University of Southampton

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

Use of IoT to enhance rehabilitation monitoring in leg surgery patients

by Demetris Mouzouris

September 05, 2019

Supervisor: Professor Neil White

Examiner: Dr Gary B Wills

A dissertation submitted in partial fulfilment of the degree of

MSc Computer Science

University of Southampton

Abstract

Faculty of Engineering and Physical Sciences

Electronics and Computer Science

School of Electronics and Computer Science

A dissertation submitted in partial fulfilment of the degree of

MSc Computer Science

By

Demetris Mouzouris

This dissertation has been written to demonstrate how an application can be utilised to aid specialised medical personnel in monitoring patients with severe leg injuries who are either in the process of getting surgery or are in the process of rehabilitation in the comfort of their own home. This application was developed in Android Studio and is a native application. It enables one-to-one communication between patients and doctors and communicates between the two individuals the status of the elevated leg, demonstrating whether its location is higher or lower by utilising multiple algorithms, also displaying the total elevation time of the patient throughout the day. There are multiple algorithms involved in this application that make this project feasible. Four fusion algorithms are implemented ranging from Kalman Filter, Madgwick and Mahony with a conversion algorithm from quaternions to Euler Angles and Google Functions displaying Fusion Matrixes with High and Low Pass Filters. These algorithms enable us to determine angles and orientation and alleviate sensor drift which could hinder the accuracy of our readings. The data outputted resulting from these algorithms are key in measuring elevation. In addition to the previous fusion algorithms an Elevation detection algorithm is created utilising the fusion data and implementing multiple threshold and stabilising algorithms which enable this project to accurately measure the users movements in order to detect whether the leg is at a state where the position is higher than the patients' heart or whether the user has moved the leg from that position to a lower state. The doctor can monitor the patients current and previous recordings and has the ability to measure post-surgery readings in matters of footsteps. A Step measurer algorithm has been developed in order to determine whether the patient has utilised the operated leg in order to aid its recovery.

Statement of Originality

- I have read and understood the ECS Academic Integrity information and the University's Academic Integrity Guidance for Students.
- I am aware that failure to act in accordance with the Regulations Governing Academic Integrity may lead to the imposition of penalties which, for the most serious cases, may include termination of programme.
- I consent to the University copying and distributing any or all of my work in any form and using third parties (who may be based outside the EU/EEA) to verify whether my work contains plagiarised material, and for quality assurance purposes.

You must change the statements in the boxes if you do not agree with them.

We expect you to acknowledge all sources of information (e.g. ideas, algorithms, data) using citations. You must also put quotation marks around any sections of text that you have copied without paraphrasing. If any figures or tables have been taken or modified from another source, you must explain this in the caption and cite the original source.

I have acknowledged all sources, and identified any content taken from elsewhere.

If you have used any code (e.g. open-source code), reference designs, or similar resources that have been produced by anyone else, you must list them in the box below. In the report, you must explain what was used and how it relates to the work you have done.

Madgwicks Algorithm [1] **Kalman Algorithm** [2] **Mahony Algorithm** [3] **The above algorithms have been ported from white papers and utilised for the project to retrieve fused sensory data.** **Open Source Sensor tag implementation on Android.** **Android Open Source packages:** **Graphview, Android plot** **CircleImageView** **Glide** **Retrofit - Notifications** **Kotlin, JAVA, XML** **Google Proprietary Functions and Sensor Functions** **The above open-source libraries and Languages have been utilised in this project.**

You can consult with module teaching staff/demonstrators, but you should not show anyone else your work (this includes uploading your work to publicly accessible repositories e.g. GitHub, unless expressly permitted by the module leader), or help them to do theirs. For individual assignments, we expect you to work on your own. For group assignments, we expect that you work only with your allocated group. You must get permission in writing from the module teaching staff before you seek outside assistance, e.g. a proofreading service, and declare it here.

I did all the work myself, or with my allocated group, and have not helped anyone else.

We expect that you have not fabricated, modified or distorted any data, evidence, references, experimental results, or other material used or presented in the report. You must clearly describe your experiments and how the results were obtained, and include all data, source code and/or designs (either in the report or submitted as a separate file) so that your results could be reproduced.

The material in the report is genuine, and I have included all my data/code/designs.

We expect that you have not previously submitted any part of this work for another assessment. You must get permission in writing from the module teaching staff before re-using any of your previously submitted work for this assessment.

The Literature Review and the Plan were utilised from ELEC6211 Project Preparation in order to aid in the completion of this project.

If your work involved research/studies (including surveys) on human participants, their cells or data, or on animals, you must have been granted ethical approval before the work was carried out, and any experiments must have followed these requirements. You must give details of this in the report and list the ethical approval reference number(s) in the box below.

Ergo Approval: 50707 shown clearly in the Appendix

ECS Statement of Originality Template, updated August 2018, Alex Weddell *aiofficer@ecs.soton.ac.uk*

Acknowledgements

I would like to take this opportunity to thank the people that have helped me to make this study feasible. My deepest gratitude and appreciation to my project supervisor Dr Neil White who has enabled me to work on such a fascinating sector of technology, and without his guidance and persistence, this project would have never been realised.

Secondly, I would also like to thank my second examiner Dr Gary Wills who was keen in enabling this project to progress to the next stage and with his guidance, motivated me to produce a better outcome.

Thirdly, I would also like to thank all the individuals who contributed to my research and have enabled me to test this application.

Table of Contents

Table of Tables

Table of Figures

Definitions and Abbreviations

FDA - Food and Drug Administration
BLE - Bluetooth Low Energy
UML - Unified Modelling Language
RSSI - Received Signal Strength Indicator
AHRS - Attitude and Heading Reference System
UUID - Universal Unique Identifier
GUI - Graphical User Interface

Chapter 1 Introduction

Currently, there are multiple technological advances that are being developed in order to aid medicinal practises ranging from artificial organs, by utilisation of 3D printing, to robotic surgeries and wireless sensors. One example is the new Apple Watch 4 which has been FDA approved, which enables the data collected through the Electro-Cardiogram (ECG) implemented on the bottom of the watch to be provided to the doctor. These readings can be considered as official medical data and a doctor can assess them in order to determine if there is any problem with the patient's readings [4]. Thanks to these advanced developments, medicinal practices have become less troublesome for patients and doctors and has enabled a higher success rate during operations. There are multiple injuries occurring daily with an increase in hospital admissions in the UK of 0.5 per cent yearly and an increase of 23.3 per cent since 2007-2008 according to the NHS [5]. One of the most common injuries being ankle spraining which is considered as one of the top injuries in the world according to WebMD [6], and a broken ankle is one of the most common fractures worldwide according to MedicineNet [7]. Some leg injuries could sometimes be more severe than others, with fractures at the lower extremities requiring immediate treatment in order to alleviate further problems to the patient's lower limbs. In such scenarios, there is a possibility that the doctor would instruct the patient to keep the leg elevated for a certain period of time which in turn would reduce the swelling and enable the doctor to operate on the leg. The problem with this scenario is that there are no current means of communication between the patient and the doctor and a phone call is the only way of contacting the doctor in times of need. There is a gap in the market for such an application that would implement hierarchical states such as doctors, patients and administrators that will enable these functions to take place, along with the readings of the patient to be communicated to the doctor, which will than enable the doctor to monitor how well the treatment is progressing and whether the patient is complying with what has been instructed by the doctor. SPHERE [8], is an example of a commercialised platform, although is relatively intrusive and outdated by the time of writing this dissertation, therefore an alternative implementation is required that will, replace previous attempts with a fully functional solution.

1.1 Motivation

Multiple advances taking place in the medicinal sector enable further progression in how the technology sector is a key part in enabling a higher success in medicinal practices, this has motivated the use of various techniques to improve their engagements within technology. Even though there has been increasing development in this sector there is a still need for further advancement into applications that include patients and doctors and how both individuals can interact with one another. This application can have multiple benefits in the medicinal sector as it will enable the doctors and patients to have instantaneous

communication, with regards to their readings and data and whether these readings are a cause of concern to either individual.

1.2 Project Aims and Objectives

1.2.1 Aims

There are currently multiple aims that are required for this implementation to be successful. The main aim will be to incorporate the sensor on an individual's leg that measures whether the leg was inclined or not. This will then define whether the leg is at a position that will reduce swelling. This will then enable the doctor to continue with the procedure in order to operate successfully on the patients' leg. The reason why this aim is considered as main is due to the lack of research in the area with limited applications available that have the ability to maintain communication between the patient and the doctor and have the ability to monitor the progress that has been made by the patient who is able to utilise the sensor at the comfort of their own home. "Medical sensors" is a brand-new field within the area of medicine and technology and only recently has been utilised for aiding the medical personnel to carry out their daily practices successfully and with higher accuracy.

The secondary aim would be to implement an application that will operate as an interface between the doctor and the patient in order to keep track of the tasks of the patient that involve his lower limb that requires attention. This will be an application that can inform the doctor how long has the patient kept the leg elevated throughout the day and to keep track of his footsteps in post-surgery situations which will enable the doctor to comprehend the rehabilitation state of the patients and how well is the state of the patients lower limb. The application has multiple requirements that need to be set in place before it is seen as fully functioning. One such requirement is ensuring the security of the medical data and how are they being stored within the database. The data needs to be displayed to both parties, the patient and the doctor, in a secure yet user-friendly manner that will include a graph that indicates the time and the elevation period of the patient. This data needs to be accurately redrawn from the patient's perspective to the doctor's perspective. This will then enable the doctor to determine if the leg has been elevated for a predetermined amount of time desired by the doctor and then the patient can be contacted in order to proceed with surgery.

The third aim of this project would be to implement an application that reduces the overall price of such applications, medical equipment that handle such delicate tasks and have been approved are relatively expensive. The implementation will also require that the device oppose minimal intrusion for the patients' home or the patient as an individual. With a limited requirement for maintenance and without or minimal discomfort to the patients in order to install the device.

1.2.2 Objectives

According to the aims listed above the objectives are the following:

Learning & Designing

- Research on how the project can be achieved.
- Learning multiple requirements and algorithms for the implementation of the project.
- Selection of the sensor which corresponds to the criteria we require.
- Design the application.
- Design the communication means if required and establish a wireless connection with the sensor.
- Design collection and processing of the data.
- Identify implementation process on patient.
- Design the testing parameters required for the process to be successful.

Implementing

- Programming in appropriate languages in order to utilise the data of the sensor.
- Identification of the communication means between the device and the mobile application, as well as bespoke programming for the devices to communicate.
- Adapt data of the sensor and clear all unnecessary data from the data being received by the devices.
- Produce visual data and graphs from the mobile application.
- Implement code that will be used for the specific purpose of raising the leg.
- Implement code that will be used for the specific purpose of measuring footsteps.
- Distinguish users and set security measures within the application.

Optimisation and Testing

- Optimise the code used by the communicating device and the sensor.
- Implement the sensor with the communication system
- Optimise the implementation in a finalised stage.
- Test the implementation in various scenarios in order to identify and alleviate errors or problems.
- Human Trials
- If time permits, additional features will be implemented.

Writing & Documenting

- Document process along the way of researching and implementation for later use.
- Write thesis displaying what has the project achieved.

1.3 Structure of Report

This dissertation will follow the following structure, in Chapter 2 the literature review will take place which will enable us to further understand the background study of this application. In Chapter 3 the requirements of this application will be specified in order to enable us to have a base of what is the functional and non-functional requirements for such an application. In Chapter 4 the Design of the application will be portrayed and followed by it is Chapter 5 which will emphasize on implementing the application. Chapter 6 will mainly focus on the results of this application. In Chapter 7 there will be an evaluation of what has been achieved by this application and the outcomes of the human trial results. Chapter 8 will be the conclusion of our project, followed by an Appendix for all the extra information and the source code.

Chapter 2 Literature Review

2.1 Clinical aspects

Current Medical research is being adapted to aid doctors in better patient diagnosis, whether it is from robotics or even data processing. Recently there has been a need for use of medical assistant sensor devices that could enable the doctor to operate human monitoring remotely and during out of scheduled hours enabling the doctor to maintain correspondence with the patient and monitor his behaviour and his improvement during recovery. Several sensor projects have been trialled such as SPHERE [8] although they have not yet been globally commercialised and utilised to their fullest. Currently one of the most amusing projects that have been recently developed is project "Emma" in collaboration with Microsoft, which implements a wrist bracelet that reduces tremors from people that have Parkinson's disease or any other types of disease that stimulates tremor to the patients' limbs [9]. Although some sectors of the medical field have little adaptation to current technological advances thus minimising the number of trials that have been undertaken in the process of monitoring.

2.2 Internet of Things in Medical Sectors

2.2.1 Internet of Things

"Internet of Things" is considered as devices that can connect to the Internet but also to one another [10]. This process has been growing relatively fast and as a trend in recent years but is currently being adapted to different business needs and evolving into sectors that are outside of the scope of Information Technology. However, there has recently been a big focus in medical applications and how IoT and how sensors have evolved to reach the standard they are currently in.

2.2.2 Medical Sensor Requirements

"Internet of Things" is constantly changing and adapting accordingly in order to tackle current and even future technological needs. These advancements in technology started to develop towards the medical field where they can make a great impact. According to A. Reisner [11], there have been many advancements that could easily be adapted to the needs of the medical sector and they have been attempting to produce accurate data that could significantly enhance the diagnosis that of the doctor. The paper by A. Reisner illustrates the requirements needed for the sensors to be successful in that field with a heavy attitude towards accuracy and the minimisation of false alarms, although medical sensors is not only about those two values, where the portraying of data is missing within this article as well as how the data need to be transmitted in real-time. Some intrusive implementations are being described such as the use of cameras [11], which is relatively invasive for the patient to have

a camera constantly monitoring their behaviour which could require further approval due to the Data Protection Act, from the individuals in order for their video feeds to be recorded along with their daily activities.

2.2.3 Sensors & Protocols

According to Jun Qi [12] different protocols were tested for their accuracy and it indicates problems that affect all sensors, the way the network should be set in order to process the data, which sensors should be used for different experiments and how they fair, as well as which platform should be used in order to enable availability. There is a protocol based on IPv6 and 6lowPan which utilises multiple sensors or swarms for long-distance recording of data, however this implementation has shown lack of security, this is due to the reason that attaching a public IP address to the sensors, which means it is browsable, would require the network to have a gateway that blocks any incoming connections to the sensor, as classified client data can be bridged. Another negative impact would be due to the world not being readily available to IPv6 it can cause interoperability between different devices to fail which could pose a risk to the implementation of such protocols [12].

The paper also indicates that data can either be locally processed or remotely according to the sensors that you will be using as well as the capability of the processor within the units and how AI can be incorporated to manipulate the data.

Once again, we see lack of how the data can be perceived by the doctors in order to aid the diagnosis although the protocols and algorithms required for an accurate implementation vary from the type of medical emergency. This can be a major research area as to how doctors perceive the data in order to understand them and what each data being presented to them is, since time-critical applications require real-time data. There is also a lack of identification of how the sensors will operate in different conditions as medical hospitals or even homes can be in inappropriate environments for the data to be retrieved or even recorded.

Wi-Fi and Bluetooth are two of the main means of sending data between devices and recent advancements in technology, enabling these two protocols to be implemented within our sensors. There are positives and negatives for each situation at stake, where Bluetooth will have a limited range as well as speed, on the other hand, Wi-Fi is more resource-intensive and has longer range and speed. Although Wi-Fi seems like the clear choice the fact that it is a resource-intensive protocol and it requires additional resources to be implemented within the sensor, this could alleviate our attempt to make the sensor non-intrusive in this situation. A subset of Bluetooth is Bluetooth LE which stands for Bluetooth Low Energy. It can be utilised for streaming data to a mobile device that could then be streamed to the cloud for later storage and utilisation. Another protocol that has recently been up and coming is 802.15.4 according to Arthur Gatouillat, [13] it has a deficiency in order of remote configuration in order to tailor the sensor accordingly and that there will be a need for the sensor to constantly stream data at states that the sensor is functional and non-functional, that would give a sense of starting and stopping readings to save energy, although could potentially increase the

values of false positives in our readings, the implementor then decides to move to Bluetooth LE where has managed to implement the sensor successfully.

2.2.4 Beyond the Sensors

There is a need for performance servers in order to enlist how the data will be treated and how the data will be stored to enable it to be streamed back to the mobile application that the doctor will be using.

According to Jao Rei [14], there have been different server tested in matters of load handling, the matter of requests and how well it operates in matters of real-time applications.

They have identified that the HBase type of Apache server is more able to correlate to medical data rather than a Cassandra Apache server as it can withstand better and provide real-time and accurate data to the doctor. Therefore, there is a battle of NoSQL databases that need to be tested in order to make sure that the database server which will be serving the information to our devices and the doctors are up to date and in real-time in order to aid the faster diagnosis since medical sensors are time-critical. Especially sensors that require data that are time related. There will be encryption implemented in the mobile application that will transmit the data to a secure database through secure/encrypted measures that will then be able to be received and formatted into a graph that would indicate the total time that the patient had their leg up during the period of their home rehabilitation.

2.2.5 The Receiving End

Figure 2-1 - The process of data traversal

The application has several aspects that need to be considered such as:

2.2.5.1 Requirements

The application would need to provide real-time data to the doctor in order to monitor the way the patient will be keeping the leg in an elevated position as well as the ability to contact the patient when he is not complying to the instructed rehabilitation, this could also be reversed, in a situation where the patient will have the ability to call the doctor in case of a

problem that requires medical assistance such as unbearable pain and therefore the doctor can guide the patient into remedial actions until an expert arrives on scene.
The doctor will have the ability to monitor his patient's recovery rate and will also inform him when the time that was required for the patient to remain in the rehabilitation state before surgery, has lapsed.
The user should be able to monitor their own performance in the rehabilitation state as well as their own doctor although without accessing any further information. This would be done using a login function that could authorise and authenticate each user to receive and share their data.

2.2.5.2 Platform

As there have been relative advancements in mobile applications the platform that would be easier is a Cross-Platform application which would require a single application to be developed that would interoperate between the two individuals and can be easily distributed to multiple devices. However, due to the nature of the application and the required APIs' of the application the support for native applications is much more diverse, enabling the application to be much more advanced than a Cross-Platform application thanks to the level of support it has received. The application will then be easily adapted to future needs and could be used as a base in developing the opposite platform version whether it is Android or IOS.

