06-27T21:26:40.134 -242,-258,-281,-331,-378,-344,-198,-22,|
      ↪ 70,39,-38,-101,-122,-120,-114,-113
  2023-06-27T21:26:40.253 -119,-120,-136,-124,-94,-37,-28,-458,-870,|
      ↪ 215,2905,5051,4666,2338,343,-175
  2023-06-27T21:26:40.404 -64,-268,-514,-492,-458,-467,-439,-404,|
      ↪ -375,-344,-320,-285,-249,-213,-177,-141
  2023-06-27T21:26:40.524 -95,-55,-20,18,55,103,152,194,235,274,|
      ↪ 301,318,320,317,286,203
  2023-06-27T21:26:40.643 59,-125,-297,-426,-523,-579,-609,-612,|
      ↪ -596,-581,-564,-539,-509,-477,-457,-440
  2023-06-27T21:26:40.763 -427,-410,-395,-386,-373,-365,-357,-350,|
      ↪ -345,-339,-337,-335,-329,-320,-304,-289
Update server started   2023-06-27 19:26:40 +0000
  2023-06-27T21:26:40.884 -282,-277,-271,-260,-246,-241,-238,-230,|
      ↪ -220,-207,-208,-217,-234,-268,-315,-336
Update local data   2023-06-27 19:26:40 +0000
  2023-06-27T21:26:41.306 -257,-95,49,92,37,-45,-92,-98,-85,-80,|
      ↪ -91,-94,-97,-105,-84,-60
  2023-06-27T21:26:41.344 20,-103,-665,-644,1183,3950,5202,3841,|
      ↪ 1406,-7,-118,-93,-380,-499,-444,-436
  2023-06-27T21:26:41.367 -428,-393,-362,-327,-294,-261,-234,-200,|
      ↪ -169,-128,-91,-54,-12,32,79,130
  2023-06-27T21:26:41.392 168,204,239,282,318,339,342,342,340,306,|
      ↪ 202,39,-136,-299,-425,-509
  2023-06-27T21:26:41.513 -551,-565,-573,-569,-548,-510,-477,-461,|
      ↪ -445,-430,-417,-402,-383,-357,-343,-342
  2023-06-27T21:26:41.632 -339,-332,-312,-303,-314,-339,-351,-338,|
      ↪ -311,-295,-286,-281,-280,-275,-248,-214
Update server completed   2023-06-27 19:26:41 +0000
  2023-06-27T21:26:41.753 -207,-233,-253,-239,-214,-205,-205,-197,|
      ↪ -192,-185,-186,-227,-305,-341,-247,-61
  2023-06-27T21:26:41.904 84,99,19,-61,-85,-83,-78,-85,-100,-99,|
      ↪ -100,-115,-102,-71,-1,-142
  2023-06-27T21:26:42.024 -679,-641,1121,3763,4926,3578,1230,-99,|
      ↪ -146,-72,-349,-506,-467,-449,-430,-390
  2023-06-27T21:26:42.143 -359,-335,-307,-278,-235,-193,-163,-134,|
      ↪ -102,-66,-22,17,59,98,140,181
  2023-06-27T21:26:42.293 223,266,298,320,339,356,359,324,218,51,|
      ↪ -144,-331,-474,-574,-630,-648
  2023-06-27T21:26:42.384 -648,-630,-612,-593,-575,-558,-534,-513,|
      ↪ -497,-483,-475,-464,-458,-450,-440,-433
  2023-06-27T21:26:42.503 -427,-421,-414,-407,-403,-395,-381,-372,|
      ↪ -374,-374,-351,-326,-309,-311,-316,-309
  2023-06-27T21:26:42.654 -289,-276,-269,-263,-259,-254,-240,-229,|
      ↪ -224,-227,-243,-267,-302,-309,-231,-81
  2023-06-27T21:26:42.774 52,86,26,-64,-112,-120,-114,-109,-118,-117,|
      ↪ -118,-130,-105,-70,-13,-119
  2023-06-27T21:26:42.893 -627,-655,1003,3616,4812,3452,1067,-228,-160,|
      ↪ 15,-275,-464,-403,-373,-363,-330
  2023-06-27T21:26:43.013 -295,-267,-244,-210,-175,-147,-114,-73,|
      ↪ -24,14,53,99,149,204,248,291
  2023-06-27T21:26:43.134 337,380,431,478,501,521,531,515,416,221,|
      ↪ -30,-257,-425,-540,-617,-653
  2023-06-27T21:26:43.254 -661,-640,-612,-589,-563,-537,-514,-485,|
      ↪ -462,-450,-436,-427,-418,-406,-394,-384
```