2.2.5.3 Security

Multiple secure measures will be put in place so that the data becomes encrypted, as well as the login function, this will be authorised and authenticated within a server and will allow different operations accordingly to their status of either Administrator, Doctor or Patient and grant them authorised access accordingly. Another measure is the transmission measure that will be implemented, there will be a secure connection to the database that the handheld device will be streaming data in an encrypted format, therefore, no eavesdropping will be available since patient data is of utmost confidentiality. As stated by Minho Shin [15] there is great importance in the security implemented and the overall robustness of the application. There is a need to authenticate for falsifying wrong data or even allowing the data to be in an eavesdropping situation and a risk analysis has been carried out to verify what can be altered to tackle this problem. Even the patient can be a threat to incorrect data with tampering of the sensor involuntarily. There are multiple ways available to tackle the different threats towards the successful implementation of the sensors as well as the accuracy and the false positives that can occur.

2.2.6 Efficiency

Due to the sensor running into low battery situations which minimises the size of the sensor in order to be implemented within a cast, this could potentially mean that there will be limited processing power of the sensor, therefore protocols will be adapted such as Bluetooth LE in

order to utilise less energy and enable the sensor to last for a longer period of time while data will be transmitted and received at either end. According to T.M.S. Sayeed [16] who has produced a prototype medical sensor for heart monitoring, he has utilised Bluetooth LE and has reduced his power usage to enable the battery to last longer so as the transmission is the most resource-intensive operation in matters of power usage in sensors. There is lack of research within sensors with efficiency in mind and further research is required to identify the extensions of different protocols that would enable the outcome that is projected as well as the power usage kept to a minimum during transmission or even processing using smart algorithms.

2.3 Implementing Sensors

There are multiple sensors available in the current market.

The board has several requirements for the application to be utilised for this medical situation. There are criteria that need to be passed and research carried out to rule out unusable sensors in matters of size, adaptability to their environment, the longevity of operation, their ease of use and their accuracy. As stated by Atis Elsts [8] the ease of installation, maintenance and usage are important aspects of sensors that are implemented for medical reasons although, in this example, there is an implementation of mesh networks and a whole range of different sensors which utilise two protocols to transmit data, Bluetooth LE and 6LowPan although this requires an intermediary device that will enable the forwarding between these two protocols. With that in mind some viable sensor options are:

2.3.1 Libelium Waspmote

Waspmote is a sensor utilising the ZigBee protocol to configure and transmit over the network, it consists of accelerometers as well as other sensors that can be implemented on the board. The negative about this board would be its size being overly large to be implemented into contained spaces such as a cast.

2.3.2 SparkFun

They produce sensors along with development boards called “The Thing” which can incorporate various sensors, although this could prove ineffective due to the size and the complexity of implementing such a sensor.

2.3.3 Custom Sensor

A custom sensor would be a suitable approach rather than a ready-made one although a negative impact would be that it drives costs higher. This makes these implementations not only difficult to implement but also very costly and time-consuming to customise.

2.3.4 Arduino Nano 33 BLE Sense

One of the latest sensors to hit the market is the Nano BLE sense board which has all the appropriate sensors required for this project to become a reality although due to the reason of its late release date being Mid-August 2019, it can't be utilised for the matters of this project. And can be recommended for having up to date sensors that can surpass some of our previous recommendations. It comprises of all of Arduinos' functionalities with a motion sensor implemented on the board and headers for any peripherals you would require.

2.3.5 Texas Instruments – CC2650-STK

CC2650-STK is mainly based on Bluetooth LE which has recently become a substantially robust standard and has been utilised several times in medical sensors. This enables it to stream data through Bluetooth devices, to the database, and in turn transmitted to the doctor for further analysis. The negative about it is range and battery life as the size of this sensor reduces resources on the device, this would make the data to be vulnerable to interruptions when the subject is not in range with the receiving device.

Figure 2-2 – Texas Instruments -CC2650-STK Sensor

Out of all the sensors, TI-CC2650-STK stood out as the most viable option due it being robust with a whole variety of sensors as well as the price being at an acceptable rate.

2.4 Algorithms

There are currently multiple algorithms that can be utilised for the project to take form.
A few of the algorithms that can be tested for this project are:

2.4.1 Madgwicks IMU AHRS

This is an algorithm that has been developed back in 2010 that was developed by Sebastian O.H Madgwicks which was done as a part of his PhD research [17]. The problem with this algorithm is that the algorithm was never maintained and contained multiple bugs. It was then rehosted by x-io.co.uk and has been implemented with a DCM filter in the quaternion form [14]. The algorithm requires readings from three separate sensors the Accelerometer the

Gyroscope and the Magnetometer. It is based in quaternion forms and enables the Magnetometer and the Accelerometer to be utilised in a gradient descent algorithm in order to compute the error produced by the gyroscope as a quaternion derivative. Its performance was matched to Kalman based sensor fusion algorithms. And operates in a way that reduces power consumption, therefore, providing higher usage capabilities with prolonged usage.

2.4.2 Mahony IMU AHRS

Mahoney has the same concept and utilises Quaternions just like Madgwicks although it has some minor differences. Mahony utilises a proportional along with an integral controller to correct the gyroscope bias, whereas Madgwicks uses only a proportional controller.

2.4.3 Sensor Fusion with High & Low Pass filter

This method is implemented by utilising proprietary functions provided by Google rather than standard AHRS algorithms. These functions can calculate those readings for us within the Android studio development application, these proprietary functions enable the elimination of drift produced by the gyroscope, and reduce the computational power required for the application to operate as these implementations have been produced to operate on smartphone devices [18]. By applying these functions, filters would then be utilised on the orientation matrixes of this application which would enable us to get more accurate and stable data to enable our readings to take place and output correct results.

2.4.4 Sensor fusion with Kalman Filter

Kalman filter is a relatively mastered algorithm that is being utilised throughout the world from NASA to space aircraft. The simplified way that it operates is based on the state of the sensor or any other information you are providing in the previous state, the Kalman filter has the ability to utilise the previous state to make an educated guess of what the next state could be. It utilises what we call as Gaussians to constantly predict the next movement that is expected from the sensors based on all its previous state data that has been collected. In layman's terms, it is making an educated guess for its next move based on the state that it was previously in, and it can be considered a prediction-update formulation [19].

2.4.5 Euler Angles

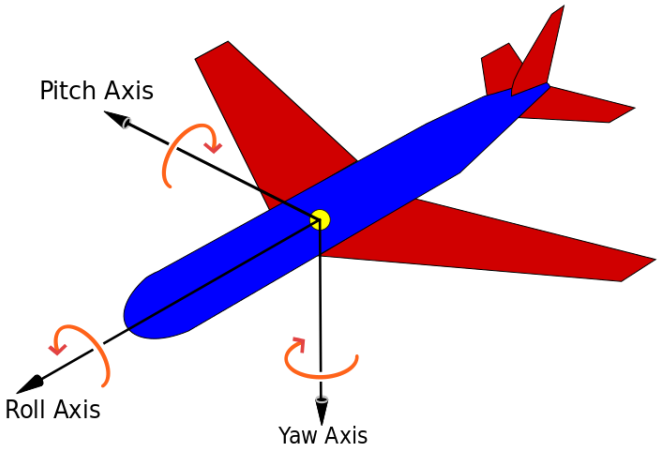

Figure 2-3 Euler angles in aeroplane simplified Source(Wikipedia: Euler Angles [20])

It is used to indicate the orientation of a rigid body based on fixed coordinates. Euler angles are a representation in terms of three consecutive rotations around predefined axes. A problem that although coincides by the utilisation of Euler angles is the known effect of Gimbal Lock, therefore, making it unsuitable for our implementation in order to display orientation and position.

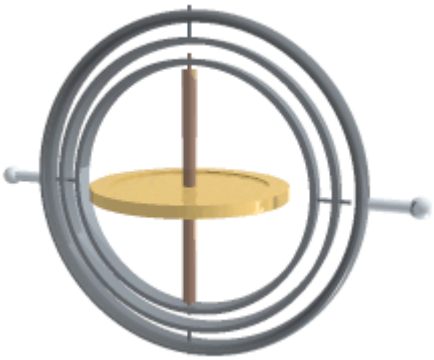

Figure 2-4 Gimbal lock fix with 4th rotation axis Source (Wikipedia: Gimbal Lock [21])

They are worth mentioning as they are the base of all the various algorithms that have been developed over the past years and have been fixed by the utilisation of a 4th parameter rather than pitch roll yaw an external factor that would then enable the rotation to be free of the gimbal lock effect.

2.5 Summary

There are different aspects of the "Internet of Things" available for medical sensors and different applications are better for different types of medical situations. The implementation of accuracy and efficiency are important in the lifetime of the sensors. Since medical data being of utmost importance, security is also one of the main implementations. Sensors are readily available for these implementations and there exists a whole range of them to choose from, but only a few of them can fit the criteria required for this situation, that will enable accurate results and most importantly to keep the cost prices low. Many algorithms have been developed throughout the years which can fit in the criteria of this project, therefore, there is some time required in further examining them and choosing the appropriate algorithm to utilise in order to produce clean and accurate data for the project to be a success.

Chapter 3 Requirements

For this project to be a success, there are multiple requirements that need to be set and implemented before enabling this project to operate. These requirements will be divided between functional and non-functional requirements based on the application that we will be developing.

3.1 Project scope

The scope of this project is to produce an application that will be utilised by multiple doctors and multiple patients. There will be multiple features implemented in the application that are not limited to retrieving processing and transmitting data. There will be an online communication channel between the patient and the doctor that will enable either party to communicate with the other individual if they are not complying with the instructions or whether the patient party has problems with their leg.

There will be a need for the application to interact with the sensor via BLE in order to not be intrusive with wires located on the patient. And there will be a hierarchical system implemented with requests that will enable the patient to have multiple doctors as well as the doctors to have multiple patients, therefore, allowing the patient to have multiple opinions on the process of his recovery or prior to the time of his surgery. There will be an algorithm implemented within the application that will process the data of the sensor in a way that it will start at the initial position of stabilised and down and after the initialisation process the patient will be able to move the leg up and it will detect it, following the same procedure will be the implementation where the leg is down. An online database will also be utilised to enable the data to be traversed through various users and enable the data to be easily visible to multiple individuals at the same time. The entire project is one application that is divided into multiple sections:

The detection algorithm, the fusion algorithm with filtration, the administrative functions, the patient functions and the doctor functions.

The detection algorithm will include:

- Multiple arithmetic notations that will detect signatures of up movements nature. As well as the reverse function will be utilised to detect down movement.
- There will be two different stabilising algorithms implemented one that takes into consideration the overall acceleration that the sensor is receiving.
- And one that detects whether the fused data in the concept of Pitch, Roll, Yaw are static in order to comprehend whether the sensor is stabilised. These functions will be utilised to detect when the user has made a move upwards and only when the user has stabilised his leg will the detection for down starts and vice versa.

The fusion algorithm with filtration will include:

- The fusion algorithm will include the gathering of the information into arrays that are being received from the sensor.
- The calculations required to get the orientation of the device.
- The alleviation of drift accumulated on these devices over time
- Not true filtration with the use of a variable alpha as indicated by the google sensor implementation.
- Conversion of the raw data received into appropriate and accurate data.

The administrative functions are:

- The approval of Doctors.

The patient functions are:

- Communication with the doctors
- Seeing all your doctors and their relevant information
- Using the application to connect to the Bluetooth device/sensor
- Altering their personal information and can request to become doctors.
- Upload function for step sessions.
- Requesting doctors for becoming their patient.

The doctors' functions are:

- Communication with the patients
- Seeing all your patients and their relevant information along with their data and the progress at any day they would like to choose.
- Altering their personal information and can alter between patient and doctor functions.
- Approving patients to come under their supervision.
- Monitoring for each patient and their relevant information.

3.2 Functional Requirements

There are currently multiple requirements in place the initial requirement is based on what the doctor has requested and the application should illustrate to the doctor the time of the leg elevation of the patient, along with the number of footsteps that had been completed by the patient since his surgery.

As in system requirements, they have been categorised in matters of priority which has the range between 1-5, 5 being of utmost priority and are illustrated in the following table:

Table 3-1 – System Requirements with Priority

Requirements ID	Client Type	Requirements	Priority
FR1	Patient/Doctor /Administrator	Functions for each individual must be differentiated to know what functions each user can carry out.	5
FR2	Patient/Doctor /Administrator	Forgot password functions put in place to return accounts to their rightful owners.	4
FR3	Doctor/Patient	Communication with the other party needs to be established	5
FR4	Doctor/Patient	The system should implement login and signup functions that enable the users to be recorded and records kept, maintaining accountability and assign data to individuals.	5
FR5	Doctor/Patient	See all assigned Doctors or patients to each individual.	5
FR6	Doctor/Patient	The data needs to be in a user-friendly and easy to access manner	3
FR7	Doctor/Patient	Need to modify the profile to see how the patient sees the application for troubleshooting purposes.	3
FR8	Doctor	Display the data of the patient to assess the situation prior to and after surgery. On real-time.	5
FR9	Doctor	Request approval for Doctors from Administrator	5
FR10	Doctor	There needs to be a method to approve patients to be able to see their data.	5
FR11	Patient	Make requests to the doctors for approval. And see their state.	5
FR12	Patient	See all available doctors and search and communicate with them to establish an initial contact before requesting their approval.	5
FR13	Patient	Some modification of the sensor readings can be implemented to enable reduced battery usage.	2
FR14	Developer	There needs to be an established database communication	5
FR15	Developer	The system should implement appropriate algorithms to detect the various requirements	5

3.3 Use Cases

The use case is utilised to illustrate all the various requirements in order to further understand how all these requirements are utilised for this application to work.

In the use case diagram, all users need to access through the same portal. Based on a function within that portal each user will have the ability to do various actions based on their access. The data from the patient are transmitted to the database. That gives them the ability to be stored and retrieved on the doctors' end in order to assess the situation. Approvals on doctors' end and patients end are required to enable the doctor to be visible to the patients as approved doctors and a secondary approval rate to input patients under the supervision of the specified doctor. Chat must be implemented to aid communication between the two parties.

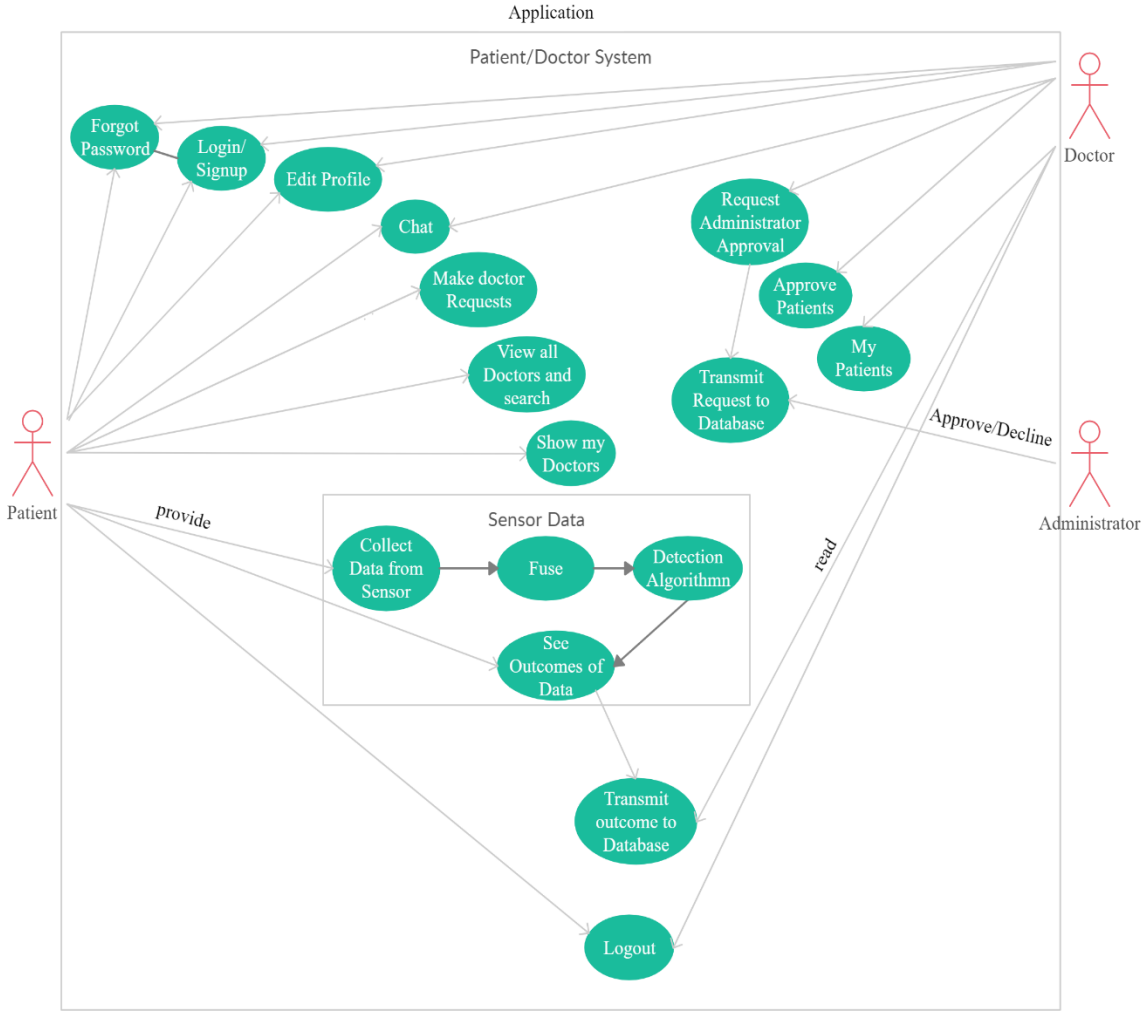

Figure 3-1– Use case diagram based on system requirements.

3.4 Non-Functional Requirements

Non-functional requirements are the requirements that are not related to any parts of the system being implemented although describes several constraints of the system.

The following table illustrates the non-functional requirements:

Table 3-2 – Non-Functional Requirements

Requirements ID	Category	Requirements
NFR1	Accessibility	The application must be available to the patient and the doctor 24/7
NFR 2	Reusability	The application and the source code are easy to modify for further endeavours or further explorations of the project.
NFR 3	Integrity, Privacy	Security must be implemented in the application to alleviate any problems or any malicious activities.
NFR 4	Readability	The application must display the data in user-friendly graphs that enable the doctor or patients to fully comprehend their readings
NFR 5	Performance	Must implement functions that are not expensive to enable the users to utilise that application and its features with much less cost in comparison to any alternatives.
NFR 6	Availability, Interoperability	The application needs to adapt to all available Android devices no matter of scale.
NFR 7	Fault Tolerance, Resilience	Backups need to be made of the database for protection cases.