```
112   2023-06-27T21:26:43.405 -372,-367,-364,-361,-361,-360,-350,-337,|
          ↪ -319,-303,-296,-287,-280,-273,-259,-249
113   2023-06-27T21:26:43.523 -234,-223,-216,-210,-220,-240,-273,-298,|
          ↪ -269,-171,-42,51,60,11,-53,-84
114   2023-06-27T21:26:43.644 -86,-80,-81,-92,-89,-94,-97,-69,-33,-1,|
          ↪ -261,-707,-130,2050,4219,4235
115   2023-06-27T21:26:43.764 2161,139,-357,28,30,-322,-411,-323,-297,|
          ↪ -282,-252,-220,-185,-156,-131,-98
116   2023-06-27T21:26:43.883 -63,-23,13,52,96,138,184,237,284,328,378,|
          ↪ 451,518,561,582,599
117   2023-06-27T21:26:44.003 620,616,536,359,112,-142,-348,-497,-592,|
          ↪ -647,-669,-663,-639,-603,-576,-559
118   2023-06-27T21:26:44.154 -539,-510,-475,-449,-436,-426,-419,-407,|
          ↪ -394,-385,-380,-386,-383,-376,-368,-357
119   2023-06-27T21:26:44.274 -347,-342,-337,-326,-313,-301,-296,-288,|
          ↪ -280,-268,-260,-250,-229,-220,-223,-237
120   2023-06-27T21:26:44.394 -265,-310,-334,-272,-130,14,84,60,-11,|
          ↪ -73,-106,-116,-111,-108,-102,-93
121   2023-06-27T21:26:44.543 -102,-94,-67,-13,-11,-443,-826,277,2936,|
          ↪ 5010,4567,2246,304,-170,-43,-245
122   2023-06-27T21:26:44.633 -480,-455,-430,-442,-409,-370,-335,-306,|
          ↪ -283,-250,-213,-179,-140,-99,-56,-21
123   2023-06-27T21:26:44.753 16,57,100,134,176,222,261,297,319,336,|
          ↪ 350,351,328,248,112,-73
124   2023-06-27T21:26:44.904 -253,-393,-500,-571,-606,-614,-600,-581,|
          ↪ -559,-530,-500,-475,-456,-444,-434,-426
125   2023-06-27T21:26:45.023 -407,-389,-378,-375,-373,-362,-351,-347,|
          ↪ -345,-339,-330,-322,-317,-315,-309,-300
126   2023-06-27T21:26:45.145 -283,-273,-269,-266,-257,-242,-238,-235,|
          ↪ -235,-244,-268,-315,-350,-303,-161,1
127   2023-06-27T21:26:45.281 74,36,-52,-110,-121,-122,-121,-123,-130,|
          ↪ -125,-141,-144,-116,-66,-32,-389
128   2023-06-27T21:26:45.406 -858,-53,2413,4674,4610,2457,389,-242,|
          ↪ -91,-221,-492,-487,-428,-440,-434,-412
129   2023-06-27T21:26:45.505 -374,-329,-298,-266,-234,-203,-167,-129,|
          ↪ -83,-40,0,39,78,124,166,215
130   2023-06-27T21:26:45.656 262,300,331,351,363,370,346,272,131,-64,|
          ↪ -255,-410,-523,-589,-624,-633
131   2023-06-27T21:26:45.774 -630,-616,-586,-555,-526,-504,-483,-457,|
          ↪ -438,-427,-417,-407,-393,-382,-379,-377
132   2023-06-27T21:26:45.893 -365,-352,-349,-349,-349,-339,-323,-307,|
          ↪ -299,-293,-289,-277,-261,-252,-246,-242
133 Update local data  2023-06-27 19:26:45 +0000
134   2023-06-27T21:26:46.014 -234,-225,-217,-205,-205,-224,-270,-330,|
          ↪ -341,-252,-89,50,87,33,-42,-85
135   2023-06-27T21:26:46.134 -97,-97,-93,-98,-105,-110,-127,-113,-76,|
          ↪ -3,-44,-540,-796,551,3203,4896
136   2023-06-27T21:26:46.253 4033,1671,19,-204,-41,-269,-483,-442,|
          ↪ -415,-419,-394,-363,-325,-298,-274,-242
137   2023-06-27T21:26:46.404 -202,-163,-127,-94,-53,-12,39,78,120,168,|
          ↪ 211,257,298,332,363,385
138   2023-06-27T21:26:46.524 405,411,379,280,108,-91,-285,-442,-556,|
          ↪ -626,-642,-628,-612,-603,-591,-564
139   2023-06-27T21:26:46.644 -530,-504,-479,-461,-443,-426,-414,-404,|
          ↪ -395,-385,-379,-380,-376,-370,-365,-350
```

```
  2023-06-27T21:26:46.764 -334,-315,-304,-313,-311,-297,-273,-259,|
      ↪ -256,-246,-236,-235,-235,-226,-208,-197
  2023-06-27T21:26:46.884 -201,-219,-250,-280,-283,-217,-85,47,100,|
      ↪ 61,-18,-70,-88,-91,-90,-95
  2023-06-27T21:26:47.004 -88,-89,-107,-88,-50,10,-121,-639,-612,1117,|
      ↪ 3714,4765,3267,881,-295,-120
  2023-06-27T21:26:47.155 72,-244,-457,-396,-355,-342,-313,-281,-249,|
      ↪ -219,-187,-156,-114,-73,-38,-5
  2023-06-27T21:26:47.274 28,70,124,174,221,267,327,398,454,492,515,|
      ↪ 550,581,583,527,386
  2023-06-27T21:26:47.394 173,-68,-298,-472,-587,-650,-675,-675,-655,|
      ↪ -629,-601,-579,-550,-512,-477,-456
  2023-06-27T21:26:47.514 -442,-440,-437,-422,-402,-386,-384,-387,|
      ↪ -385,-375,-361,-352,-352,-350,-349,-337
  2023-06-27T21:26:47.634 -319,-306,-301,-290,-276,-264,-267,-272,|
      ↪ -284,-309,-349,-354,-271,-116,14,60
  2023-06-27T21:26:47.754 12,-63,-117,-140,-137,-118,-125,-131,-130,|
      ↪ -133,-107,-75,-30,-103,-544,-653
  2023-06-27T21:26:47.906 763,3240,4526,3325,980,-345,-212,96,-162,|
-431,-392,-329,-317,-278,-233,-219
  2023-06-27T21:26:48.024 -206,-185,-142,-89,-49,-23,13,60,118,168,|
209,251,301,368,437,506
  2023-06-27T21:26:48.143 565,611,641,653,650,603,456,212,-72,-318,|
-487,-596,-662,-691,-696,-680
  2023-06-27T21:26:48.263 -650,-607,-572,-556,-541,-517,-475,-441,|
-427,-422,-415,-403,-394,-392,-387,-378
  2023-06-27T21:26:48.383 -370,-362,-357,-351,-341,-334,-331,-326,|
-315,-296,-287,-280,-267,-252,-249,-259
  2023-06-27T21:26:48.504 -272,-286,-324,-364,-338,-204,-37,60,45,|
-30,-93,-115,-118,-115,-121,-131
  2023-06-27T21:26:48.654 -124,-130,-129,-104,-68,-14,-301,-826,-232,|
2143,4611,4841,2765,544,-226,-55
  2023-06-27T21:26:48.773 -147,-470,-519,-457,-462,-443,-407,-371,|
-335,-298,-272,-248,-223,-189,-149,-110
  2023-06-27T21:26:48.894 -69,-24,19,56,97,146,206,254,288,308,324,|
346,354,334,255,102
  2023-06-27T21:26:49.014 -92,-289,-450,-564,-643,-683,-686,-670,|
-644,-623,-597,-570,-540,-508,-482,-464
  2023-06-27T21:26:49.135 -453,-437,-417,-406,-401,-398,-390,-380,|
-371,-364,-360,-358,-352,-343,-331,-320
  2023-06-27T21:26:49.254 -318,-307,-288,-280,-280,-281,-277,-263,|
-254,-245,-241,-238,-237,-257,-304,-361
  2023-06-27T21:26:49.404 -366,-255,-89,34,43,-29,-106,-141,-145,|
-141,-139,-147,-150,-155,-176,-165
  2023-06-27T21:26:49.546 -131,-53,-148,-695,-790,903,3713,5200,|
4019,1540,-31,-216,-148,-429,-591,-542
  2023-06-27T21:26:49.643 -529,-516,-477,-448,-413,-379,-350,-318,|
-282,-249,-215,-182,-136,-91,-54,-19
  2023-06-27T21:26:49.764 22,73,124,163,202,237,261,272,281,279,|
236,131,-22,-190,-348,-481
  2023-06-27T21:26:49.884 -573,-617,-633,-636,-629,-611,-580,|
-551,-525,-500,-476,-450,-427,-410,-402,-398
  2023-06-27T21:26:50.004 -383,-362,-345,-336,-334,-331,-328,|
-320,-306,-298,-288,-279,-276,-262,-252,-240
  2023-06-27T21:26:50.156 -227,-215,-203,-198,-199,-194,-184,|
```