All the non-functional requirements stated above indicate their need for performance, user experience, lowering expenses and security.

3.5 Constraints of Project

There are currently constraints involved within the application and the whole project. The constraints are listed in the following table:

Table 3-3 - Project Constraints

Constraint ID	Constraint
1	The project must be finalised before the 5th of September
2	Ergo approval is required to carry out the tests involving humans.
3	Identifying individuals willing to participate in the study.
4	Database reads and writes for the free version of Firebase

The constraints listed will be considered during the implementation of the project.

3.6 Risk Analysis

Risks Analysis is required to identify whether there are any problems that might arise and require immediate attention to the application. They are a key part in identifying preventing and reducing risks within the application.

Table 3-4 - Risk Analysis

Risk ID	Risk	Probability (1-5)	Severity (1-5)	Overall Risk (0-25)	Mitigation Action
1	Unknown Technical skills	3	5	15	Research-based on previous implementations and what is available in the general public in order to grab a concept of what is required prior beginning
2	Developer being ill	1	5	5	Exercise and eat healthy.
3	Project time scheduling overrun	2	4	8	Implementing minor goals weekly leading up to a big goal monthly in order to schedule timing appropriately
4	Sensor being Bricked	3	5	15	Before pushing firmware to the sensor tag monitor the code and identify if there are any mistakes, purchase the dev pack that enables wired connection rather than over the air with Bluetooth
5	Source Code misuse	2	3	6	Daily backups onto a private GitHub repository owned by the individual.
6	Requirements being unreachable due to technological conflicts	2	4	8	Identify possible problems before attempting to implement the requirement. Finding workarounds if possible or altering what the overall outcome could be

The risks identified will be averted by closely following the rules of these implementations and setting goals for this project to make it feasible. These risks will be constantly implemented and monitored in case they occur and will take appropriate action in order to avert them.

3.7 Summary

In this chapter we have identified all the requirements that this application and project will have along with their non-functional and functional requirements, the various constraints of the project have been identified, along with several risks with their mitigation process in order to make this project feasible.

Chapter 4 Design

This chapter will illustrate the design of our application, it will consist of UML diagrams indicating the structure of this application and the overall functions that will operate on the application based on each user type. UML diagrams are the most appropriate way to illustrate functionality and the design of your application [22]. In this chapter, UML diagrams will illustrate how each person perceives the application and the overall database and frontend layout.

4.1 System Architecture

The system will utilise Android Studio and will be developed in Java and Kotlin for functionality and XML for layout files. The application is divided into two sections Frontend and Backend. The frontend is further divided based on user types which are: Administrator, Doctor and Patient and their functionality. The application is an interface between all the various individuals involved and no alternative application is required to be downloaded in order to function. The backend database is comprised by the proprietary google Firebase Database – Cloud Firestore which is a NoSQL database enables us to utilise suitable functions to display the appropriate data after they had been provided by the user. The frontend is mainly enabling the data to appear and enables the differentiation between the users.

4.1.1 Hardware

Hardware is required for this application to operate, for this project we have utilised the Sensor Tag CC2650 sensor to retrieve motion data, and any android phone that has sufficient operating power and memory can also be utilised. In our case I have utilised the OnePlus 6T which has the following specifications:

- 8GB RAM
- Android v9
- Snapdragon 855
- 256 GB storage

The following diagram is how the functions are divided between the Users:

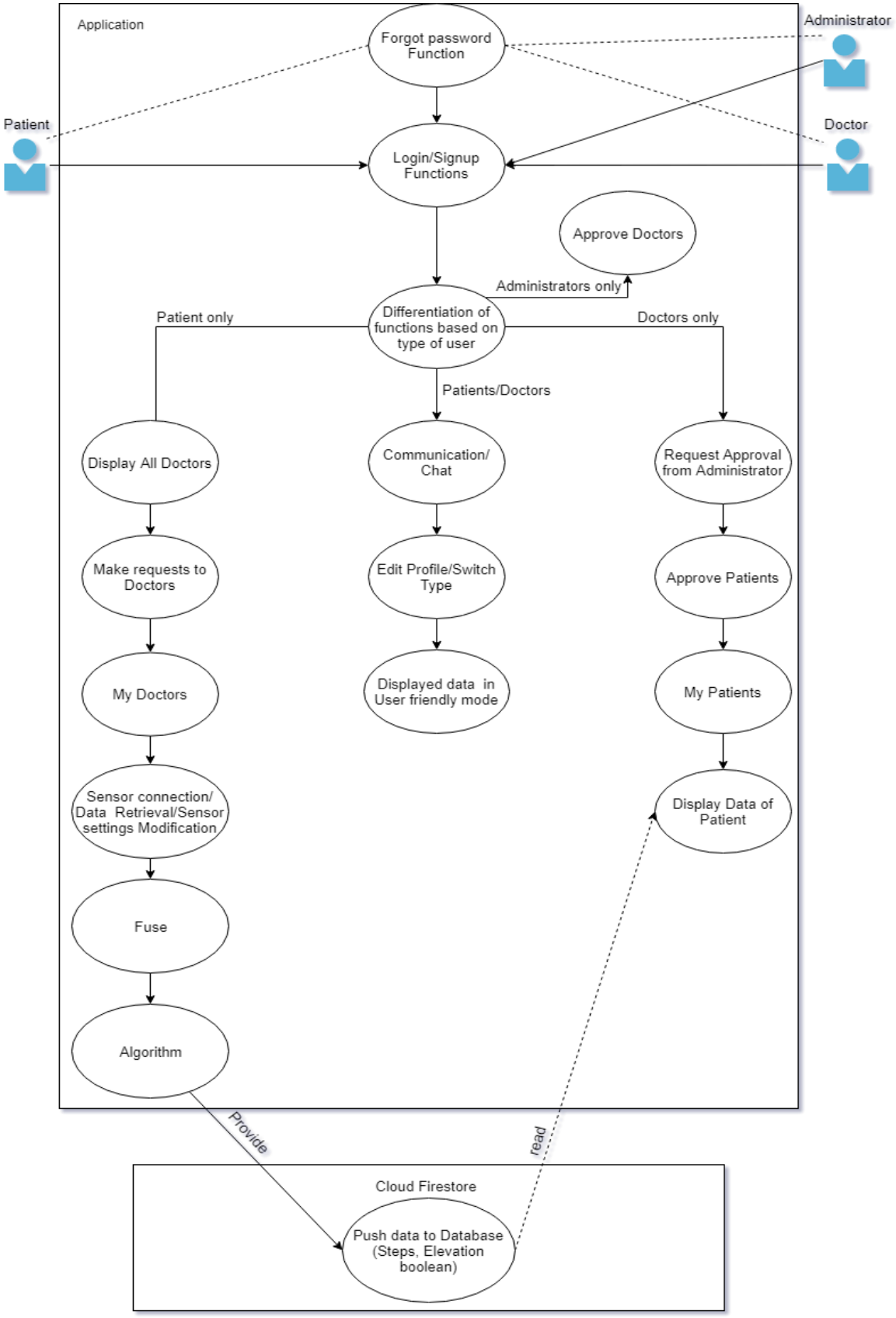

Figure 4-1 – Application Design

The following diagram indicates how the application is communicating between the Database and the Frontend:

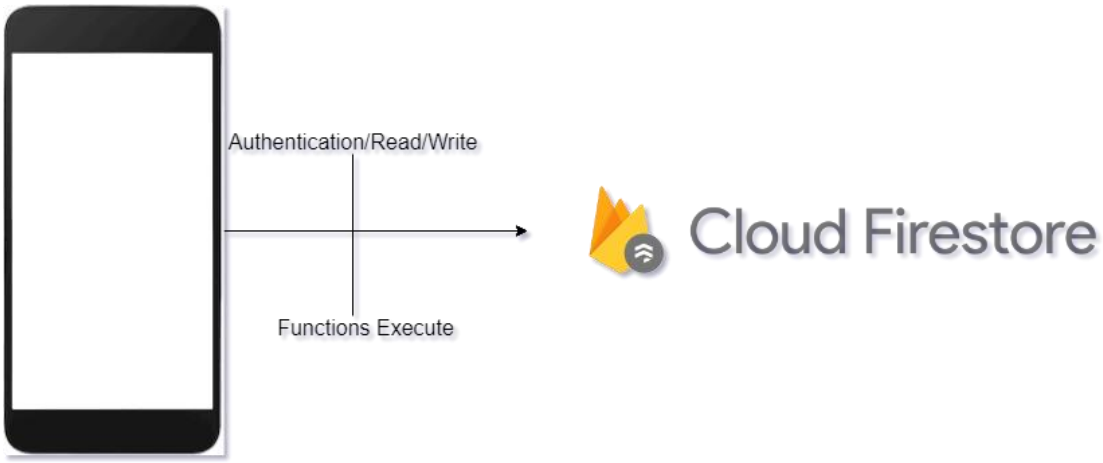

Figure 4-2 - Frontend to Backend

In the diagrams above the system which is running on the android phone communicates via OAuth. This function is automatically inserted through refactoring the application to communicate with the database. The application is then separated through a specialised function that identifies the type of user and based on that value the user is then able to carry out the appropriate functions assigned to their type.
Each type of user is greeted differently based on their type; the following indicates how each of the users' functions changes based on type.

4.2 Front End

As previously stated, the frontend is subdivided into sections based on users. Each user has its own functionalities which will then enable each user to make a variety of functions. The chapter will illustrate the structure of the application along with its interaction.

The following diagram indicates the UML of the application as seen by any user:

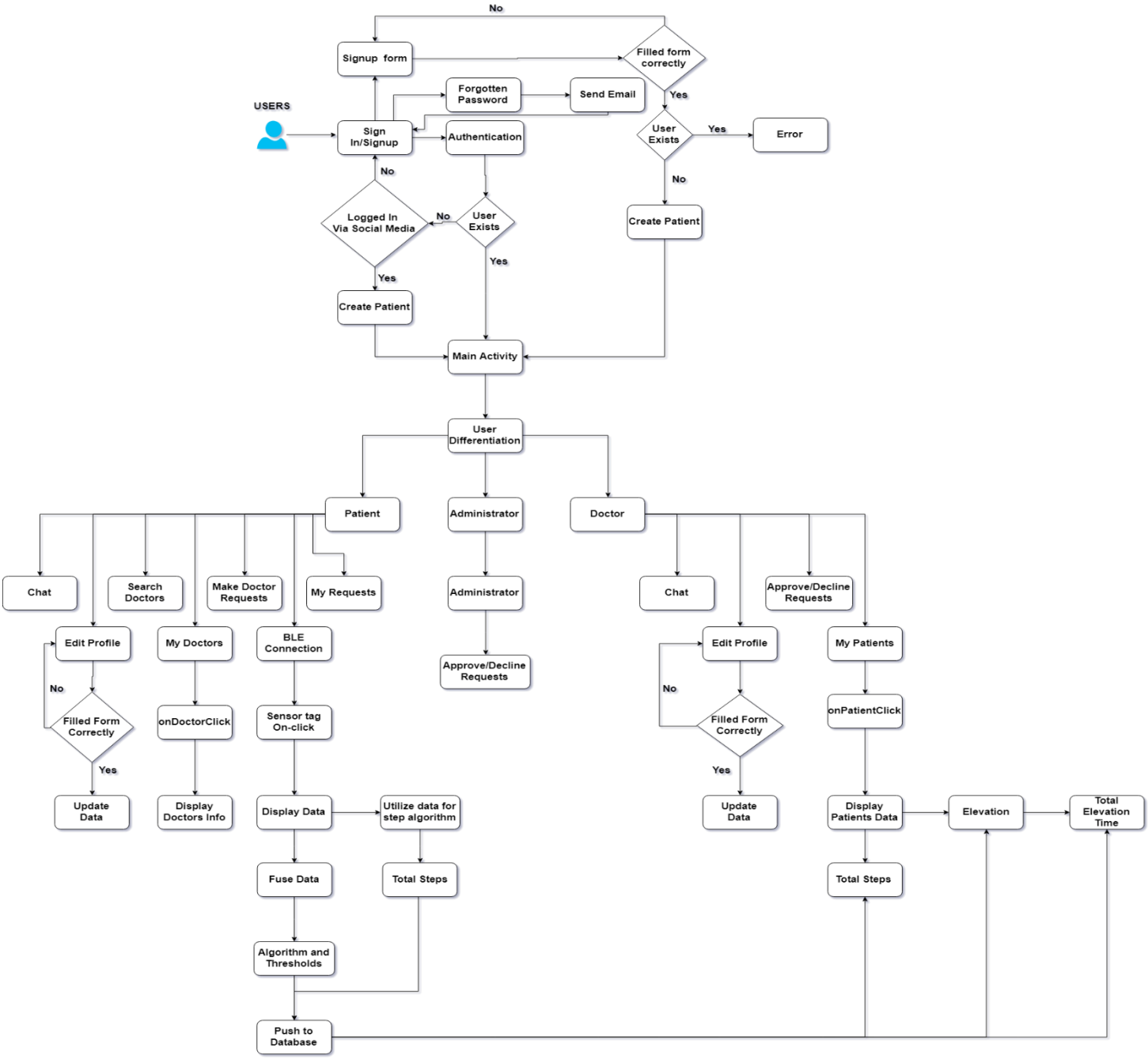

Figure 4-3 - UML diagram

The diagram starts off with authenticating the user and differentiating between the types. After the types have been differentiated, the user is then able to carry out their associated functions, based on their status and type. When the status is in an approved state this will enable the doctor to see the patient's data. The patient although can see all the doctor's information and from then on based on the doctor of his choice, can request to come under his supervision, which will then enable the doctor to see that patients' data.

4.2.1 Doctors

The doctor involves several functions as clearly indicated in the below flow diagram:

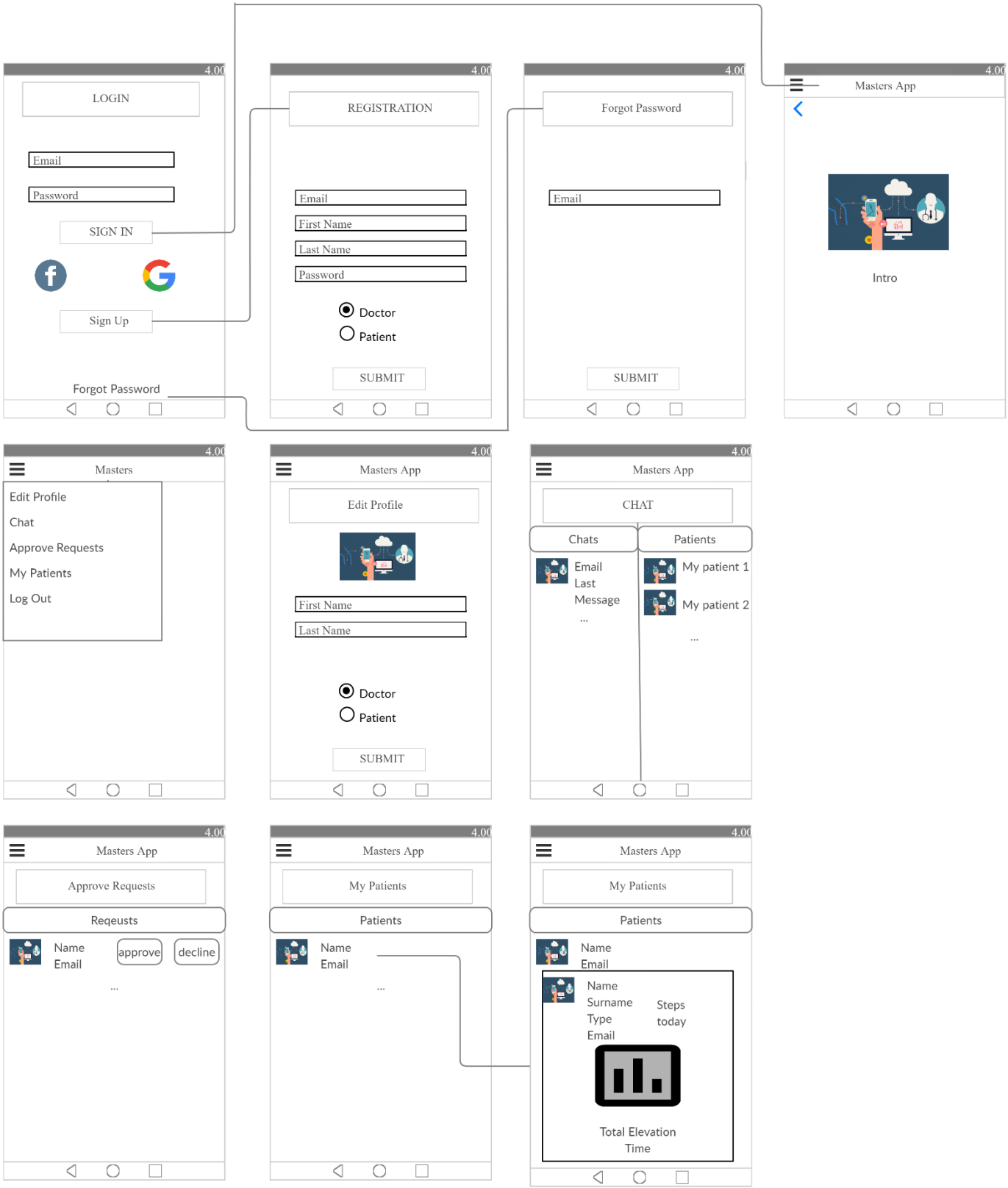

Figure 4-4 - Doctor Flow Diagram

The Figures above illustrate in a timely manner how the doctor can traverse the application in order to identify the patients' data. In addition to those diagrams, this diagram perfectly aligns with how the UML diagram represents the doctor type interaction procedure. The doctor has functionalities that enable them to see only the data they want to see by approving requests from patients, the patients supervised by the current doctor are then displayed in a separate Fragment that is populated based on their patient base.

4.2.2 Patients

Patients have multiple different functionalities that enable the user to collect data from his sensors and submit the data to the database in order to inform the doctor that they are complying with the doctors' instructions. The diagram below indicates the timely mannered flow of the patient in order to receive the data and submit them to the database for further inspection by the doctor.

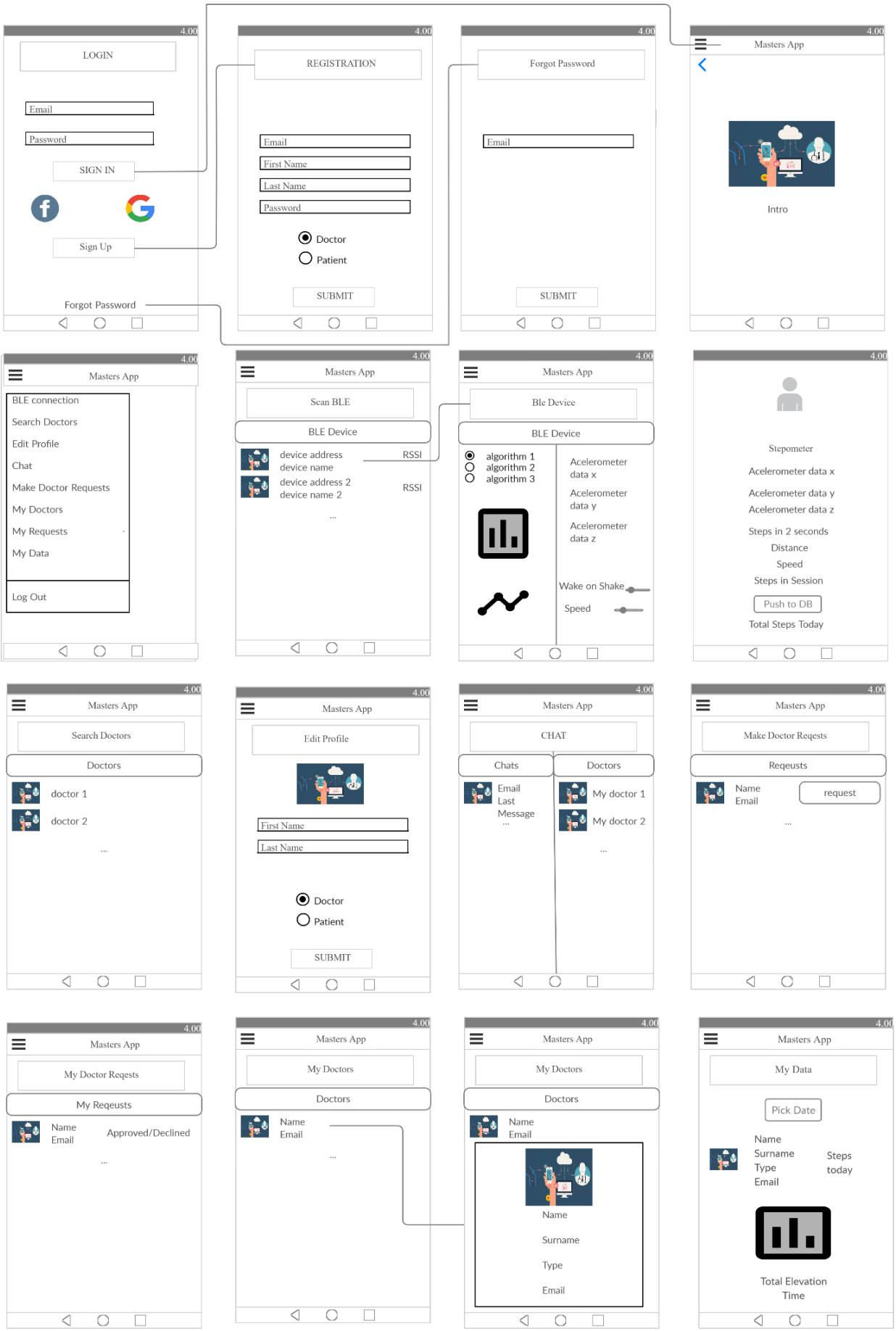

Figure 4-5 – Patient Flow Diagram

The diagram above indicates how the user normally authenticates and is then directed to the Main Activity where the user has the choice to make requests to the doctors and search for any available doctors that have not been requested yet, in addition to the previous functions a Fragment is implemented to display the status of the users' requests. The patient can edit their profile and can chat with his doctors. The most important function that the user must do is the connection between the sensor device, which will then feed the appropriate data to the application. The data will then be passed through a fusion algorithm as well as an algorithm that utilises thresholds. The application will then send data to the database in order to inform that the status of the leg has changed. In case the patient is in a rehabilitation state, the function alters to whether the user has made a step and decided to submit his workout for post-surgery applications.

4.2.3 Administrator

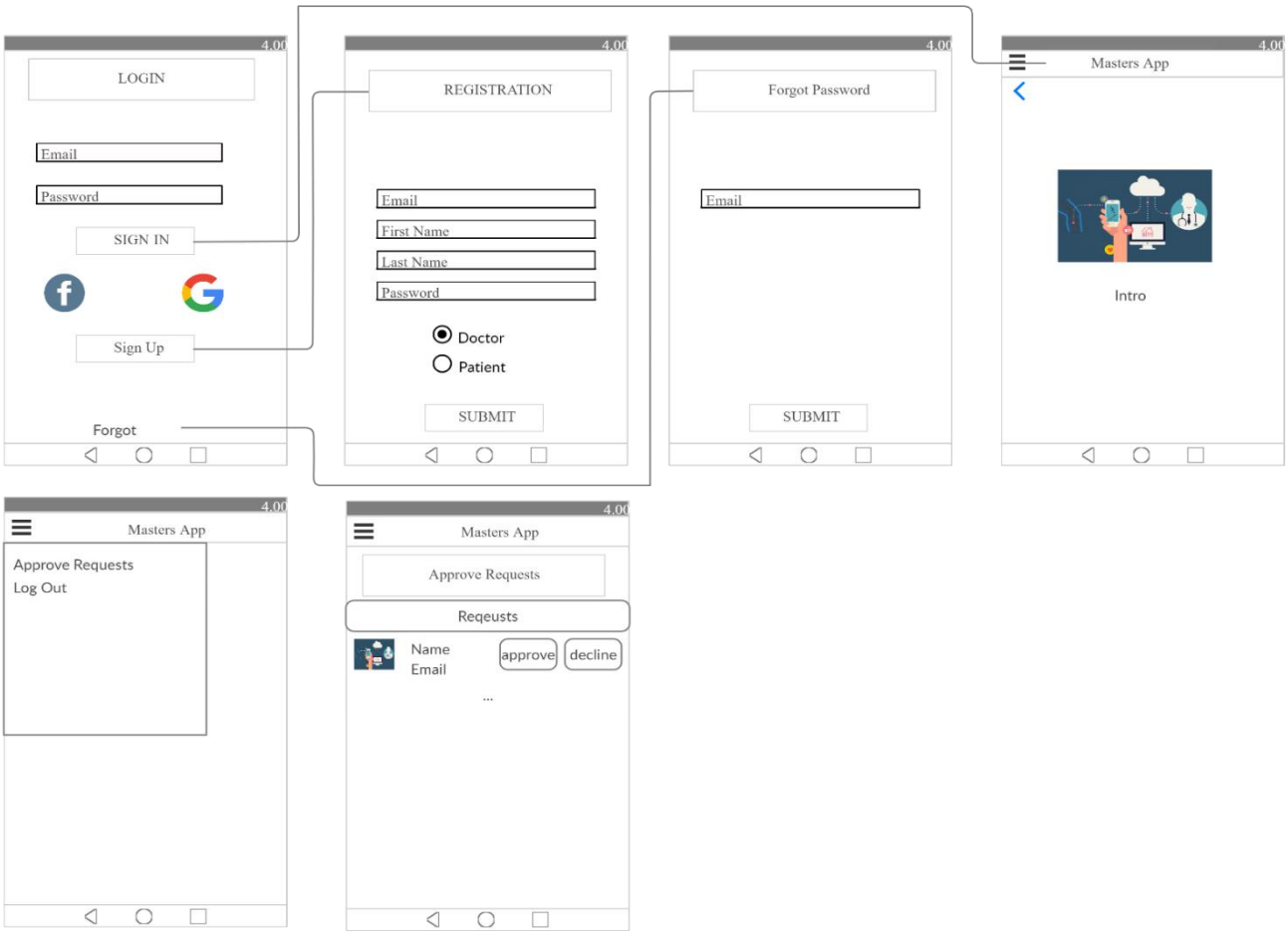

Figure 4-6 Administrator Structure

The Administrators main role of this prototype is to approve doctors. By following the exact same login procedure as the other users will enable them to log in and are then presented with the above Navigation bar which provides them with the ability to approve or decline doctor requests.

4.3 Back End

The backend is comprised of Googles' proprietary database that is Firebase. It is most of all free to use and comes with support within the IDE that is Android Studio. It enables the database to operate with the application in order to keep things up to date and keeping users informed of the data that have been published in Real-Time. Cloud Firestore is a NoSQL database comprised of collections, documents and fields as clearly indicated by the diagram below [23].

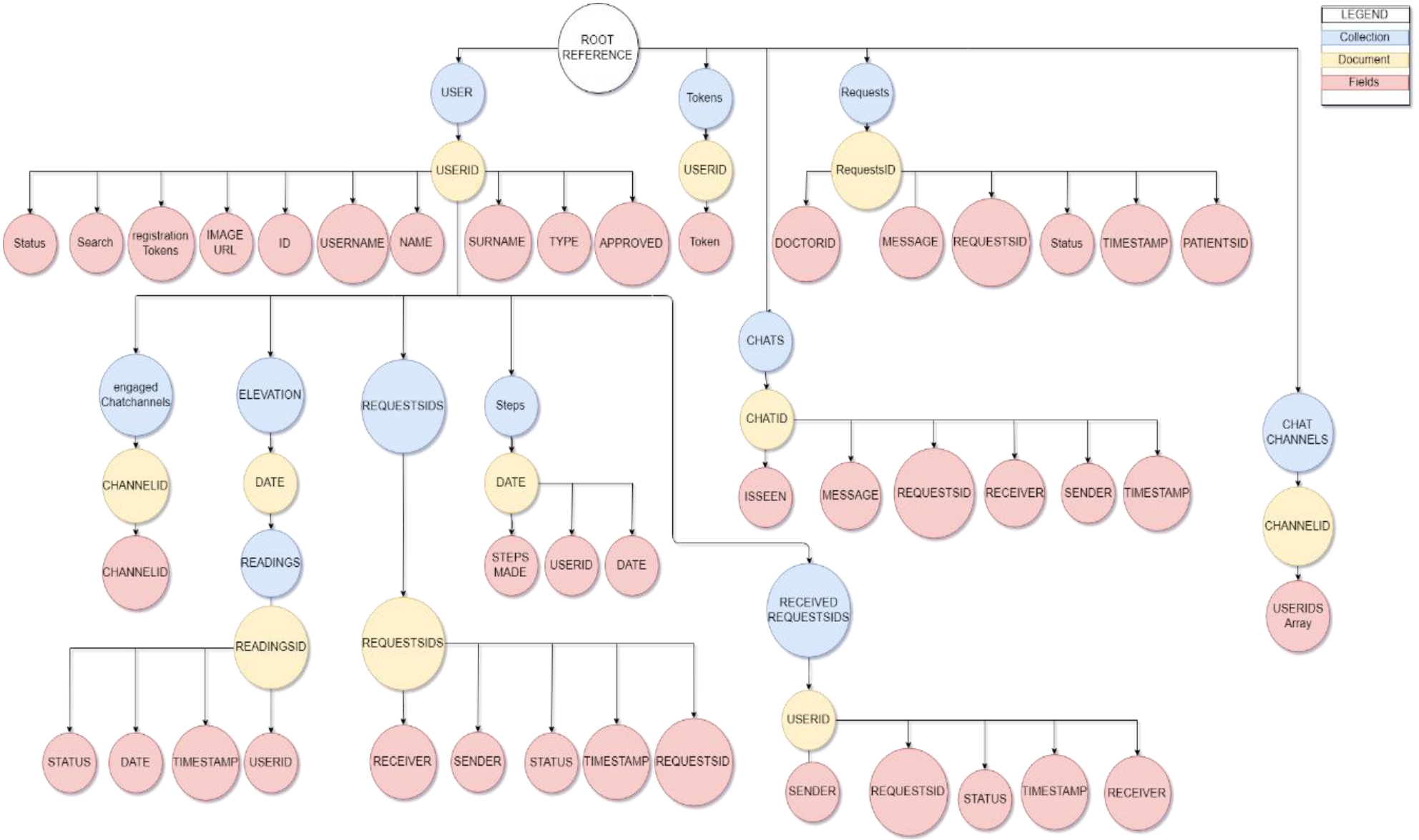

Figure 4-7 - Database Structure

The free cloud Firestore service has a constraint of limited read and writes in the free utilisation of the software [24]. Therefore, the application needed to be adapted around this limitation in order to be able to operate and enable us to troubleshoot in case of multiple reads and writes. This database will enable the structural hierarchy to remain intact, offering multiple functionalities through the Googles Function Services that enable the read and write of the data to be seamless. In the above diagram, there are currently five main collections connected to the root reference, indicated by the blue colour being USER, TOKENS, REQUESTS, CHATS and CHATCHANNELS. These collections are then subdivided into multiple documents and sub-collections that make this project feasible.

CHATS and CHATCHANNELS and TOKENS are utilised for the communication sector where tokens are utilised in order to send notifications to the user that there is a message from their doctor or whether the doctor has a message from their patients. REQUESTS are utilised between the doctor and their patient which enables the relationship to take place between those two individuals. USER stores all the users along with their very own sub-collections. Each user has sub-collections based on their type. Patients have sub-collections such as their REQUESTIDS, STEPS, ENGAGEDCHATCHANNELS and ELEVATION with multiple fields describing each one of those collections. Whereas a Doctor will only have two sub-collections RECEIVEDREQUESTIDS and ENGAGEDCHATCHANNELS, due to the lack of functionality required.

4.4 Summary

In this chapter, we have identified what is required by the application to operate. The frontend of the application has been identified for each individual and the foundations of the database illustrated. The hardware that is required has been specified and a UML diagram has been used to display the functionalities of the application, as well as the flow of the Activities available to each type of user, which is aiding the understanding of the application. This chapter enables further developers to re-enact this experiment and to utilise several of the standards in order to get a positive outcome. It has outlined what is required for the project to be feasible and the foundations have been set in order to aid in implementation.

Chapter 5 Implementation

This chapter explains how this application is feasible by following the guidelines set by the design section in order to produce a sufficient and positive outcome. Different problems were identified and then resolved by utilisation of functions and algorithms. The application will be introduced into separate sections which will include explanations for the Frontend, its Functionalities, the Database and the Algorithms utilised in order to make this project feasible.

The project has been implemented on Android Studio as an IDE and will utilise the proprietary Google Firebase Cloudstore for the backend as the database. The languages utilised for this implementation range from XML to Java and Kotlin along with minimal JavaScript for functions utilised by the Kotlin chat for the notifications part.

5.1 Database Implementation/Authentication

It was necessary for this application to have a centralised location which would keep all the information that would be required by the users.
In order to alleviate this problem a database has been implemented through the IDE which has premade refactoring implementations, this can be done by utilising the documentation presented by the "Assistant", this assistant has links with the appropriate guidelines required for a secure yet robust implementation of Authentication and Database communication.
By following the guidelines presented by the documentation [25] it will guide the developer into implementing a login function and a secure connection to the database by utilising OAuth2 and automatically enabling communication with the database, following these instructions will then guide you into implementing Authentication as shown below by the following figure. For further information refer to section 9.2 of the Appendix.

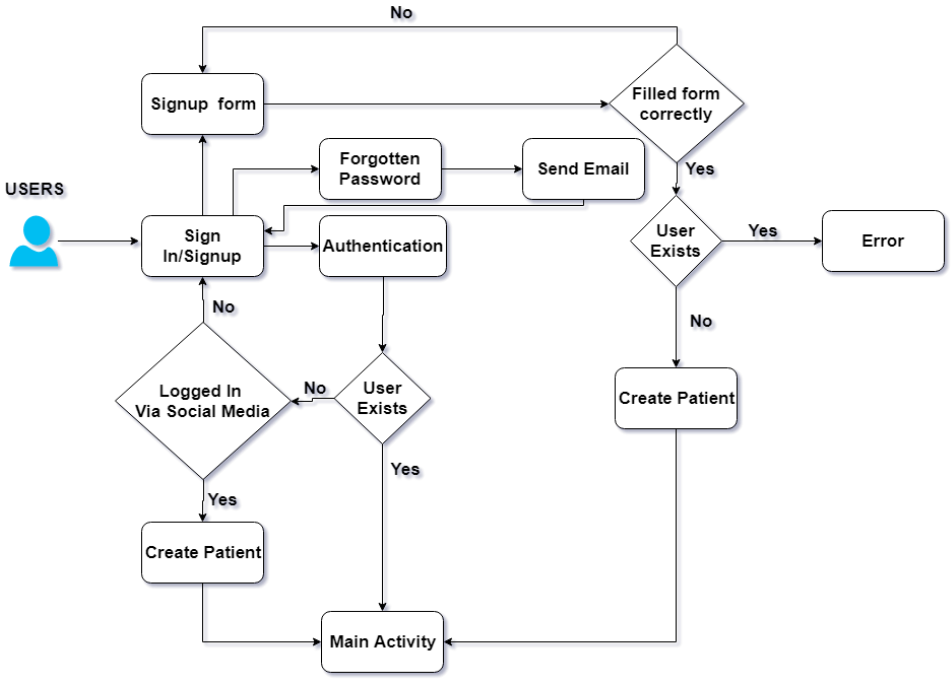

Figure 5-1 Authentication

The above UML diagram indicates the process required to be done by a user in order to log in. The user is required to input their information to either signup or sign in or in case of forgotten password to alter it. Upon submission, will directly read from the database in order to check if the user exists and authenticate accordingly. This will also result in the data provided to create a shared preferences session, which will be utilised during periods that the user is offline. This is used in case the application loses its internet connection to maintain some functionality of the application even offline. In addition to these functions, the ability to utilise social media as means of authenticating the user has also been implemented, all users signed in with social media will be considered as patients and can request to become doctors in their profile Fragment. At that time, they will then be able to get approved by the administrator, which will enable them to become authenticated and approved doctors who will then be searchable by the patients.

5.2 HideItems

As all the data was centralised a method was then required in order to differentiate between the users based on their type and whether they are approved or not.

To solve this there was a need for a function that differentiates users based on those criteria. HideItems is a function utilised for the differentiation of the type of user and their status, each user as soon as they are authenticated will be greeted with his own navigation panel which has all the required functionalities.

The function has two variation modes:

5.2.1 Online Mode

Online mode is when the device is connected to the Internet which will enable it to call the function based on the up to date information from the database.