```
 -165,-153,-150,-142,-137,-128,-123,-121,-111
  2023-06-27T21:26:50.274 -110,-109,-121,-136,-165,-218,-235,|
 -141,23,152,158,81,4,-29,-22,-15
  2023-06-27T21:26:50.393 -16,-33,-36,-30,-39,-32,-10,51,-21,|
 -542,-689,874,3619,5160,4074,1633
  2023-06-27T21:26:50.515 46,-133,-31,-293,-479,-428,-414,-416,|
 -379,-344,-311,-288,-266,-234,-203,-168
  2023-06-27T21:26:50.634 -130,-92,-50,-9,30,69,109,150,196,237,|
 271,309,337,353,358,364
  2023-06-27T21:26:50.754 344,275,141,-31,-198,-347,-458,-524,|
 -551,-560,-568,-558,-526,-495,-475,-459
Update local data   2023-06-27 19:26:50 +0000
  2023-06-27T21:26:50.904 -440,-419,-403,-388,-372,-364,-358,-354,|
 -345,-328,-320,-320,-329,-323,-308,-295
  2023-06-27T21:26:51.024 -289,-285,-273,-261,-250,-242,-241,-232,|
 -217,-213,-218,-218,-200,-174,-160,-160
  2023-06-27T21:26:51.144 -158,-148,-140,-141,-139,-128,-118,-114,|
 -117,-118,-109,-104,-97,-102,-122,-144
  2023-06-27T21:26:51.265 -183,-198,-145,-28,87,138,117,56,-2,-32,|
 -42,-29,-15,-11,-9,-32
  2023-06-27T21:26:51.384 -45,-20,29,44,-338,-748,212,2756,4836,|
 4439,2116,173,-227,36,-119,-428
  2023-06-27T21:26:51.504 -432,-375,-380,-359,-322,-283,-250,-229,|
 -202,-166,-122,-87,-52,-13,27,65
  2023-06-27T21:26:51.654 103,140,187,239,291,341,393,432,456,|
 470,475,471,431,309,122,-84
  2023-06-27T21:26:51.773 -269,-412,-517,-584,-608,-605,-593,|
 -575,-546,-515,-492,-471,-448,-429,-412,-400
  2023-06-27T21:26:51.894 -388,-369,-359,-359,-363,-359,-353,|
 -342,-333,-330,-334,-332,-318,-301,-292,-287
  2023-06-27T21:26:52.015 -278,-272,-259,-242,-231,-224,-216,-210,|
 -200,-195,-187,-181,-168,-164,-160,-149
  2023-06-27T21:26:52.133 -140,-139,-134,-127,-120,-119,-123,-118,|
 -119,-144,-189,-215,-170,-57,67,136
  2023-06-27T21:26:52.254 113,44,-22,-47,-46,-41,-48,-62,-62,-68,|
 -65,-41,9,8,-393,-746
  2023-06-27T21:26:52.405 329,2888,4804,4225,1896,81,-217,26,-165,|
 -454,-451,-401,-411,-385,-351,-323
  2023-06-27T21:26:52.524 -293,-262,-227,-194,-163,-125,-89,-50,|
 -10,24,64,105,151,202,248,290
  2023-06-27T21:26:52.643 332,359,389,409,424,409,356,240,61,-145, |
      ↪ -329,-467,-553,-602,-630,-633
  2023-06-27T21:26:52.764 -626,-606,-576,-549,-528,-508,-480,-456, |
      ↪ -445,-435,-424,-406,-388,-379,-371,-370
  2023-06-27T21:26:52.884 -372,-359,-353,-350,-350,-345,-333,-320, |
      ↪ -307,-300,-297,-294,-287,-277,-267,-263
  2023-06-27T21:26:53.004 -255,-241,-227,-217,-221,-222,-207,-189, |
      ↪ -176,-182,-180,-171,-157,-145,-147,-149
  2023-06-27T21:26:53.186
      ↪ -151,-165,-195,-240,-240,-164,-38,67,103,75,15,-45,-77,-83,-75,-71
  2023-06-27T21:26:53.274 -75,-80,-98,-83,-51,22,-90,-633,-651, |
      ↪ 1104,3834,5105,3773,1348,-51,-125
  2023-06-27T21:26:53.394 -55,-350,-509,-451,-447,-447,-415,-384, |
      ↪ -346,-314,-288,-254,-218,-184,-148,-115
  2023-06-27T21:26:53.516
```

```
      ↪ -75,-33,7,43,79,121,166,212,252,287,318,343,350,343,308,239
  2023-06-27T21:26:53.634 122,-42,-221,-381,-493,-559,-600,-616,-619, |
      ↪ -608,-579,-549,-527,-509,-490,-460
  2023-06-27T21:26:53.754 -434,-421,-414,-403,-384,-373,-365,-365, |
      ↪ -363,-348,-341,-337,-336,-329,-314,-308
  2023-06-27T21:26:53.905 -306,-297,-286,-273,-265,-253,-245,-243, |
      ↪ -242,-231,-220,-209,-206,-196,-192,-185
  2023-06-27T21:26:54.024
      ↪ -172,-165,-161,-162,-167,-171,-165,-164,-186,-230,-254,|
-221,-117,4,78,85
  2023-06-27T21:26:54.144 31,-48,-103,-122,-106,-96,-97,-107,-131,-125,|
-91,-38,-36,-445,-860,123
  2023-06-27T21:26:54.264 2667,4733,4372,2124,205,-259,-82,-260, |
      ↪ -525,-517,-484,-492,-462,-422,-389,-363
  2023-06-27T21:26:54.383
      ↪ -340,-308,-270,-232,-200,-169,-135,-92,-50,-6,36,89,134,180,224,268
  2023-06-27T21:26:54.504 310,335,346,351,334,277,158,-15, |
      ↪ -214,-394,-530,-620,-675,-689,-686,-669
  2023-06-27T21:26:54.656 -647,-620,-591,-569,-547,-526,-506, |
      ↪ -485,-473,-465,-454,-444,-424,-404,-395,-398
  2023-06-27T21:26:54.802 -403,-397,-390,-382,-374,-359,-342, |
      ↪ -331,-325,-310,-294,-282,-280,-280,-270,-256
  2023-06-27T21:26:54.906 -243,-234,-227,-216,-195,-181,-189, |
      ↪ -202,-189,-185,-218,-255,-261,-197,-90,18
  2023-06-27T21:26:55.013
      ↪ 84,87,26,-52,-96,-89,-64,-56,-65,-81,-94,-65,-27,20,-88,-549
  2023-06-27T21:26:55.134 -563,1020,3538,4663,3284,921,-310, |
      ↪ -143,108,-176,-407,-353,-307,-310,-279,-239
  2023-06-27T21:26:55.254
      ↪ -208,-179,-152,-112,-68,-33,-1,37,76,119,163,225,291,353,408,459
  2023-06-27T21:26:55.403
      ↪ 519,587,639,672,689,702,678,558,346,79,-173,-371,-517,-604,-641,-647
      ↪
  2023-06-27T21:26:55.524 -639,-633,-614,-585,-549,-520,-493, |
      ↪ -473,-458,-444,-420,-392,-379,-373,-370,-357
  2023-06-27T21:26:55.644 -350,-363,-366,-353,-328,-321,-323, |
      ↪ -317,-302,-292,-282,-280,-264,-252,-238,-225
  2023-06-27T21:26:55.765
      ↪ -222,-219,-214,-204,-196,-210,-233,-262,-265,-201,-89,23,86,71,6,-63
      ↪
Update server started   2023-06-27 19:26:55 +0000
Update local data  2023-06-27 19:26:55 +0000
  2023-06-27T21:26:55.883
      ↪ -100,-105,-99,-101,-95,-88,-102,-89,-58,6,-76,-574,-672,906,3553,4890
      ↪
  2023-06-27T21:26:56.003 3659,1267,-128,-133,30,-269,-481,-416,-381, |
      ↪ -387,-357,-326,-289,-255,-226,-191
Update server completed   2023-06-27 19:26:56 +0000
  2023-06-27T21:26:56.154
      ↪ -148,-114,-80,-49,-6,42,85,131,179,231,285,341,409,465,505,527
  2023-06-27T21:26:56.276 539,539,487,361,167,-63,-274,-442,-562, |
      ↪ -625,-649,-647,-639,-627,-607,-572
  2023-06-27T21:26:56.394 -540,-511,-497,-468,-444,-424,-410,-402, |
      ↪ -393,-378,-368,-364,-364,-358,-348,-336
  2023-06-27T21:26:56.514 -331,-337,-334,-321,-311,-297,-288,-280, |
```

```
      ↪ -274,-269,-259,-252,-242,-236,-231,-223
263   2023-06-27T21:26:56.633
      ↪ -214,-207,-194,-195,-218,-261,-296,-249,-132,-7,67,65,16,-47,-90,-112
      ↪
264   2023-06-27T21:26:56.753 -108,-100,-105,-105,-118,-120,-98,-51,-1, |
      ↪ -326,-829,-102,2365,4742,4794,2639
265   2023-06-27T21:26:56.906 480,-214,-57,-174,-468,-489,-438,-447,|
266 -422,-385,-350,-315,-289,-256,-225,-190
267   2023-06-27T21:26:57.024 -147,-107,-73,-43,-1,52,99,145,194,246,|
268 298,338,374,401,419,424
269   2023-06-27T21:26:57.143 407,336,197,2,-195,-362,-490,-580,-633, |
      ↪ -644,-636,-618,-596,-571,-544,-516
270   2023-06-27T21:26:57.264 -487,-458,-435,-416,-403,-387,-371,-357, |
      ↪ -357,-359,-354,-338,-322,-315,-311,-310
271   2023-06-27T21:26:57.384 -305,-297,-286,-278,-255,-213,-188,-204, |
      ↪ -234,-249,-228,-199,-186,-175,-165,-152
272   2023-06-27T21:26:57.504 -144,-146,-139,-133,-146,-168,-195,-197, |
      ↪ -148,-59,47,118,120,64,-19,-72
273   2023-06-27T21:26:57.654 -79,-57,-49,-64,-80,-102,-93,-43,20,20, |
      ↪ -400,-823,129,2647,4737,4433
274   2023-06-27T21:26:57.774 2218,284,-199,-24,-183,-450,-443,-393, |
      ↪ -402,-399,-393,-363,-317,-275,-244,-205
275   2023-06-27T21:26:57.894 -154,-99,-66,-46,-23,22,78,132,177,226, |
      ↪ 273,323,370,410,441,465
276   2023-06-27T21:26:58.015 472,434,351,211,28,-174,-360,-500,-584, |
      ↪ -627,-642,-648,-640,-622,-602,-575
277   2023-06-27T21:26:58.134 -547,-518,-491,-478,-466,-449,-438,-431, |
      ↪ -416,-388,-381,-404,-427,-426,-406,-388
278   2023-06-27T21:26:58.254 -391,-383,-361,-340,-341,-346,-344,-329, |
      ↪ -308,-291,-286,-280,-284,-289,-287,-267
279   2023-06-27T21:26:58.404 -240,-230,-240,-258,-272,-291,-315,-309, |
      ↪ -228,-102,-5,6,-56,-136,-181,-183
280   2023-06-27T21:26:58.524 -166,-166,-176,-178,-186,-196,-183,-146, |
      ↪ -84,-299,-810,-458,1602,4022,4513,2694
281   2023-06-27T21:26:58.643 487,-375,-161,-117,-423,-532,-474,-457, |
      ↪ -433,-406,-386,-344,-283,-244,-228,-215
282   2023-06-27T21:26:58.764
      ↪ -168,-105,-64,-43,-3,72,149,189,205,241,311,392,449,495,539,566
283   2023-06-27T21:26:58.883 550,464,300,87,-134,-326,-479,-599,-669, |
      ↪ -702,-704,-699,-679,-634,-583,-556
284   2023-06-27T21:26:59.004 -547,-525,-492,-473,-473,-469,-444,-406, |
      ↪ -394,-407,-417,-416,-402,-394,-395,-385
285   2023-06-27T21:26:59.154 -370,-352,-355,-370,-367,-340,-317,-305, |
      ↪ -305,-305,-304,-298,-279,-269,-258,-255
286   2023-06-27T21:26:59.273 -272,-304,-330,-317,-249,-149,-46,8,-2,-71, |
      ↪ -148,-176,-161,-135,-136,-150
287   2023-06-27T21:26:59.394 -152,-157,-141,-115,-69,-129,-532,-625, |
      ↪ 752,3170,4416,3175,790,-531,-336,71
288   2023-06-27T21:26:59.514
      ↪ -128,-409,-387,-324,-316,-282,-234,-198,-174,-142,-95,-47,-16,9,41,95
      ↪
289   2023-06-27T21:26:59.634 150,203,256,313,372,434,514,608,679,723,750,|
290 787,814,762,600,351
291   2023-06-27T21:26:59.753 61,-207,-430,-592,-687,-733,-743,-745, |
      ↪ -725,-695,-657,-618,-582,-553,-525,-503
```