5.2.2 Offline Mode

Offline mode is when the device is not connected to the Internet which will enable it to call the function based on the information from the shared preferences session.

Both functions operate similarly and based on the values it receives, it will display the appropriate navigation. Shown in the figure below is the UML layout for hiding items in the navigation Drawer.

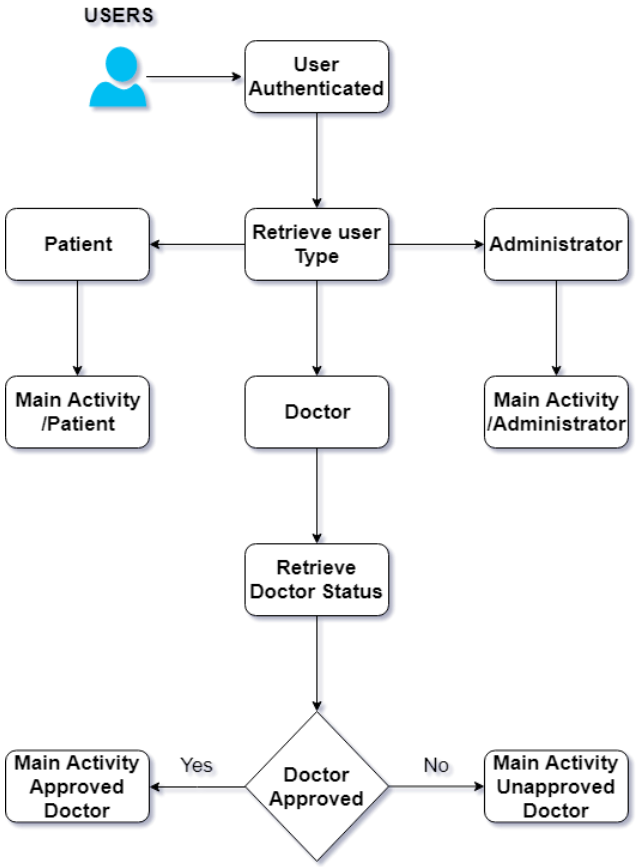

Figure 5-2 Hide Items

Based on this function users are demonstrated differential navigation panels with different functionalities matching exactly what has been identified by the Design Diagrams. This function is repeated when the user decides that he is not willing to change his type into a doctor or as a doctor would like to see what the functionalities of the patient are, they can easily switch back and forth between those two functions with no loss of their status. This needs to be done through the Edit profile Fragment which can modify the data accordingly in order to enable the switch from Patient to Doctor and vice versa. More information on the source code can be found in section 9.2 of the Appendix.

5.3 Foundation Functionalities

The foundation functionalities which had been implemented can be found in the Appendix. Below is a table indicating all the implemented foundation Functionalities which make this project feasible. These functions enable the application to be more secure by establishing access control on data and accountability of changes by seeing who has access to which data. Medical data are of utmost importance and they are required to be secure in order to avoid tampering. These functions will also increase its expandability since these foundations have been set to enable this project to progress and focus on other types of medical emergencies or other readings. It has an implementation that will enable the cooperation of this system in a live environment with real human beings and makes it more readily available for commercial usage.

Refer to the table indicating the sections within the Appendix to see the corresponding functions.

Table 5-1 Foundation Functions

ID	User	Functionality	Status	Section
1	Administrator	Approve Doctors	Implemented	9.1.1.1
2	Doctor	Edit Profile, switch to Patient and vice versa	Implemented	9.1.2.1
3	Doctor	Chat/Communication, notifications, Patient chat lists	Implemented	9.1.2.2
2	Doctor	Approve Requests from patients	Implemented	9.1.2.3
4	Doctor	Show all My Patients and their Data appropriately	Implemented	9.1.2.4
8	Patient	BLE Connection, Retrieve Data from Sensor	Implemented	9.1.3.1
9	Patient	Search all Doctors	Implemented	9.1.3.2
6	Patient	Edit Profile	Implemented	9.1.3.3
7	Patient	Chat/Communication, notifications, Doctor chat lists	Implemented	9.1.3.4
10	Patient	Make Requests	Implemented	9.1.3.5
11	Patient	Show my Requests	Implemented	9.1.3.6

12	Patient	Show my Doctors	Implemented	9.1.3.7
13	Patient	See my data	Implemented	9.1.3.8
14	Patient	Sensor Tag Connect	Implemented	9.2

5.4 Algorithms

For this application to operate, and to seamlessly retrieve the data of the patient in order to detect Steps and Elevation there is a need for several algorithms to be implemented. Since no libraries were readily available an implementation of these algorithms was required. The algorithms will be divided into sections in the order of Fusion, Elevation and Steps which will then further be subdivided into appropriate sections according to their functionality.

5.4.1 Fusion

As soon as the data have been received and plotted, the outcome of the graph indicated that the sensors are susceptible to noise when implemented separately. In order to retrieve more accurate data and get exact orientations of the device and eliminate drift over time, there is a need for an algorithm to be implemented in order to fuse the data appropriately and to pass it through filters so as to stabilise the data and output angles/in degrees with minimal error. There are currently 4 live implementations on this application. The Google Fusion based on Google functions and a High& Low pass filter. A Kalman Fusion algorithm, Mahony AHRS and Madgwicks' Fusion AHRS algorithm. Below are the Google functions with Matrixes whereas the Madgwicks, Mahony and Kalman Algorithms are shown in the Appendix in section 9.4. The project has all four algorithms implemented, although the reason Google Fusion algorithms are utilised is due to the functions available being refined for mobile devices. This enables reduced battery consumption and only utilises a fraction of the processing required. The key feature that is available through the Google Functions is multiple implementations that enable the mitigation and alleviation of drift, whereas in order to receive accurate data through The Madgwicks & Mahony algorithm there was a requirement for a calibration algorithm to be developed. These functions also enable the application to properly utilise these outputs to the correct detection of this project. Not accumulating errors over time, is key in retrieving accurate data that are not susceptible to drift which could seriously hinder the results of our application. Therefore, enabling this application to withstand prolonged operative hours.

All the Fragments in this implementation require the broadcast receiver of the application which is shown below:

```
01. private final BroadcastReceiver motionUpdateReceiver = new BroadcastReceiver() {
02.         @Override
03.         public void onReceive(Context context, Intent intent) {
04.
05.             final String action = intent.getAction();
06.
07.             if (IntentNames.ACTION_MOV_CHANGE.equals(action)) {
08.
09.                 //Log.i(TAG, "**************** MOTION sensed ****************");
10.                 byte[] value = intent.getByteArrayExtra(IntentNames.EXTRAS_MOV_DATA);
11.                 Point3D v;
12.
13.
14.                 v = SensorConversion.MOVEMENT_ACC.convert(value);
15.                 double[] accelvalues = new double[3];
16.                 accelvalues[0] = v.x;
17.                 accelvalues[1] = v.y;
18.                 accelvalues[2] = v.z;
19.                 System.arraycopy(accelvalues, 0, accel, 0, 3);
20.
21.
22.                 v = SensorConversion.MOVEMENT_GYRO.convert(value);
23.                 double[] gyrovalues = new double[3];
24.                 gyrovalues[0] = v.x;
25.                 gyrovalues[1] = v.y;
26.                 gyrovalues[2] = v.z;
27.                 System.arraycopy(gyrovalues, 0, gyro, 0, 3);
28.
29.
30.                 v = SensorConversion.MOVEMENT_MAG.convert(value);
31.                 double[] magvalues = new double[3];
32.                 magvalues[0] = v.x;
33.                 magvalues[1] = v.y;
34.                 magvalues[2] = v.z;
35.                 System.arraycopy(magvalues, 0, magnet, 0, 3);
36.
37.             }
38.         }
39.     };
```

Figure 5-3 Broadcast Receive (Fragments)

By retrieving the raw values of the sensors through a broadcast enables the simplification of retrieving the values in any Fragment housed in the connection activity enabling us to implement multiple functions and algorithms. In the following section, the utilised fusion algorithm is illustrated.

5.4.1.1 Google Fusion with Orientation Matrixes

For this implementation to operate, multiple google functions have been utilised in order to retrieve matrixes which are then converted into angles and posted into graphs that will enable us to utilise the data appropriately. As soon as the broadcast data have been received there are multiple functions called each time. The following diagram illustrates the algorithm flow.

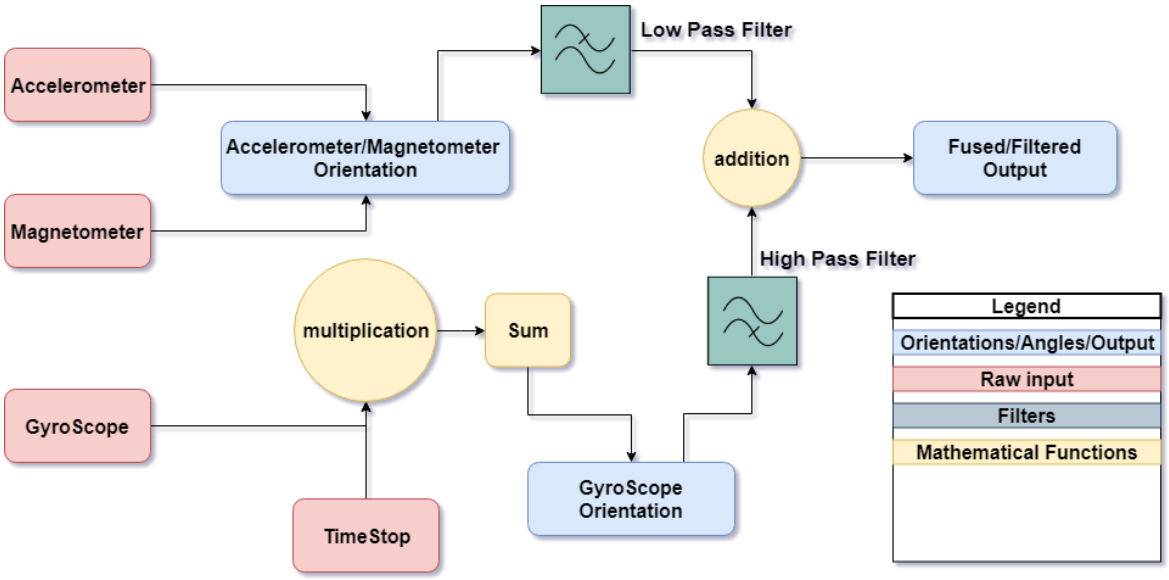

Figure 5-4 Algorithm Layout

In order to enable the fusion algorithm to operate the other two orientations need to be calculated and accurately received, as the fusion algorithms take those two outputs as input. The two inputs of the algorithm being Accelerometer/Magnetometer Orientation and the Gyroscope Orientation are further explained in section 9.4.4 of the Appendix. The fusion algorithm is now explained in the following section.

5.4.1.1.1 Fused Orientation

Fused orientation utilises the two outputs of the previous algorithms and fuses and filters them.

```
class calculateFusedOrientationTask extends TimerTask {
    public void run() {
        float oneMinusCoeff = 1.0f - FILTER_COEFFICIENT;

        // yaw
        if (gyroOrientation[0] < -0.5 * Math.PI && accMagOrientation[0] > 0.0) {
            fusedOrientation[0] = (float) (FILTER_COEFFICIENT * (gyroOrientation[0] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[0]);
            fusedOrientation[0] -= (fusedOrientation[0] > Math.PI) ? 2.0 * Math.PI : 0;
        }
        else if (accMagOrientation[0] < -0.5 * Math.PI && gyroOrientation[0] > 0.0) {
            fusedOrientation[0] = (float) (FILTER_COEFFICIENT * gyroOrientation[0] + oneMinusCoeff * (accMagOrientation[0] + 2.0 * Math.PI));
            fusedOrientation[0] -= (fusedOrientation[0] > Math.PI)? 2.0 * Math.PI : 0;
        }
        else {
            fusedOrientation[0] = FILTER_COEFFICIENT * gyroOrientation[0] + oneMinusCoeff * accMagOrientation[0];
        }

        // pitch
        if (gyroOrientation[1] < -0.5 * Math.PI && accMagOrientation[1] > 0.0) {
            fusedOrientation[1] = (float) (FILTER_COEFFICIENT * (gyroOrientation[1] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[1]);
            fusedOrientation[1] -= (fusedOrientation[1] > Math.PI) ? 2.0 * Math.PI : 0;
        }
        else if (accMagOrientation[1] < -0.5 * Math.PI && gyroOrientation[1] > 0.0) {
            fusedOrientation[1] = (float) (FILTER_COEFFICIENT * gyroOrientation[1] + oneMinusCoeff * (accMagOrientation[1] + 2.0 * Math.PI));
            fusedOrientation[1] -= (fusedOrientation[1] > Math.PI)? 2.0 * Math.PI : 0;
        }
        else {
            fusedOrientation[1] = FILTER_COEFFICIENT * gyroOrientation[1] + oneMinusCoeff * accMagOrientation[1];
        }

        // roll
        if (gyroOrientation[2] < -0.5 * Math.PI && accMagOrientation[2] > 0.0) {
            fusedOrientation[2] = (float) (FILTER_COEFFICIENT * (gyroOrientation[2] + 2.0 * Math.PI) + oneMinusCoeff * accMagOrientation[2]);
            fusedOrientation[2] -= (fusedOrientation[2] > Math.PI) ? 2.0 * Math.PI : 0;
        }
        else if (accMagOrientation[2] < -0.5 * Math.PI && gyroOrientation[2] > 0.0) {
            fusedOrientation[2] = (float) (FILTER_COEFFICIENT * gyroOrientation[2] + oneMinusCoeff * (accMagOrientation[2] + 2.0 * Math.PI));
            fusedOrientation[2] -= (fusedOrientation[2] > Math.PI)? 2.0 * Math.PI : 0;
        }
        else {
            fusedOrientation[2] = FILTER_COEFFICIENT * gyroOrientation[2] + oneMinusCoeff * accMagOrientation[2];
        }

        // overwrite gyro matrix and orientation with fused orientation
        // to comensate gyro drift
        gyroMatrix = getRotationMatrixFromOrientation(fusedOrientation);
        System.arraycopy(fusedOrientation, 0, gyroOrientation, 0, 3);


        // update sensor output in GUI
        mHandler.post(updateOreintationDisplayTask);
    }
}
```

Figure 5-5 Complete Fusion with filters

Filtering is done on a thread of its own to get the most out of implementing it.

The low-pass filter is implemented on Accelerometer/Magnetometer Orientation by utilising the average of the angles over time.

Whereas the High pass filter is implemented on the gyroscope and is done by replacing the filtered high-Frequency component from Accelerometer/Magnetometer Orientation with the corresponding gyroscope orientation values:

fusedOrientation =

(1 - FilterCoefficient) * latestgyrovalue **for the high-frequency component**

+ *FilterCoefficient * latestAcc/Magvalue* **for the low-frequency component**

Figure 5-6 Filter pseudocode

Below are the results of the fused and filtered data, which are indicated by the smooth outputted lines and stationary output in Yaw (Green) Pitch (Cyan) and Roll (Red).

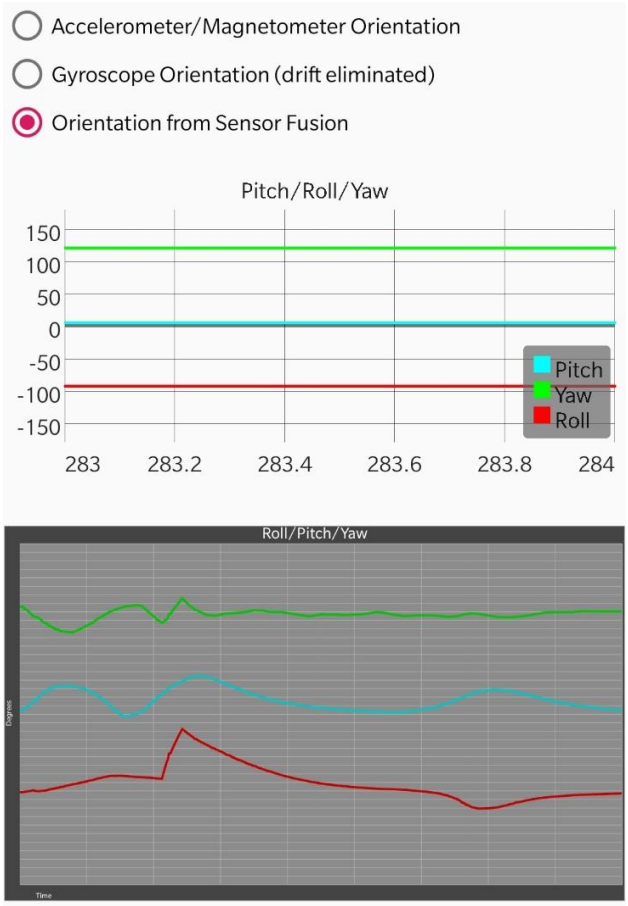

Figure 5-7 Fused Orientation

This fragment is what is illustrated to the patient meanwhile his leg is either being raised or lowered the variation of the angles are the forces being applied on each of the angles of the sensor outputting into orientation. In the following section, the Elevation algorithm developed is displayed.

5.4.2 Elevation

In matters of Elevation, there was a need for an algorithm to be created as nothing was readily available for detecting the signatures of up and down movements. The implementation of elevation is illustrated below which shows the utilisation of multiple functions and has two stabilising algorithms. One algorithm utilises the means of the fused data and the other the acceleration data. Timers are also used to keep writing to the database minimal.

The algorithm has been implemented in the Fragment where the Google Functions have been utilised. This is due to the output having stable readings and because it's utilising android algorithms which enable the minimisation of battery usage and straining of the mobile device. It also enabled the differentiation between the fusion algorithms and all the variations of orientations retrieved from 3 sensor combinations to be perceived. In addition, filtering has been utilised on the outputted orientations and once fused would result in an increase in the accuracy of readings dramatically, which in turn would make the detection algorithm more robust and accurate.

5.4.2.1 Start checking up/Start checking Down after 1.5 seconds

The functions utilised within the broadcast in order to check for up and down are illustrated below.

```
01.  final String action = intent.getAction();
02.          lastreceived++;
03.          if (lastreceived == 30) {
04.              startingposition();
05.
06.          }
07.          if (lastreceived > 30){
08.              Log.d(TAG, "last received going to check for up");
09.              getconstantvaluesmean();
10.
11.
12.                  if (!isup&& stationary&&isStationary) {
13.
14.                      mHandler.postDelayed(Checkforup, 1500);
15.                  } else if (isup&& stationary&&isStationary ) {
16.
17.                      mHandler.postDelayed(CheckforDown, 1500);
18.                  }
19.
20.          }
```

Figure 5-8 Check for Up and Down

This function starts operating after the first 30 values have been received, this is due to the conversion time required along with the filtration to start in order to retrieve the accurate data.