```
292   2023-06-27T21:26:59.904 -482,-457,-430,-409,-404,-408,-405, |
          ↪ -400,-391,-386,-381,-367,-345,-325,-318,-317
293   2023-06-27T21:27:00.023 -314,-305,-288,-273,-270,-266,-260, |
          ↪ -265,-287,-314,-295,-211,-95,5,60,63
294   2023-06-27T21:27:00.145 -2,-96,-148,-132,-79,-73,-108,-121, |
          ↪ -124,-101,-66,-28,-21,-376,-749,182
295   2023-06-27T21:27:00.266 2624,4612,4205,1943,73,-266,45,-86, |
          ↪ -403,-416,-347,-341,-326,-289,-244,-206
296   2023-06-27T21:27:00.384 -176,-145,-108,-67,-33,5,43,87,133,184,242,|
297  296,348,412,473,534
298   2023-06-27T21:27:00.503 583,620,643,648,603,481,275,27,-207, |
          ↪ -394,-539,-639,-694,-701,-691,-669
299   2023-06-27T21:27:00.654 -642,-613,-578,-552,-520,-490,-466,-450, |
          ↪ -431,-407,-389,-378,-375,-379,-383,-382
300   2023-06-27T21:27:00.773 -372,-356,-353,-352,-351,-350,-340,-328,-328, |
          ↪ -327,-321,-303,-291,-284,-284,-282
301 Update local data  2023-06-27 19:27:00 +0000
302   2023-06-27T21:27:00.894 -273,-275,-287,-304,-341,-349,-294, |
          ↪ -190,-88,-36,-42,-85,-149,-196,-214,-210
303   2023-06-27T21:27:01.014 -221,-242,-241,-245,-227,-199,-164,-156,-525, |
          ↪ -949,-43,2482,4661,4419,2155,119
304   2023-06-27T21:27:01.134 -408,-187,-304,-578,-588,-546,-549,-521,-485, |
          ↪ -454,-422,-390,-350,-313,-276,-233
305   2023-06-27T21:27:01.253 -187,-138,-87,-42,5,58,132,198,258,320,382,|
306  445,499,531,539,496
307   2023-06-27T21:27:01.406 388,209,-14,-228,-419,-562,-664,-729,-756, |
          ↪ -754,-734,-696,-663,-633,-606,-576
308   2023-06-27T21:27:01.524 -547,-527,-514,-497,-470,-440,-431,-420, |
          ↪ -409,-402,-400,-396,-386,-378,-382,-374
309   2023-06-27T21:27:01.644 -358,-344,-339,-339,-326,-307,-302,-300, |
          ↪ -300,-287,-275,-269,-272,-289,-300,-276
310   2023-06-27T21:27:01.764 -213,-126,-58,-15,-4,-36,-90,-135,-146, |
          ↪ -129,-118,-129,-130,-121,-117,-87
311   2023-06-27T21:27:01.883 -39,5,-169,-626,-388,1469,3845,4475, |
          ↪ 2753,488,-431,-128,75,-197,-353,-272
312   2023-06-27T21:27:02.004 -213,-198,-189,-192,-187,-162,-121, |
          ↪ -68,-20,13,57,93,137,184,229,280
313   2023-06-27T21:27:02.155 329,384,444,509,582,643,688,721,733,712,|
314  632,467,230,-28,-256,-431
315   2023-06-27T21:27:02.274 -560,-633,-665,-663,-652,-637,-614,-583,-548, |
          ↪ -510,-478,-451,-433,-409,-380,-358
316   2023-06-27T21:27:02.394 -341,-342,-341,-337,-331,-323,-319,-317,-310, |
          ↪ -302,-292,-284,-276,-270,-257,-246
317   2023-06-27T21:27:02.515 -232,-229,-226,-221,-208,-198,-191,-180,-171, |
          ↪ -166,-164,-164,-159,-156,-167,-191
318   2023-06-27T21:27:02.634 -219,-223,-165,-56,45,97,92,42,-20,-67,-89,|
319  -83,-75,-88,-85,-81
320   2023-06-27T21:27:02.754 -84,-65,-40,12,-167,-674,-419,1553,4087, |
          ↪ 4831,3133,806,-266,-147,-65,-349
321   2023-06-27T21:27:02.904 -461,-398,-392,-387,-357,-323,-287,-250, |
          ↪ -216,-179,-139,-105,-69,-25,10,51
322   2023-06-27T21:27:03.024 95,140,187,239,302,368,428,483,520,551,574,|
323  579,534,416,224,10
324   2023-06-27T21:27:03.144 -191,-351,-469,-543,-583,-595,-594, |
          ↪ -583,-563,-532,-507,-484,-459,-436,-408,-380
```

```
  2023-06-27T21:27:03.264 -358,-346,-339,-329,-321,-318,-313, |
      ↪ -312,-304,-296,-296,-294,-292,-285,-271,-260
  2023-06-27T21:27:03.384 -256,-257,-252,-234,-224,-228,-234, |
      ↪ -228,-215,-202,-193,-185,-188,-182,-177,-165
  2023-06-27T21:27:03.504 -152,-154,-163,-166,-171,-180,-203,-222,|
 -216,-145,-45,38,73,59,2,-55
  2023-06-27T21:27:03.656 -105,-118,-98,-86,-94,-104,-123,-116, |
      ↪ -77,-29,-34,-388,-787,59,2478,4573
  2023-06-27T21:27:03.774 4301,2044,43,-401,-73,-133,-441,-487, |
      ↪ -415,-397,-378,-345,-320,-288,-253,-216
  2023-06-27T21:27:03.894 -176,-143,-115,-67,-15,30,57,91,153,216,270,|
 318,376,444,506,554
  2023-06-27T21:27:04.013 589,598,588,515,371,175,-45,-254,-429, |
      ↪ -558,-628,-660,-665,-666,-655,-632
  2023-06-27T21:27:04.133 -608,-570,-535,-515,-501,-492,-466,-444, |
      ↪ -425,-415,-409,-402,-404,-402,-403,-390
  2023-06-27T21:27:04.254 -378,-367,-367,-366,-364,-350,-332,-318, |
      ↪ -314,-310,-302,-287,-275,-268,-254,-241
  2023-06-27T21:27:04.404 -235,-244,-249,-238,-229,-235,-261,-282, |
      ↪ -266,-202,-116,-33,8,7,-49,-107
  2023-06-27T21:27:04.524 -139,-134,-117,-125,-144,-142,-150,-139, |
      ↪ -103,-65,-68,-385,-727,118,2435,4386
  2023-06-27T21:27:04.644 4000,1719,-202,-510,-59,-75,-396,-450, |
      ↪ -364,-344,-333,-303,-269,-237,-204,-165
  2023-06-27T21:27:04.765 -124,-88,-60,-20,32,76,120,165,221, |
      ↪ 281,343,412,479,550,614,664
  2023-06-27T21:27:04.883 699,711,689,609,442,201,-68,-309,-485, |
      ↪ -602,-680,-720,-722,-698,-670,-651
  2023-06-27T21:27:05.004 -632,-609,-569,-536,-508,-484,-461, |
      ↪ -441,-426,-418,-418,-416,-402,-384,-373,-367
  2023-06-27T21:27:05.155 -372,-365,-350,-343,-340,-334,-323, |
      ↪ -309,-302,-288,-283,-268,-259,-247,-229,-225
  2023-06-27T21:27:05.294 -227,-223,-222,-232,-262,-289,-254, |
      ↪ -151,-37,36,43,11,-42,-91,-122,-123
  2023-06-27T21:27:05.410 -104,-108,-113,-122,-122,-87,-58,-10, |
      ↪ -207,-737,-439,1642,4255,4989,3231,860
  2023-06-27T21:27:05.514 -222,-138,-114,-422,-519,-453,-445,-431, |
      ↪ -399,-374,-336,-309,-278,-247,-210,-171
  2023-06-27T21:27:05.634 -124,-77,-37,-7,32,81,133,184,225,277, |
      ↪ 339,392,427,447,459,452
  2023-06-27T21:27:05.753 403,290,116,-80,-270,-421,-522,-585,-616, |
      ↪ -622,-614,-591,-573,-551,-529,-502
Update local data   2023-06-27 19:27:05 +0000
  2023-06-27T21:27:05.906 -474,-453,-430,-410,-395,-381,-367,-359, |
      ↪ -353,-352,-344,-338,-333,-323,-318,-311
  2023-06-27T21:27:06.024 -298,-291,-281,-278,-270,-256,-248,-238, |
      ↪ -234,-227,-217,-207,-195,-188,-190,-190
  2023-06-27T21:27:06.144
      ↪ -187,-185,-200,-233,-250,-213,-118,-12,57,70,34,-24,-72,-101,-95,-84
      ↪
  2023-06-27T21:27:06.264 -90,-99,-112,-115,-85,-46,7,-211, |
      ↪ -734,-405,1678,4235,4904,3142,818,-230
  2023-06-27T21:27:06.384 -143,-129,-411,-497,-439,-438,-434, |
      ↪ -404,-369,-332,-298,-264,-232,-201,-169,-129
  2023-06-27T21:27:06.504 -90,-52,-15,28,75,128,172,214,260,311,360,|
```

```
355 394,416,431,429,392
356  2023-06-27T21:27:06.654 294,126,-75,-265,-408,-513,-574,-606, |
         ↪ -616,-610,-594,-574,-546,-526,-505,-487
357  2023-06-27T21:27:06.774 -466,-448,-433,-422,-411,-398,-387,-380, |
         ↪ -378,-374,-365,-352,-346,-345,-344,-339
358  2023-06-27T21:27:06.894 -327,-313,-301,-292,-282,-270,-257, |
         ↪ -248,-243,-240,-234,-227,-220,-207,-206,-223
359  2023-06-27T21:27:07.015 -260,-281,-244,-143,-33,31,39,10, |
         ↪ -44,-88,-108,-108,-106,-114,-114,-123
360  2023-06-27T21:27:07.134 -133,-107,-67,-5,-227,-753,-381, |
         ↪ 1737,4238,4801,2983,704,-258,-134,-127,-416
361  2023-06-27T21:27:07.254 -501,-441,-435,-420,-387,-363,-327, |
         ↪ -295,-263,-229,-199,-159,-114,-73,-38,-6
362  2023-06-27T21:27:07.404 39,88,134,174,208,260,316,362,392,401,|
363 411,414,375,265,85,-115
364  2023-06-27T21:27:07.524 -294,-434,-531,-600,-637,-641,-630, |
         ↪ -608,-587,-568,-551,-529,-498,-475,-462,-453
365  2023-06-27T21:27:07.644 -445,-426,-412,-404,-401,-395,-386, |
         ↪ -377,-374,-372,-366,-356,-345,-339,-334,-325
366  2023-06-27T21:27:07.764 -314,-306,-298,-288,-281,-276,-267, |
         ↪ -257,-246,-243,-249,-252,-264,-290,-323,-321
367  2023-06-27T21:27:07.884 -251,-137,-43,-5,-37,-102,-162,-191, |
         ↪ -199,-200,-213,-221,-223,-239,-230,-202
368  2023-06-27T21:27:08.004 -138,-238,-746,-809,804,3419,4697, |
         ↪ 3452,1087,-292,-331,-199,-464,-637,-583,-565
369  2023-06-27T21:27:08.156 -566,-526,-488,-458,-438,-415,-382, |
         ↪ -343,-307,-262,-225,-186,-138,-87,-42,-11
370  2023-06-27T21:27:08.274 25,75,136,194,236,266,288,308,329, |
         ↪ 313,220,51,-160,-364,-520,-625
371  2023-06-27T21:27:08.394 -679,-707,-717,-718,-706,-677,-651, |
         ↪ -622,-598,-573,-549,-535,-519,-501,-484,-472
372  2023-06-27T21:27:08.513 -469,-465,-447,-428,-427,-434,-431, |
         ↪ -406,-382,-373,-366,-345,-319,-309,-314,-311
373  2023-06-27T21:27:08.634 -296,-271,-259,-249,-231,-240,-269, |
         ↪ -299,-300,-228,-112,-5,53,45,-5,-62
374  2023-06-27T21:27:08.754 -97,-103,-86,-83,-96,-106,-113,-87, |
         ↪ -40,8,-35,-466,-699,555,3088,4684
375  2023-06-27T21:27:08.905 3763,1384,-190,-237,89,-112,-398, |
         ↪ -380,-312,-302,-271,-241,-213,-174,-141,-107
376  2023-06-27T21:27:09.023 -71,-39,0,41,87,133,179,226,265, |
         ↪ 310,366,420,480,541,587,619
377  2023-06-27T21:27:09.144 634,641,616,511,317,70,-167,-353,-492, |
         ↪ -572,-606,-608,-594,-581,-558,-531
378  2023-06-27T21:27:09.266 -500,-476,-454,-429,-412,-404,-393,-384, |
         ↪ -368,-356,-356,-349,-344,-332,-316,-306
```

Listing B.1: ECG value logs.

B.3 Faros implementation

The initial Faros integration project branch can be found **here**.

The most relevant types will be presented in the appendix, namely the Bluetooth-Manager class and the ECGPacket model have been created.

ECGPacket Model

```
[language=Swift,basicstyle=\ttfamily,columns=fullflexible,postbreak=\mbox{\
    textcolor{red}{$\hookrightarrow$}\space},breaklines=true,caption=ECG Packet
     type interface,label=ecgpackettype]
import Foundation

public struct ECGPacket {
    let signature: String
    let flag: UInt8
    let packetNumber: UInt32
    let ecgData: [Int16]
    let accelerometerData: (x: Int16, y: Int16, z: Int16)
    let marker: UInt16
    let rrInterval: Int16?
    let temperature: UInt16?
    let batteryVoltage: UInt16?
    let timestamp: UInt32?
    let pacemakerEvents: [PacemakerEvent]
    let reserved: [UInt8]
    let padding: [UInt8]?
    let checksum: UInt16
}

public extension ECGPacket {
    init?(data: Data) {
        guard data.count >= 25 else { return nil }

        signature = String(bytes: data[0..<3], encoding: .ascii) ?? ""
        guard signature == "MEP" else { return nil }

        flag = data[3]
        packetNumber = data[4...7].withUnsafeBytes { $0.load(as: UInt32.self)
            }.byteSwapped