5.4.2.2 Starting position of Sensor

```
01.  private void startingposition() {
02.
03.
04.      int initialvalues = 30;
05.   if ((fusedOrientation[0] * 180 / Math.PI) != 0) {
06.
07.
08.          for (int y = 0; y < initialvalues; y++) {
09.
10.              f3startpitchvalues.add((float) (fusedOrientation[1] * 180 / Math.PI));
11.              f3startrollvalues.add((float) (fusedOrientation[2] * 180 / Math.PI));
12.              f3startyawvalues.add((float) (fusedOrientation[0] * 180 / Math.PI));
13.
14.              f3sumroll = (f3startrollvalues.get(y) + f3sumroll);
15.              f3sumyaw = (f3startyawvalues.get(y) + f3sumyaw);
16.              f3sumpitch = (f3startpitchvalues.get(y) + f3sumpitch);
17.          }
18.
19.
20.          f3startingroll = f3sumroll / initialvalues;
21.          f3startingyaw = f3sumyaw / initialvalues;
22.          f3startingpitch = f3sumpitch / initialvalues;
23.          Log.d(TAG, "the f3roll: "+f3startingroll+"the f3yaw: "+f3startingyaw+"the f3pitch: "+f3startingpitch);
24.
25.
26.
27.
28.      }
29.
30.  }
```

Figure 5-9 Starting position function

During the first 30 values, the starting position is identified by utilising standard deviation of the first 30 values and finding the mean which can then illustrate an accurate starting position.

5.4.2.3 Mean Values

After the values had reached 30 values, an algorithm was developed that takes the values that are constantly being streamed and calculating their mean every 8 values, shown below.

```
01.          private void getconstantvaluesmean() {
02.                  int howmanytogetformean = 8;
03.                   f3sumpitch = 0;
04.                   f3sumroll = 0;
05.                   f3sumyaw = 0;
06.
07.                  f3pitchvalues.add((float) (fusedOrientation[1] * 180 / Math.PI));
08.                  f3rollvalues.add((float) (fusedOrientation[2] * 180 / Math.PI));
09.                  f3yawvalues.add((float) (fusedOrientation[0] * 180 / Math.PI));
10.
11.                  f3currentpitch = (float) (fusedOrientation[1] * 180 / Math.PI);
12.                  f3currentroll = (float) (fusedOrientation[2] * 180 / Math.PI);
13.                  f3currentyaw = (float) (fusedOrientation[0] * 180 / Math.PI);
14.
15.
16.
17.          int f3listsize = f3pitchvalues.size();
18.
19.                  if (f3listsize > howmanytogetformean) {
20.                    int f3startfrom = f3listsize - howmanytogetformean;
21.
22.                      for (int y = f3startfrom; y < f3listsize; y++) {
23.
24.                          f3sumroll = (f3rollvalues.get(y) + f3sumroll);
25.                          f3sumyaw = (f3yawvalues.get(y) + f3sumyaw);
26.                          f3sumpitch = (f3pitchvalues.get(y) + f3sumpitch);
27.
28.
29.                      }
30.
31.
32.                      f3meanroll = f3sumroll / howmanytogetformean;
33.                      f3meanyaw = f3sumyaw / howmanytogetformean;
34.                      f3meanpitch = f3sumpitch / howmanytogetformean;
35.                     // Log.d(TAG, "the f3meanroll: " + f3meanroll + "the f3meanyaw: " + f3meanyaw + "the f3pitch: " + f3meanpitch);
36.
37.
38.                  } else {
39.                      for (int y = 0; y < f3listsize; y++) {
40.                          f3sumroll = (f3rollvalues.get(y) + f3sumroll);
41.                          f3sumyaw = (f3yawvalues.get(y) + f3sumyaw);
42.                          f3sumpitch = (f3pitchvalues.get(y) + f3sumpitch);
43.
44.
45.                      }
46.                      f3meanroll = f3sumroll / f3listsize;
47.                      f3meanyaw = f3sumyaw / f3listsize;
48.                      f3meanpitch = f3sumpitch / f3listsize;
49.                     // Log.d(TAG, "the f3meanroll: " + f3meanroll + "the f3meanyaw: " + f3meanyaw + "the f3meanpitch: " + f3meanpitch);
50.      }
51.  }
```

Figure 5-10 - Mean values algorithm

The mean value algorithm is to detect how much of a difference there is from the previous reading and utilise it in the elevation detection algorithm in order to reduce false-positive values. These values could then potentially make the algorithm trigger more accurately if there is substantial change being detected.

5.4.2.4 Stabilising algorithms

In order to alleviate false positive on the algorithm, there is an implementation of two stabilisation detection functions that will alleviate the algorithm triggering if there is no movement, further eliminating false-positive errors from taking place.

The first stabilising algorithm is implemented by using the raw data of the accelerometer sensor in order to detect whether there is any acceleration in any direction.

The second stabilising algorithm implemented listens for changes of the fused data and whether the change since the previous values is minimal which would indicate the leg is in a stationary position.

Below is the algorithm utilising the raw data of the accelerometer.

```
01. private void checkforstationary(float[] accelerometer){
02.
03.     double x = accelerometer[0];
04.     double y = accelerometer[1];
05.     double z = accelerometer[2];
06.     mAccelLast = mAccelCurrent;
07.     mAccelCurrent = Math.sqrt(x * x + y * y + z * z);
08.     double delta = mAccelCurrent - mAccelLast;
09.     mAccel = mAccel * 0.9f + delta;
10.
11.     int SAMPLE_SIZE = 1;
12.     if (hitCount <= SAMPLE_SIZE) {
13.         hitCount++;
14.         hitSum += Math.abs(mAccel);
15.     } else {
16.         hitResult = hitSum / SAMPLE_SIZE;
17.
18.         Log.d(TAG, String.valueOf(hitResult));
19.
20.         double THRESHOLD = 0.2;
21.         isStationary = !(hitResult > THRESHOLD);
22.
23.         hitCount = 0;
24.         hitSum = 0;
25.         hitResult = 0;
26.     }
27. }
```

Figure 5-11 Acceleration stationary algorithm

With the results shown below

```
com.example.myapplication D/the displayreading2 frag:: stationary: true
```

Figure 5-12 Stationary 1

There are currently 3 consecutive algorithms running calculating their own different orientation at the same time. The algorithm has been adapted to this function and detects which algorithm the user is using and only the one chosen by the user will be utilised.

Below are the stationary functions of the fused data and their orientation.

```
private void checkifstationary(){
    boolean pitchstationary = false;
    boolean rollstationary = false;
    boolean yawstationary = false;

    Log.d(TAG, "the what is now pitch "+f1currentpitch+"what is mean pitch"+ f1meanpitch);
    Log.d(TAG, "the what is now roll "+f1currentroll+"what is mean roll"+ f1meanroll);
    Log.d(TAG, "the what is now yaw "+f1currentyaw+"what is mean yaw"+ f1meanyaw);

    //if rounding = 0 then put valious of two decimals up and two decimals down
     if (((((Math.round(f1currentpitch) == 0)|| (Math.round(f1currentpitch) == 1))
|| ((Math.round(f1meanpitch) == 0)|| (Math.round(f1meanpitch) == 1)))
     || (((Math.round(f2currentpitch) == 0)|| (Math.round(f2currentpitch) == 1))
|| ((Math.round(f2meanpitch) == 0)|| (Math.round(f2meanpitch) == 1)))
     || (((Math.round(f3currentpitch) == 0)|| (Math.round(f3currentpitch) == 1))
|| ((Math.round(f3meanpitch) == 0)|| (Math.round(f3meanpitch) == 1))))
              //negative values
      || ((( Math.round(f1currentpitch) == -1) || (Math.round(f1meanpitch) == -1))
      || ( ( Math.round(f2currentpitch) == -1) || (Math.round(f2meanpitch) == -1))
      || ( ( Math.round(f3currentpitch) == -1) || (Math.round(f3meanpitch) == -1))))

    {
         if ((((f1currentpitch < f1meanpitch + 3) && (f1currentpitch > f1meanpitch - 3))
|| ((f1currentpitch > f1meanpitch + 3) && (f1currentpitch < f1meanpitch - 3)))
                || (((f2currentpitch < f2meanpitch + 3) && (f2currentpitch > f2meanpitch - 3))
|| ((f2currentpitch > f2meanpitch + 3) && (f2currentpitch < f2meanpitch - 3)))
                || (((f3currentpitch < f3meanpitch + 3) && (f3currentpitch > f3meanpitch - 3))
|| ((f3currentpitch > f3meanpitch + 3) && (f3currentpitch < f3meanpitch - 3))))
                {
             Log.d(TAG, "the pitch is stationary now ");
             pitchstationary=true;
         }else
         {
             Log.d(TAG, "the pitch is not stationary yet moves to second class ");

         }

     } else{                                                                    //negative values
     if ((((f1currentpitch < f1meanpitch * 1.03) && (f1currentpitch > f1meanpitch * 0.97))
|| ((f1currentpitch > f1meanpitch * 1.03) && (f1currentpitch < f1meanpitch * 0.97)))
             || (((f2currentpitch < f2meanpitch * 1.03) && (f2currentpitch > f2meanpitch * 0.97))
|| ((f2currentpitch > f2meanpitch * 1.03) && (f2currentpitch < f2meanpitch * 0.97)))
             ||(((f3currentpitch < f3meanpitch * 1.03) && (f3currentpitch > f3meanpitch * 0.97))
|| ((f3currentpitch > f3meanpitch * 1.03) && (f3currentpitch < f3meanpitch * 0.97))))

     {
         Log.d(TAG, "the pitch is stationary now ");
         pitchstationary=true;
     }else
     {
         Log.d(TAG, "the pitch is not stationary on second class fail");

     }



     }
```

Figure 5-13 Pitch fusion stationary algorithm

The algorithm above has been implemented for Pitch this was developed with detecting changes in mind. Any value that is capable to show a significant difference of 3% or -3% can trigger the algorithm which indicates that the patient is moving their leg. On the other hand, anything not passing this threshold is not worth identifying. For values that are under 10 degrees, the values of 3% are minimal. Therefore, the algorithm was adapted to withstand such rates and detect whether the algorithm change was 3 degrees up or down. For the implementation of Roll and Yaw please refer to section 9.4.5.1 in the Appendix.

The results are shown below.

```
/com.example.myapplication D/the displayreading2 frag:: the pitch is stationary now
/com.example.myapplication D/the displayreading2 frag:: the roll is stationary now
/com.example.myapplication D/the displayreading2 frag:: the yaw is stationary now
/com.example.myapplication D/the displayreading2 frag:: finally stationary
```

Figure 5-14 Stationary 2

5.4.2.5 Up/Down

The algorithm implemented in this application has been developed in order to detect the signatures of up and down that will affect the angles of the sensor to change accordingly to the movement. For the initialisation of the raw data received and the angle functions please refer to section 9.4.5.2 in the Appendix.

The sensor was implemented into 3 locations and repeated on the other leg with 4 different possible scenarios. The sensor tag power button facing up, down, sideways and sideways reverse. This will enable the user to implement the Sensor tag at any given location without affecting the application and the detection algorithm. The positions of the sensor tag implemented are front, left and right as the sensor tag could potentially be damaged if stationed at the back of the patients' leg. The algorithm for the front up will be displayed and based on the same procedure the rest of the detection algorithms are implemented as for down the reverse of the algorithm has been utilised.

```
                    boolean fpwup = false;
                    boolean fpar = false;
                    boolean fpwdown = false;
                    boolean frevpar = false;

                    boolean lpwup = false;
                    boolean lpar = false;
                    boolean lpwdown = false;
                    boolean lrevpar = false;

                    boolean rpwup = false;
                    boolean rpar = false;
                    boolean rpwdown = false;
                    boolean rrevpar = false;

//works great
                    if ((f1pitchuptodown
                            && ((f1roll1 >= f1roll8) || (f1roll1 <= f1roll8))
                            && (f1yawuptodownflip || f1yawuptodown || f1yawdowntoup))
                            || (f2pitchuptodown
                            && ((f2roll1 >= f2roll8) || (f2roll1 <= f2roll8))
                            && (f2yawuptodownflip || f2yawuptodown || f2yawdowntoup))
                            || (f3pitchuptodown
                            && ((f3roll1 >= f3roll8) || (f3roll1 <= f3roll8))
                            && (f3yawuptodownflip || f3yawuptodown || f3yawdowntoup))) {

                        if ((!isStationary) && (!stationary)) {

                            Log.d(TAG, "it's up front for pwup");
                            fpwup = true;
                        }
                    }

//works great
                    else    if (((f1pitch1 >= f1pitch8) || (f1pitch1 <= f1pitch8))
                            && (f1rolldowntoup
                            && f1yawuptodownflip || f1yawdowntoup)
                            || (((f2pitch1 >= f2pitch8) || (f2pitch1 <= f2pitch8))
                            && f2rolldowntoup
                            && f2yawuptodownflip || f2yawdowntoup)
                            ||(((f3pitch1 >= f3pitch8) || (f3pitch1 <= f3pitch8))
                            && f3rolldowntoup
                            && f2yawuptodownflip || f3yawdowntoup)) {
                        if ((!isStationary) && (!stationary)) {
                            Log.d(TAG, "it's up front for par");
                            fpar = true;
                        }
                    }
```

Figure 5-15 Front Power Up and Power Parallel

```
//works great
                    else   if ((f1pitchdowntoup
                            && ((f1roll1 >= f1roll8) || (f1roll1 <= f1roll8))
                            && (f1yawuptodown || f1yawdowntoup))
                            || (f2pitchdowntoup
                            && ((f2roll1 >= f2roll8) || (f2roll1 <= f2roll8))
                            && (f2yawuptodown || f2yawdowntoup))
                            || (f3pitchdowntoup
                            && ((f3roll1 >= f3roll8) || (f3roll1 <= f3roll8))
                            && (f3yawuptodown || f3yawdowntoup))) {
                        if ((!isStationary) && (!stationary)) {

                            Log.d(TAG, "it's up front for pwdown");

                            fpwdown = true;
                        }
                    }
//works great
                    else   if (((f1pitch1 >= f1pitch8) || (f1pitch1 <= f1pitch8))
                            && f1rolluptodown
                            && (f1yawdowntoupflip || f1yawuptodown)
                            ||(((f2pitch1 >= f2pitch8) || (f2pitch1 <= f2pitch8))
                            && f2rolluptodown
                            && f2yawdowntoupflip || f2yawuptodown)
                            || (((f3pitch1 >= f3pitch8) || (f3pitch1 <= f3pitch8))
                            && f3rolluptodown
                            && f3yawdowntoupflip || f3yawuptodown)) {
                        if ((!isStationary) && (!stationary)) {

                            Log.d(TAG, "it's up front for rev par");

                            frevpar = true;
                        }


                    }
```

Figure 5-16 Front Power Down and Power Reverse Parallel

Based on the same concept the rest of the algorithm has been implemented. The source code above illustrates how the sensor should respond at the 4 different positions at the front of the leg. The algorithm will trigger if any of these functions is true enabling the database to write a limited amount of times and reassuring that the movement has been made. The stationary algorithms are also utilised here as a failsafe method that the patient is indeed moving, and only then will the algorithm be true.

5.4.3 Steps

A steps algorithm has been developed by utilising multiple criteria and detection of peaks which will then increase the counter of steps.

```
if (running) {
                    double x = accelvalues[0];
                    double y = accelvalues[1];
                    double z = accelvalues[2];
                    acceleration.setText("X: " + x + "\nY: " + y + "\nZ: " + z);
                    // calculate the magnitude mag^2 = x^2 + y^2 + z^2 and add mag to the list
                    // we deal with mag due to count steps in all directions as magnitude neglects directions.
                    double mag = Math.sqrt(Math.pow(x, 2) + Math.pow(y, 2) + Math.pow(y, 2));
                    list.add(mag);
                    bigList.add(mag);
                    Log.d(TAG, "THIS IS WHAT it has in the main list" + bigList.toString());
                    endTime = System.currentTimeMillis();
                    time = (endTime - startTime) / 1000.0;
                    if ((time >= 2.0) && (bigList.size()==11)){
                        int stepsInTwoSeconds = getminiSteps(list);
                        steps2second.setText("#Steps in 2 second: " + stepsInTwoSeconds);

                        height = 1.75;
                        if (stepsInTwoSeconds > 0 && stepsInTwoSeconds <= 2) {
                            stride = height / 5;
                        } else if (stepsInTwoSeconds > 2 && stepsInTwoSeconds <= 3) {
                            stride = height / 4;
                        } else if (stepsInTwoSeconds > 3 && stepsInTwoSeconds <= 4) {
                            stride = height / 3;
                        } else if (stepsInTwoSeconds > 4 && stepsInTwoSeconds <= 5) {
                            stride = height / 2;
                        } else if (stepsInTwoSeconds > 5 && stepsInTwoSeconds <= 6) {
                            stride = height / 1.2;
                        } else if (stepsInTwoSeconds > 6 && stepsInTwoSeconds < 8) {
                            stride = height;
                        } else if (stepsInTwoSeconds >= 8) {
                            stride = 1.2 * height;
                        }
                        distance += stepsInTwoSeconds * stride;
                        speed = stepsInTwoSeconds * stride / 2.0;
                        speedtext.setText("Speed: " + speed + " m/s");
                        distanceText.setText("Distance: " + distance + " m");
                        steps.setText("#Steps: " + getSteps(bigList));
                        //Log.d(TAG, "THIS IS WHAT it has in the main list after "+bigList.toString());
                        startTime = endTime;


                    }
                    //steps.setText("#Steps: " + getSteps(bigList));
                    //Log.d(TAG, "THIS IS WHAT it has in the main list after "+bigList.toString());

                }
```

Figure 5-17 Retrieve data start collecting

This function utilises height in order to determine the distance travelled by an individual as well as the angular velocity of an individual in order to better determine the steps being implemented. A counter has been set for every two seconds or 10 readings which would try to capture peaks and display them to the user in a format of footsteps, speed and distance covered. This algorithm enables live-time data being submitted and shown to the user, in order to understand how many steps, the user has made throughout his exercise period. It shows how many steps the person made in those two seconds and at the bottom of the Fragment, it illustrates the total amount of steps made on that specific day.

```
private int getminiSteps(List<Double> list) {
    StatisticsUtil su = new StatisticsUtil();
    double mean = su.findMean(list);
    double std = su.standardDeviation(list, mean);
    int stepsNumber = su.finAllPeaks(list, std);
    list.clear();
    return stepsNumber;
}


private int getSteps(List<Double> list) {
    if (list.size() > 10) {
        StatisticsUtil su = new StatisticsUtil();
        double mean = su.findMean(list);
        double std = su.standardDeviation(list, mean);
        int stepsNumber = su.finAllPeaks(list, std);
        //list.clear();
        stepno += stepsNumber;

    }
    list.clear();
    return stepno;
}
```

Figure 5-18 Get steps

This function is utilising the class below in order to identify steps and illustrate them back to the user.

```
01.  public class StatisticsUtil {
02.
03.      public double findMean(List<Double> list) {
04.          double total = 0;
05.          for (int i = 0; i < list.size(); i++) {
06.              total += list.get(i);
07.          }
08.          return total / list.size();
09.      }
10.
11.      public double standardDeviation(List<Double> list, double mean) {
12.          double sum = 0;
13.          for (int i = 0; i < list.size(); i++) {
14.              list.set(i, Math.pow(list.get(i) - mean, 2));
15.              sum += list.get(i);
16.          }
17.          return Math.sqrt(sum / list.size());
18.      }
19.
20.      public int finAllPeaks(List<Double> list, double minPeak) {
21.          int counter = 0;
22.          for (int i = 0; i < list.size(); i++) {
23.              if (i + 2 < list.size()) {
24.                  double one = list.get(i), two = list.get(i + 1), three = list.get(i + 2);
25.                  if (one < two && two > three && two > minPeak) {
26.                      counter++;
27.                  }
28.              }
29.          }
30.          return counter;
31.      }
32.
33.  }
```

Figure 5-19 Find Peaks, Standard Deviation

This class identifies the number of steps made by utilising the raw-data list and incrementing a counter when the threshold has been passed. In order to enable the doctor and the user to see their steps a function to submit to the database has been made which enables the user to go back in time and see their daily workout as shown below.

```
01.      private void pushtodb() {
02.          long date = System.currentTimeMillis();
03.          SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
04.          final String dateString = sdf.format(date);
05.
06.          final HashMap<String, Object> stepshashmap = new HashMap<>();
07.          stepshashmap.put("stepsmade", stepno);
08.          stepshashmap.put("date", dateString);
09.          stepshashmap.put("user", fuser.getUid());
10.
11.
12.          document = FirebaseFirestore.getInstance().collection("User").document(fuser.getUid()).collection("Steps").document(dateString);
13.          document.addSnapshotListener(new EventListener<DocumentSnapshot>() {
14.      @Override
15.      public void onEvent(@androidx.annotation.Nullable DocumentSnapshot documentSnapshot, @androidx.annotation.Nullable FirebaseFirestoreException e) {
16.          if (documentSnapshot.exists()) {
17.
18.              document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
19.                  @Override
20.                  public void onComplete(@NonNull Task<DocumentSnapshot> task) {
21.
22.                      if (task.isSuccessful()) {
23.                          final MyStepsModel mystepsModel = Objects.requireNonNull(task.getResult()).toObject(MyStepsModel.class);
24.                          assert mystepsModel != null;
25.                          Log.d(TAG, "steps already made are" + mystepsModel.getStepsmade());
26.                          int newsteps = (mystepsModel.getStepsmade() + stepno);
27.                          FirebaseFirestore.getInstance().collection("User").document(fuser.getUid()).collection("Steps")
28.  .document(dateString).update("stepsmade", newsteps);
29.                          stepno = 0;
30.                          distance = 0;
31.                          distanceText.setText("Distance: " + distance + " m");
32.                          steps.setText("#Steps: " + 0);
33.                      }
34.                  }
35.
36.              });
37.
38.
39.      }else {
40.              FirebaseFirestore.getInstance().collection("User").document(fuser.getUid())
41.              .collection("Steps").document(dateString).set(stepshashmap);
42.              stepno = 0;
43.              distance = 0;
44.              distanceText.setText("Distance: " + distance + " m");
45.              steps.setText("#Steps: " + 0);
46.          }
47.      }
48.  });
49.      }
```

Figure 5-20 Push to Database function

5.5 Summary

In this chapter, we have illustrated the implementation of the project which enables this application to be feasible. Each function of the application is clearly illustrated with source code and additional features have been implemented and shown in the Appendix. We have identified all the problems that arose in the implementation of the application and how those problems were solved by utilising algorithms accordingly. Some advantages have been identified and why those methods had been chosen instead of alternatives. In the next chapter, we will discuss the testing of the system.

Chapter 6 Testing

In this chapter we will be evaluating the application in order of functionality and whether it has passed the expected goals of the project. In the second part of this chapter the Human trials will take place in order to prove the functionality of the application in human environments where individuals will be instructed to follow some guidelines in order to test the application and its robustness, will also enable us to do integration testing where a near-complete application operates in accordance to end-users. A method of white box testing will be utilised since the developer is also the tester simultaneously. This will enable us to identify code completion. In white testing, the following apply, unit testing, integration testing and system testing. System testing will enable us to see whether the application has responded according to our objectives, goals and requirements.

6.1 Implemented Application Walkthrough

The Figures in this chapter are utilised in order to illustrate the application and its completeness. The Figures are divided based on each user and his functionalities. For the authentication Figures please refer to section 9.5.1 in the Appendix.

6.1.1 Patients

The general functions of the patient are displayed in the Appendix.
The Foundation functions can be found in the following table and are corresponding to their section in the Appendix.

Table 6-1 Patients Foundation Functions

ID	User	Functionality	Status	Section
1	Patient	Initial Fragment	Implemented	9.5.2.1
1	Patient	BLE Connection, Retrieve Data from Sensor	Implemented	9.5.2.2
2	Patient	Search all Doctors	Implemented	9.5.2.3
3	Patient	Edit Profile	Implemented	9.5.2.4
4	Patient	Chat/Communication, notifications, Doctor chat lists	Implemented	9.5.2.5
5	Patient	Make Requests	Implemented	9.5.2.6
6	Patient	Show my Requests	Implemented	9.5.2.7
7	Patient	Show my Doctors	Implemented	9.5.2.8

The key functions which are displaying the users' data and the collection of information for elevation are shown below.

6.1.1.1 Algorithm Fragments

Figure 6-1 First Fragment on connection

Figure 6-2 Madgwicks/Mahony algorithms

After the user has connected to the Sensor Tag the algorithms are stored separately in a different fragment. Above is the edit settings fragment as the initial one and the Madgwicks algorithm implemented on the right.

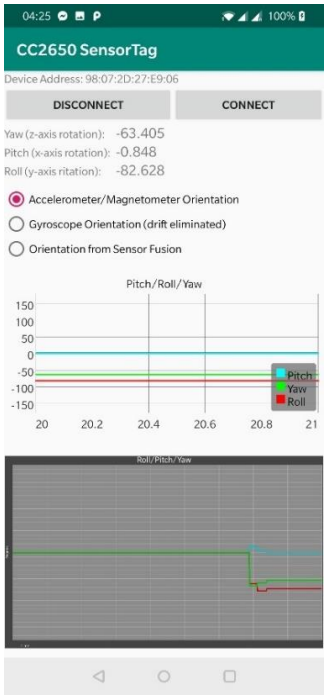

Figure 6-3 Google Functions and Matrixes

Figure 6-4 Kalman Filter

Above is the function of Google that has been utilised during this project in order to

measure elevation and showing the implementation of Kalman filter outputting orientation on angles.

6.1.1.2 Steps

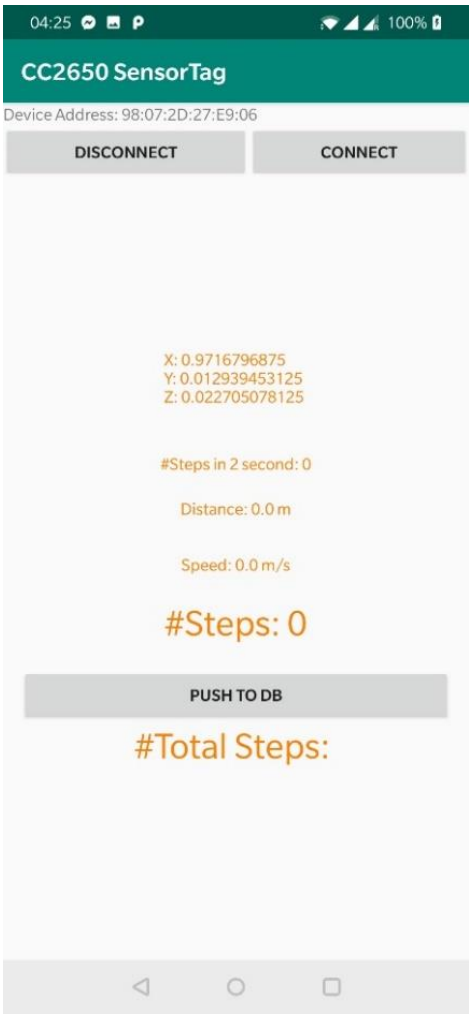

Figure 6-5 Steps Algorithm

6.1.1.3 My Data

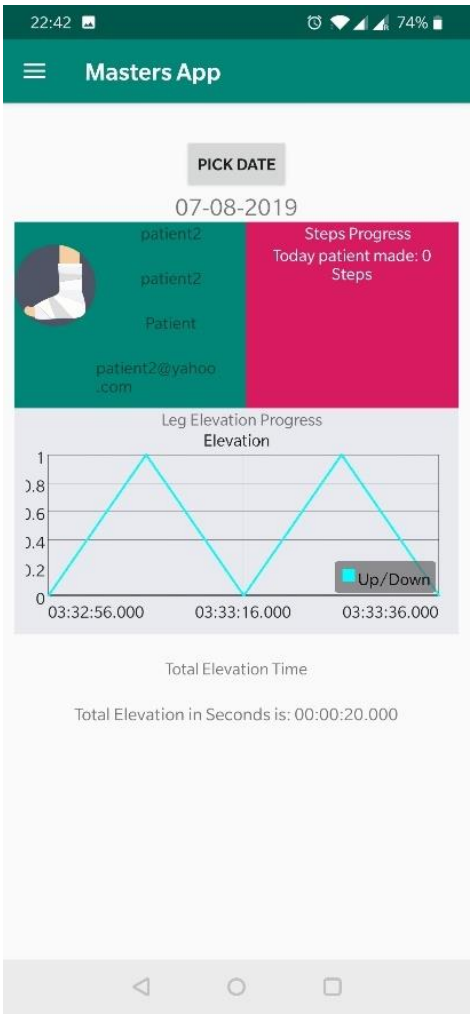

Figure 6-6 My Data

The functions above illustrate the users' steps and their Data which enable the user to comprehend their progress of either elevation or rehabilitation state. The data can be displayed based on timelines and enables the user to choose any date in the past to display his progress.

6.1.2 Doctors

Doctors can see some similar functions as the patient such as the Initial Fragment and Edit profile, although for Edit profile there is a minor difference. For Doctors, it will show their status whether they have been approved or not. For further information, refer to the table below where their corresponding functions are illustrated.

Table 6-2 Doctor Foundation Functions

ID	User	Functionality	Status	Section
1	Doctor	Chat/Communication, notifications, Patient chat lists	Implemented	9.5.3.1
2	Doctor	Approve Requests from patients	Implemented	9.5.3.2

The Doctor functions are shown below.

6.1.2.1 Patients Data

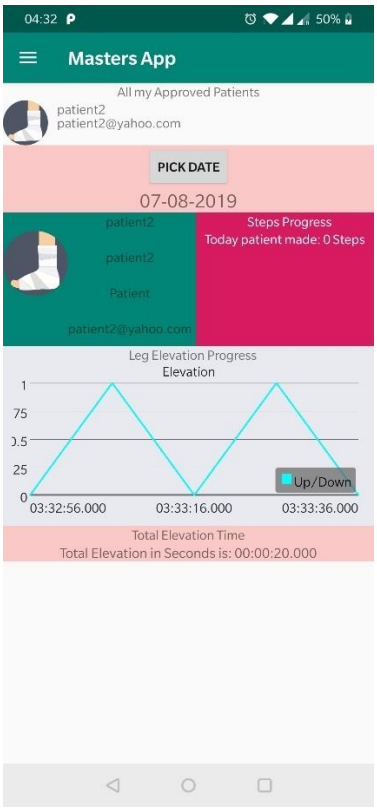

Figure 6-7 - Patient's Data

The patients' data is the most relevant information that the doctor is required to have. They are visible only to doctors who are supervising that specific patient and illustrate the steps and elevation time of the patient throughout the day. It gives them the ability to go to previous dates in order to keep track and comprehend what the user has been doing during past periods when the patient was following the doctors' instructions.

6.1.3 Administrator

The Administrator has one function available which is the to approve Doctors, clearly shown below.

6.1.3.1 Approve Doctor Requests

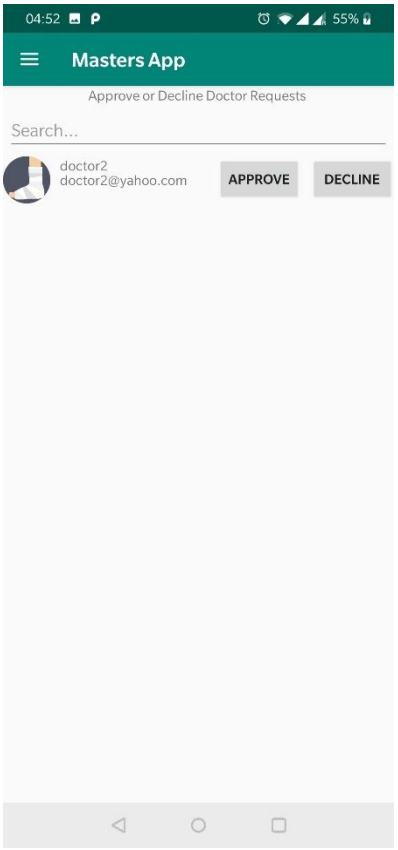

Figure 6-8 Approve Doctor Requests

By utilising this function, the Administrator gives the ability to doctors to carry out their medicinal practices.

6.1.4 System Testing

System Testing will enable us to show what has been achieved by this project and whether the user requirements of this application have been met. A comparison of the application with the expected outcome and the actual results was carried out as shown below.

Table 6-3 System Testing Table

ID	Description	Process	Expected Results	Actual Results	Pass/Fail
1	Measure Elevation	Patient Login -> BLE Connection to Device -> Utilise algorithm from DisplayReadings2.Java	Measure elevation and post to the database	As anticipated	Pass
2	Measure Steps	Patient Login ->BLE Connection to Device -> slide to the last fragment -> push to Database when done or display today's steps	Measure steps and post to the database	As anticipated	Pass
3	Display elevation to Patient	Patient Login -> My Data	Display data of the elevation	As anticipated	Pass
4	Display elevation to Approved Doctor	Doctor Login -> My Patients -> Tap on patient	Display patients' data of elevation	As anticipated	Pass
5	Do not Display data to unauthorised users	Only users that made the data and their doctors can see their information	Displays data to appropriate users	As anticipated	Pass
6	Log in With Correct information	Sign In/ Sign up or utilise Social media buttons	Log in with the information provided as previous users or signup with their information as new users	As anticipated	Pass

7	Edit current user information	User Login -> Edit Profile	Edit information of the user	As anticipated	Pass
8	Retrieve information on specified dates	Patient Login -> MY Data -> Time Picker -> choose date accordingly. Doctor Login -> My Patients -> tap accordingly and then utilise Time Picker accordingly.	Show information from various time periods	As anticipated	Pass
9	Dynamic graph for elevations	Patient Login -> My Data, implemented graph OR Patient Login -> BLE Connection to Device and see fragment DisplayReadings2.java for live data. Doctor Login -> My Patients -> tap accordingly -> view implemented graph.	Show graphs for the data to make it more readable	As anticipated	Pass
10	Total Elevation time	Calculate elevation time based on information of the database	Show the total time of elevation	As anticipated	Pass
11	Long term operation	Utilise BLE and stop functions when the phone is on standby	Minimise strain on mobile phone and battery	As anticipated	Pass
12	Communication/ Chat	Patient Login -> Chats Doctor Login -> Chats	Communicate with the appropriate individuals	As anticipated	Pass

13	Password forgotten function for account restoration	Forgotten Password -> Provide Email -> Submit	Change the password by utilising functions from Firebase	As anticipated	Pass
14	Signup through social media	Click Social Media Buttons and follow the process	Login/Signup with Social Media	As anticipated	Pass
15	Implement hierarchy, Administrator, Doctor, Patient	Utilise Hide Items Function	Differentiate the users based on their access policy	As anticipated	Pass
16	Approve functions for users accordingly	Administrator Login -> Approve Doctors Doctor Login -> Approve Patients	Approve requests accordingly	As anticipated	Pass
17	See and make requests. See the status of the request from Doctor to Administrator and from Patient to Doctor	Doctor Login –> Edit Profile Patient Login -> My Requests Patient Login -> Make Requests	Display status of users	As anticipated	Pass
18	See my patients	Doctor Login -> My Patients	Display my patients	As anticipated	Pass
19	See my doctors	Patient Login -> My Doctors	Display my doctors	As anticipated	Pass
20	Message notifications	Send message notification through functions of firebase	Send message notifications	As anticipated	Pass

21	See all Doctors	Communicate with doctors before making a request	Communicate with the doctor before making the request	As anticipated	Pass
22	Modify speed and Wake on Shake	Patient Login -> BLE Connection -> Initial Fragment	Enable the user to modify the speed of the readings and enable battery-saving features.	As anticipated	Pass

The above table illustrates the multiple results into producing a sufficient system testing table, below are the 5 values that comprise the system testing.

- Description: Is to indicate what is the actual requirement that has been set and provide a title for this requirement. They have been retrieved from the functional requirements in the "Requirements Chapter 3".
- Process: Is the method required to test the application on the specified requirement and the way to carry it out.
- Expected Results: Is what is expected from the application.
- Actual Results: Is the original result of the application.
- Pass/Fail: Shows whether the test was a success or a failure in meeting the requirements.

All the requirements of the application have been met in order to produce a sufficient outcome for the customer to receive this application.

Chapter 7 Evaluation

Evaluation is a chapter to illustrate the success of the application and whether the application is fully functional, usable and, how well it performs. It will illustrate the advantages of utilising this application and how this can benefit the medical sector in general along with the results of the human trials testing.