        // ECG data
        ecgData = [
            Int16(bigEndian: data.subdata(in: 8..<10).withUnsafeBytes { $0.load
                (as: Int16.self) }),
            Int16(bigEndian: data.subdata(in: 10..<12).withUnsafeBytes { $0.
                load(as: Int16.self) }),
        ]
        // Accelerometer data
        let accelX = data[10...11].withUnsafeBytes { $0.load(as: Int16.self) }.
            bigEndian
        let accelY = data[12...13].withUnsafeBytes { $0.load(as: Int16.self) }.
            bigEndian
        let accelZ = data[14...15].withUnsafeBytes { $0.load(as: Int16.self) }.
            bigEndian
        accelerometerData = (accelX, accelY, accelZ)

        // Marker data
        marker = data[16...17].withUnsafeBytes { $0.load(as: UInt16.self) }.
```

```
            bigEndian

        // RR-interval
        if flag & 1 == 1 {
            rrInterval = Int16(bigEndian: data.subdata(in: 20..<22).
                withUnsafeBytes { $0.load(as: Int16.self) })
        } else {
            rrInterval = nil
        }

        // Assuming the temperature, batteryVoltage, and timestamp are always
            available
        temperature = UInt16(bigEndian: data.subdata(in: 22..<24).
            withUnsafeBytes { $0.load(as: UInt16.self) })
        batteryVoltage = UInt16(bigEndian: data.subdata(in: 24..<26).
            withUnsafeBytes { $0.load(as: UInt16.self) })
        timestamp = UInt32(bigEndian: data.subdata(in: 26..<30).withUnsafeBytes
             { $0.load(as: UInt32.self) })

        // Assuming the pacemakerEvents are not always available
        if data.count > 30 {
            pacemakerEvents = [
                PacemakerEvent(data: data.subdata(in: 30..<32)),
                PacemakerEvent(data: data.subdata(in: 32..<34)),
            ]
        } else {
            pacemakerEvents = []
        }

        let reservedStartIndex = 30
        let reservedEndIndex = reservedStartIndex + 5
        reserved = Array(data[reservedStartIndex..<reservedEndIndex])

        let paddingStartIndex = reservedEndIndex
        let paddingEndIndex = paddingStartIndex + 1
        padding = Array(data[paddingStartIndex..<paddingEndIndex])

        checksum = UInt16(bigEndian: data.subdata(in: data.count - 2..<data.
            count).withUnsafeBytes { $0.load(as: UInt16.self) })

    }
}

struct PacemakerEvent {
    let isDetected: Bool
    let timestamp: UInt16

    init(data: Data) {
        isDetected = (data[0] & 0x80) != 0
        timestamp = UInt16(data[0] & 0x7F) | UInt16(data[1]) << 7
    }
}
```

.

BluetoothManager class

```
[language=Swift,basicstyle=\ttfamily,columns=fullflexible,postbreak=\mbox{\
    textcolor{red}{$\hookrightarrow$}\space},breaklines=true,caption=Faros
    Bluetooth Manager,label=farosbluetotthmanager]
public class BluetoothManager: NSObject {
    typealias PeripheralContinuation = CheckedContinuation<CBPeripheral, Error>

    private var centralManager: CBCentralManager!
    private var connectingPeripheral: CBPeripheral?

    private var connectingContinuation: PeripheralContinuation?

    var peripheralContinuation: AsyncStream<CBPeripheral>.Continuation!
    var ecgPacketContinuation: AsyncStream<ECGPacket>.Continuation!

    lazy var peripheralStream: AsyncStream<CBPeripheral> = {
        .init { cont in
        peripheralContinuation = cont
        }}()

    public lazy var ecgPacketsStream: AsyncStream<ECGPacket> = {
        .init { cont in
            ecgPacketContinuation = cont
        }}()

    public override init() {
        super.init()
        centralManager = CBCentralManager(delegate: self, queue: nil)
    }
}

//MARK: Public Interface
public extension BluetoothManager {
    func scanAvailableDevices() -> AsyncStream<CBPeripheral> {
        centralManager.scanForPeripherals(withServices: nil)
        return peripheralStream
    }

    func connectToPeripheral(_ peripheral: CBPeripheral) async throws ->
        CBPeripheral {
        return try await withCheckedThrowingContinuation { cont in
            connectingContinuation = cont
            centralManager.connect(peripheral)
            connectingPeripheral = peripheral
        }
    }
}

//MARK: CBCentralManagerDelegate
extension BluetoothManager: CBCentralManagerDelegate {
    public func centralManagerDidUpdateState(
        _ central: CBCentralManager
    ) {
        if central.state == .poweredOn {
```

```
51              print("  powered on")
52  //            centralManager.scanForPeripherals(withServices: nil)
53          }
54      }
55
56      public func centralManager(
57          _ central: CBCentralManager,
58          didDiscover peripheral: CBPeripheral,
59          advertisementData: [String : Any], rssi RSSI: NSNumber
60      ) {
61          peripheralContinuation.yield(peripheral)
62      }
63
64      public func centralManager(
65          _ central: CBCentralManager,
66          didConnect peripheral: CBPeripheral
67      ) {
68          if peripheral == connectingPeripheral {
69              connectingContinuation?.resume(returning: peripheral)
70          } else {
71              connectingContinuation?.resume(throwing: BluetoothError.
                    failedToConnectToGivenDevice)
72          }
73      }
74
75      public func centralManager(
76          _ central: CBCentralManager,
77          didFailToConnect peripheral: CBPeripheral,
78          error: Error?
79      ) {
80          if let error = error {
81              connectingContinuation?.resume(throwing: error)
82          } else {
83              connectingContinuation?.resume(throwing: BluetoothError.
                    failedToConnect)
84          }
85      }
86  }
87
88  extension BluetoothManager: CBPeripheralDelegate {
89      public func peripheral(
90          _ peripheral: CBPeripheral,
91          didDiscoverIncludedServicesFor service: CBService,
92          error: Error?
93      ) {
94
95      }
96
97      public func peripheral(
98          _ peripheral: CBPeripheral,
99          didDiscoverCharacteristicsFor service: CBService,
100         error: Error?
101     ) {
102
103     }
```

```
104 }
105
106 enum BluetoothError: Error {
107     case  failedToConnect
108     case failedToConnectToGivenDevice
109 }
```

.

APPENDIX C

Appendix

Appendix C contains documents relevant to this project.

C.1 EpiHeartMonitor Application

This sections specifies the URL to the source code of the EpiHeartMonitor application. It can be found in the following link: **EpiHeartMonitor Github repository**.

Bibliography

[AB20] Roberta Avanzato and Francesco Beritelli. "Automatic ECG Diagnosis Using Convolutional Neural Network." In: *Electronics* 9 (June 2020), page 951. DOI: 10.3390/electronics9060951.

[Appa] Apple. *Async stream continuation*. https://developer.apple.com/documentation/swift/asyncstream/continuation.

[Appb] Apple. *Checked continuation*. https://developer.apple.com/documentation/swift/checkedcontinuation.

[Appc] Apple. *Core Bluetooth*. https://developer.apple.com/documentation/corebluetooth.

[Appd] Apple. *Human Interface Guidelines Typography*. https://developer.apple.com/design/human-interface-guidelines/typography.

[Appe] Apple. *Scaling Fonts Automatically*. https://developer.apple.com/documentation/uikit/uifont/scaling_fonts_automatically.

[Appf] Apple. *Supporting VoiceOver in your app*. https://developer.apple.com/documentation/accessibility/supporting_voiceover_in_your_app/.

[Appg] Apple. *Widget Kit Documentation*. https://developer.apple.com/documentation/widgetkit.

[Bau+16] Mathias Baumert et al. "QT interval variability in body surface ECG: measurement, physiological basis, and clinical value: position statement and consensus guidance endorsed by the European Heart Rhythm Association jointly with the ESC Working Group on Cardiac Cellular Electrophysiology." en. In: *Europace* 18.6 (January 2016), pages 925–944.

[CAC] CACHET. *Create file*. https://cans.cachet.dk/dev/swagger-ui/index.html.

[ccg] ccgus. *FMDB*. https://github.com/ccgus/fmdb.

[Dev11] Orrin Devinsky. "Sudden, unexpected death in epilepsy." In: *New England Journal of Medicine* 365.19 (2011), pages 1801–1811. DOI: 10.1016/S1474-4422(16)30158-2.

[Fis+05] Robert S. Fisher et al. “Epileptic Seizures and Epilepsy: Definitions Proposed by the International League Against Epilepsy (ILAE) and the International Bureau for Epilepsy (IBE).” In: *Epilepsia* 46.4 (2005), pages 470–472. DOI: https://doi.org/10.1111/j.0013-9580.2005.66104.x. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.0013-9580.2005.66104.x. URL: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.0013-9580.2005.66104.x.

[Fis+14] Robert S. Fisher et al. “ILAE Official Report: A practical clinical definition of epilepsy.” In: *Epilepsia* 55.4 (2014), pages 475–482. DOI: 10.1111/epi.12550.

[Har] Harvard. *Test Accessibility with an iPhone*. https://accessibility.huit.harvard.edu/practices.

[Har+17] Richard Harte et al. “A Human-Centered Design Methodology to Enhance the Usability, Human Factors, and User Experience of Connected Health Systems: A Three-Phase Methodology.” In: *JMIR Human Factors* (2017).

[Hes+11] D C Hesdorffer et al. “Estimating risk for developing epilepsy: a population-based study in Rochester, Minnesota.” en. In: *Neurology* 76.1 (January 2011), pages 23–27.

[Jef] Jeff Sauro. *5 Ways to Interpret a SUS Score*. https://measuringu.com/interpret-sus-score/.

[Kod] Kodeco. *SQLite with Swift tutorial*. https://www.kodeco.com/6620276-sqlite-with-swift-tutorial-getting-started.

[Kri] Kristaps Grinbergs. *Embracing the Dynamic Type*. https://fassko.medium.com/embracing-the-dynamic-type-d8df6701aa19.

[LS01] Samden D. Lhatoo and Josemir W.A.S. Sander. “Cause-Specific Mortality in Epilepsy.” In: *Current Opinion in Neurology* 14.2 (2001), pages 189–195. DOI: 10.1111/j.1528-1167.2005.00406.x.

[Maca] MacRumors. *How to Enable Live Activities on iPhone*. https://www.macrumors.com/how-to/enable-live-activities-ios/.

[Macb] MacStories. *iOS 16 Lock Screen Widgets*. https://www.macstories.net/stories/ios-16-lock-screen-widgets-the-macstories-roundup/.

[Mas] Masha Panchenko. *Measuring the Intangible. Usability Metrics*. https://www.eleken.co/blog-posts/usability-metrics.

[Mik] Mikko Eronen. *MovesenseApi-iOS*. https://github.com/mikkojeronen/MovesenseApi-iOS.

[Mon] MongoDB. *Realm*. https://realm.io/.

[Mov] Movesense. *Movesense API Documentation*. https://www.movesense.com/docs/esw/api_reference.

[Mov20] Movesense. *OP174 MOVESENSE MD User Guide*. English. Movesense. December 14, 2020. 24 pages.

[Nas11] Lina Nashef. "Sudden unexpected death in epilepsy: terminology and definitions." In: *Epilepsia* 52.Suppl. 7 (2011), pages 6–8. DOI: 10.1111/j.1528-1157.1997.tb06130.x.

[ND88] Donald A. Norman and Stephen W. Draper. "User Centered System Design: New Perspectives on Human-Computer Interaction." In: 1988.

[NHM07] Lina Nashef, Neeti Hindocha, and Andrew Makoff. "Risk factors in sudden death in epilepsy (SUDEP): the quest for mechanisms." en. In: *Epilepsia* 48.5 (April 2007), pages 859–871.

[Noa] Noah Martin. *Supporting Dynamic Type at Airbnb*. https://medium.com/airbnb-engineering/supporting-dynamic-type-at-airbnb-b47c68b0c998.

[Org19] World Health Organization. *Epilepsy: a public health imperative: summary*. Technical documents. 2019, 12 p.

[Poi] Point-Free. *Dependencies*. https://github.com/pointfreeco/swift-dependencies.

[Pro] Proxyman. *iOS Proxyman setup*. https://docs.proxyman.io/debug-devices/ios-device.

[Ryv+13] Philippe Ryvlin et al. "Incidence and mechanisms of cardiorespiratory arrests in epilepsy monitoring units (MORTEMUS): a retrospective study." en. In: *Lancet Neurol* 12.10 (September 2013), pages 966–977.

[Sta] Stark. *What is AA vs AAA*. https://www.getstark.co/support/education/what-is-aa-vs-aaa.

[Stea] Stefan Kieleithner. *Improving Dynamic Type Support*. https://pspdfkit.com/blog/2018/improving-dynamic-type-support/.

[Steb] Stephen Celis. *SQLite.swift*. https://github.com/stephencelis/SQLite.swift.

[THF14] David J. Thurman, Dale C. Hesdorffer, and Jacqueline A. French. "Sudden unexpected death in epilepsy: Assessing the public health burden." In: *Epilepsia* 55.10 (2014), pages 1479–1485. DOI: https://doi.org/10.1111/epi.12666.

[Thu+11] David J. Thurman et al. "Standards for Epidemiologic Studies and Surveillance of Epilepsy." In: *Epilepsia* 52.Suppl. 7 (2011), pages 2–26. DOI: 10.1111/j.1528-1167.2011.03121.x.

[Usa] Usability.gov. *User-Centered Design Basics*. https://www.usability.gov/what-and-why/user-centered-design.html.

{2.7182818284}ϑφερτυθιοπσδφγηξκλ