Based on the results of our testing all the user requirements have been met and exceeded by implementing far more than necessary in order to have a more robust and thorough application. This application has been developed with performance and accuracy in mind, this is due to the reason for utilising the most appropriate algorithms that also eliminate drift and operate on mobile phones simultaneously.

Overall this is a novel application that enables the users to see the readings of patients and is easily adaptable to any future projects since this application is modular enabling the developers to implement any type of extra applications they require. The application has a minimum SDK of 19 which enables users to have access to this application even with old devices, therefore, enabling accessibility for up to 98% of the Android user base.

7.1 Requirements outcome/Results

Shown below is the table of the Functional Requirements and their status of completion.

Table 7-1 – System Requirements with Status

Requirements ID	Client Type	Requirements	Status Completed/ Tested
FR1	Patient/Doctor /Administrator	Functions for each individual must be differentiated in order to know what functions each user can do.	YES
FR2	Patient/Doctor /Administrator	Forgot password functions put in place in order to return accounts to their rightful owners.	YES
FR3	Doctor/Patient	Communication with the other party needs to be established	YES
FR4	Doctor/Patient	The system should implement login and signup functions that enable the users to be recorded and records kept in order to maintain accountability and to assign data to individuals.	YES

FR5	Doctor/Patient	See all assigned Doctors or patients to each individual.	YES
FR6	Doctor/Patient	The data need to be in a user-friendly and easy to access manner	YES
FR7	Doctor/Patient	Need to modify the profile in order to see how the patient sees the application for troubleshooting purposes.	YES
FR8	Doctor	Display the data of the patient in order to assess the situation prior to and after surgery. On real-time.	YES
FR9	Doctor	Request approval for Doctors from Administrator	YES
FR10	Doctor	There needs to be a method to approve patients in order to be able to see their data.	YES
FR11	Patient	Make requests to the doctors for approval. And see their state.	YES
FR12	Patient	See all available doctors and search and communicate with them in order to establish an initial contact before requesting their approval.	YES
FR13	Patient	Some modification of the sensor readings can be implemented in order to enable reduced battery usage.	YES
FR14	Developer	There needs to be an established database communication	YES
FR15	Developer	The system should implement appropriate algorithms to detect the various requirements	YES

Table 7-2 Non-Functional Requirements Status

Requirements ID	Requirements	Status Implemented/Tested
NFR1	The application must be available to the patient and the doctor 24/7	YES
NFR2	The application and the source code are easy to modify for further endeavours or further explorations of the project.	YES
NFR3	Security must be implemented in the application in order to alleviate any problems or any malicious activities.	YES
NFR4	The application must display the data in user-friendly graphs that enable the doctor or patients to fully comprehend their readings	YES
NFR5	Must implement functions that are not expensive in order to enable the users to utilise that application and its features with much less cost in comparison to any alternatives.	YES
NFR6	The application needs to adapt to all available Android devices no matter of scale.	YES
NFR7	Backups need to be made of the database for protection cases.	YES

7.2 Costs

The overall costs are illustrated in the table below and how these can potentially increase in time, based on the user base and usage of the application.

Table 7-3 Overall Costs

Device/Software	Description	Budget expense	Actual Cost
Mobile Phone	Utilised in order to test the application	£650	£0
Sensor Tag	Sensor for the user's readings	£35	£35
Developer Kit	Utilised to speak to the Sensor	£20	£20
Database	Utilise to enable interoperability between users	£1000	£0

Specialised software	Software utilised in order to communicate with the devices in order to gain information	£300	£0
Accounts	Accounts available and projects	£1000	£0
Restoration Capabilities	Restore Database if there is a malfunction	£1000	£0
Backups	Keep backups in case of malfunctions	£1000	£0
Total			£55

The fact that this is an application hosted on a free account, meant that the accounts' restrictions affected development. This opposed a requirement to make the application and algorithms with minimal access to the database. Posting raw data could render database usage costly and potentially be a costly endeavour if the project was commercialised. Since this application was developed with minimal reads/writes to the database, this enables the users and the corporation utilising this application to withstand a higher load of information.

Even though the raw data are not posted to the database, this algorithm can measure total elevation time accurately without sustaining the load of huge raw data files being inputted into the database. Thanks to this methodology this will enable this application to remain cost-effective with longer usability on the free plan than an ordinary backup of the raw data. The system would be completely different in case the database enabled us unlimited reads and writes from and to the application. This would enable us to write the raw data of the application to the database and could potentially be a huge hassle for administrators to cope with such load of information in order to understand the reason for a problem occurring in any situation. Therefore, keeping the database well-structured and with minimal writes will enable the administrator to easily troubleshoot and fix any malfunctions that could occur in far faster speeds rather than an open database.

Shown below is the Gant chart utilised to keep track of the progress of this project. Due to this application having multiple implementations this project took longer than anticipated, in order to reach a satisfactory level.

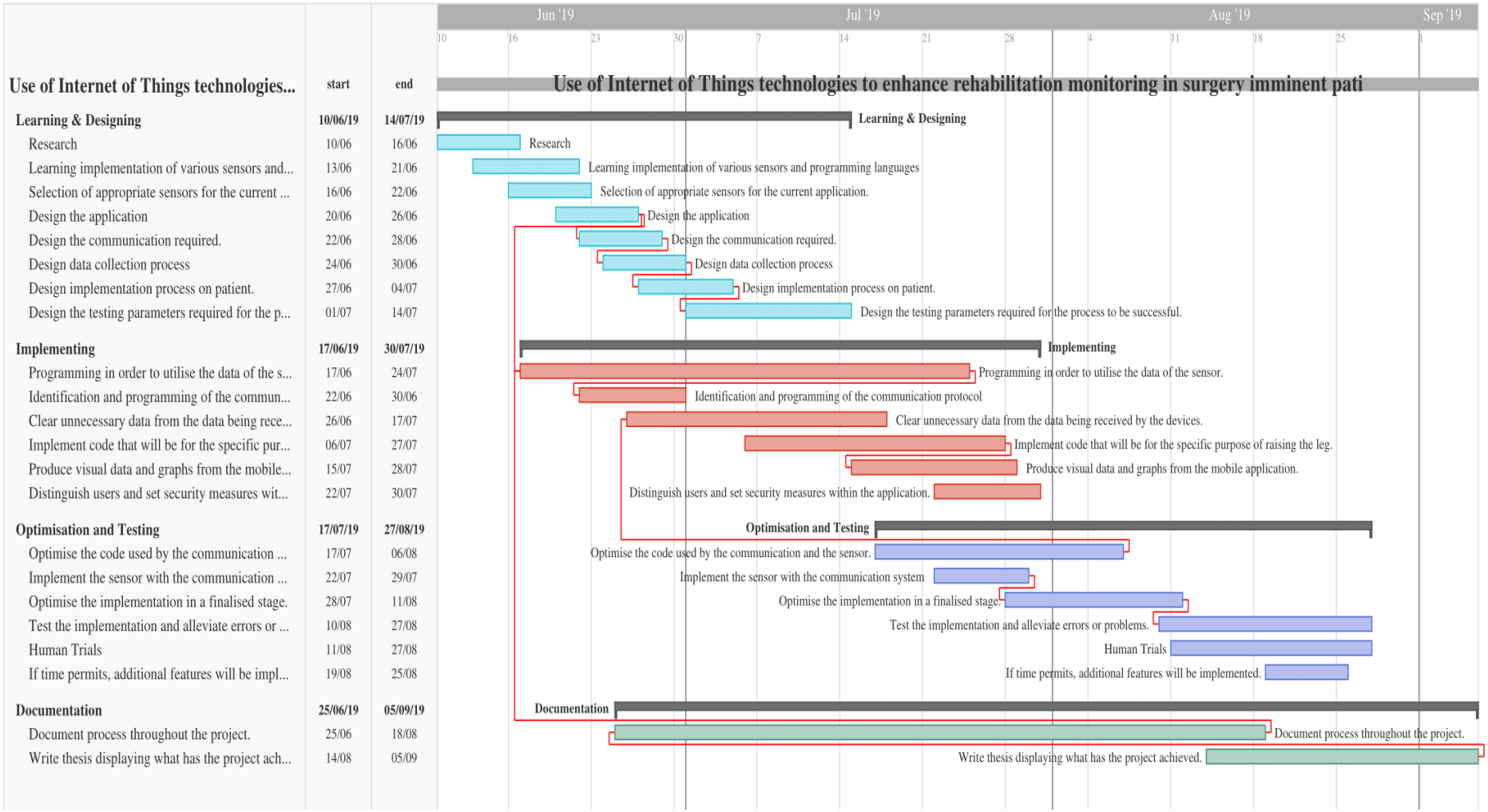

Figure 7-1 Gantt Chart

In order to alleviate these minor setbacks, the developer has enlisted the weekends of the project period in order to focus on developing a successful project that will meet all the requirements of the user and display a satisfactory outcome. Below is a table illustrating the multiple setbacks identified throughout the project and their alleviation methods.

Table 7-4 Setback Table

ID	Setback	Alleviation	Period -Lost/+Gained
1	Ergo application requiring revisions	Started the application early June and made the revisions immediately as soon as they had been identified	-5 +0
2	Algorithm errors - Madgwicks Algorithm had errors in the white paper which made the implementation much harder	Identified errors and fixed the algorithm to make this project feasible	-7/+5 Lost a week troubleshooting. Identified the problem and fixed the algorithm, along with that Mahony utilised a similar structure and made the implementation much faster
3	Fried SD Card - Fried my Raspberry Pi's SD card which the developer utilised at the start of the project in order to import Sensor Tag data to a Database	Kept backup of the iso file of the SD card which enabled a quick and fast replacement.	-3/+7 had back up of the SD card in an iso file simply restoring it and utilising a second SD card.
4	Experimental Libraries – graphing libraries in android are limited and scarce. The utilised graphing API is riddled with bugs that required troubleshooting.	Utilising experimental libraries as only a few alternatives exist.	-5/+3 Utilised different graphs and experimented continuously and identifying the patterns these libraries implemented. Made it faster to plot continuously.
5	Code Errors and missing code	Keeping a private repository in GitHub enabled me to keep it backed up with all the	-4/+6

		implementations and lost only minimal work throughout the project.	
Total			-24+21 = - 3

An overall acceptable result of the setbacks identified along with the mitigations methods enabling the developer to stay on track and on time with project delivery.

7.3 Benefits of this project

This project has been implemented with simplicity in mind for both the doctor and the patient and makes the data to the users readily available with minor effort. Since very minimal effort has been made in the past to create such a project, this is a prototype application that utilises multiple algorithms in order to produce a sufficient outcome. It enables the users to measure total elevation time as well as steps and displays this information back to the doctors for further diagnosis. There is currently no other platform to compare it against as something like this has not been developed for such specialised purposes making it a novel application.

7.3.1 Customisability and Extensibility

The application can enable for further enhancements to be inputted within the application and will enable the user to utilise it for further readings that are not necessarily restricted to a patient's legs and elevation. Since this application is modular, it enables the user to port multiple new features without much hassle in order to make this project more thorough.

7.4 Human Trials

Thanks to Ergo 2 approval we had the ability to test this implementation with human beings. The 8 individuals were guided in doing some simple movements that could trigger the elevation algorithm. Along with the elevation the human trials involved testing of the number of footsteps that had been done by those individuals. In order to test these functionalities, a plan was required. This enabled us to identify the overall robustness and accuracy of the application. You can find the full Ergo application within section 10 of the Appendix.

7.4.1 Testing Plan

For these algorithms to be tested, careful planning was required.

For Elevation algorithm, we had individuals raise their leg from a starting position of down to an elevated position on the table, whilst being in a sitting position, the test was also repeated while the individuals were standing. Therefore, testing when the algorithm triggered and if any maintenance to the algorithm was required or not. As for the "steps" algorithm, the testing plan implemented was that the individuals would have to go 10 steps which would

then be contrasted on the measure of steps outputted by the algorithm and whether those measurements were calculated correctly.
Which would indicate that the algorithm required finetuning.
Shown below are the results of the testing of the 8 individuals that partook place in this test.

Table 7-5 Human Trials/ Accuracy

User	Position of Sensor	Test Elevation standing Up.	Test Elevation siting Down.	Steps out of 10 Recorded	Total Elevation Time	Total Elevation Recorded
1	Left – Power Down	Success	Success	11	Approximately 30 seconds	00:31.791
2	Front – Power Up	Minor Error	Minor Error	8	Approximately 5 minutes	05:05.609
3	Right – Power Down	Success	Success	13	Approximately 30 seconds	00:33.754
4	Left – Power Down	Success	Success	11	Approximately 30 seconds	00:34.402
5	Front - Power Up	Success	Success	11	Approximately 1:30	01:29.333
6	Left – Power Up	Success	Success	12	Approximately 30 seconds	00:37.525
7	Right – Parallel	Success	Success	12	Approximately 30 seconds	00:31.004
8	Right - Power Up	Success	Failure	11	Approximately 1 minute	01:33.209

In order to further aid the implementation, there were several other functions recorded that would be implemented in order to test the robustness of the application.
Shown in the following table are the tests required to test for robustness.

Table 7-6 Human Trials / Robustness

User	Did the Application operate as expected?	If No state, WHY?	Are the information Shown appropriately?	What could be better?
1	Yes	-	Yes	-
2	No	Had problem with battery of sensor Tag and was not posting information	Was stopping receiving data from sensor.	Changed battery started working fine for the rest of the trials.
3	Yes	Shorter than average height made steps much higher than anticipated, although in general algorithm worked fine	Yes	-
4	Yes	-	No	Minor error with Graphview identified in GitHub. To alleviate simply re-choose the same date.
5	Yes	-	Yes	-
6	Yes	-	Yes	-
7	Yes	-	Yes	-
8	No	Detection of going down was not optimal when on the right-hand side of the leg.	Yes	Altered the algorithm and made it more susceptible to changes on that axis.

7.5 Summary

The Evaluation chapter has identified the status of the functional and non-functional requirements identified in the Requirements chapter 3 and whether they have been met in order to show them to the end-user. It also identified all the setbacks throughout the period of this project and alleviation methods required in order to make this project feasible. Extra features have been acknowledged and implemented, shown in the Appendix. Since no alternative exists for this project no comparison was available although making this a project that could be extended within the feature due to its modularity and extra features. The costs have been identified and elaborated as to how they could expand in the future based on the user base of the application.

Chapter 8 Conclusion

This chapter is made to illustrate the conclusion of this project along with discussion and possible future works that could be carried out on this application.

The application has been finalised and all requirements have been met. This makes this project a success in measuring elevation and steps. It has been identified as a cost-effective method that can fulfil a huge gap in the IoT market in the medicinal sector which could potentially have commercial value in the near future. This application is easily adaptable to any system that requires maintenance such as a hospital and data are easily integrated to any other type of database thanks to the versatility offered by the Google Firestore Database.

This application offers the user the ability to keep track of their patients and their doctors along with their data and their patients' data. There is a hierarchy established that enables people to have access by a higher tier individual before enabling them to carry out the related functions. The users can connect to sensors and see their information live while they are being recorded by the application in order to track their progress and enable them to either go for surgery or rehabilitate themselves at home. It also implements a user-friendly application that is straight to the point in showing data as medical personnel's time is critical and have minimal time to assess situations, therefore displaying the important information first makes this application easy to understand and makes the users knowledgeable of their situation instantaneously. The multiple algorithms implemented enabled this project to be a success and enabled the data being utilised as accurate as they can be due to the elaborate functions of those algorithms. There was a gap in the market for this application that could potentially help in aiding doctors and patients with their experience to the emergency rooms, minimising the hassle between the doctor and the patient. The project has been finalised well within the project time frame of 13 weeks and enabled this project to be realised. This is due to well organised time utilisation and effort in order to produce a sufficient outcome, this application is readily available to be implemented in a live scenario with human interaction. Overall this is a fascinating new chapter in the world of IoT and medicine which could potentially set a pathway for a greater project to commence.

8.1 Future Works

Even though this application provides all the requirements of the user there are many ways that this project could be expanded to further stages. Shown below are various pathways this project could be expanded on:

- An IOS application can be developed to establish a higher market value of the application.

- The algorithms implemented could be further tested and compared on their performance based on the current application.

- Newer hardware can be utilised such as the new Arduino Nano 33 BLE Sense which could enable this project to have even more accurate data.

- The project could be further extended in order to measure the total height of the elevated leg. This can be done by utilising the same method as this project with a fusion of the accelerometer and the barometer data in order to measure altitude. This could further enhance the current project as differential threshold would be implemented based on height.

Chapter 9 Appendix

9.1 Foundation Functionalities

The functionality of this project is based on multiple Activities that are being updated constantly through the utilisation of Fragments. These fragments provide the functionality to each user based on their type.

9.1.1 Administrator

The Administrator user is limited to one functionality which is the one that approves the Doctors. Shown below are the figures indicating how this is done.

9.1.1.1 Approve Doctors

The following figure illustrates how all the doctors who are not approved are presented within the list.

```
01.  private void readUsers() {
02.
03.      final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
04.      collection = FirebaseFirestore.getInstance().collection("User");
05.              collection.addSnapshotListener(new EventListener<QuerySnapshot>() {
06.                  @Override
07.                  public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
08.                      mUsers.clear();
09.                      assert queryDocumentSnapshots != null;
10.                      for (DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments()) {
11.                          UserModel userModel = documentSnapshot.toObject(UserModel.class);
12.                          assert firebaseUser != null;
13.                          assert userModel != null;
14.                          if (!userModel.getId().equals(firebaseUser.getUid()) && userModel.getType().equals("Doctor") && (!userModel.getApproved().equals("approved"))) {
15.                              mUsers.add(userModel);
16.                          }
17.                      }
18.
19.                      doctorAdapter = new DoctorAdapter(getContext(), mUsers);
20.                      recyclerView.setAdapter(doctorAdapter);
21.                  }
22.              });
23.          }
```

Figure 9-1 Non-Approved Doctors List

This source code illustrates how the application gathers all the doctors based on the status that they are not approved. These doctors will then be passed on to the Adapter that will populate the list to show to the Administrator for them to either approve or decline their request. Users that are being declined will be shown in case the administrator would like to approve the Doctors at a later stage.

```
holder.approve.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                HashMap<String, Object> hashMap = new HashMap<>();
                hashMap.put("approved", "approved");
                FirebaseFirestore.getInstance().collection("User").document(user.getId()).update(hashMap);
                Toast.makeText(mContext, "Doctor Approved", Toast.LENGTH_SHORT).show();


            }
        });

        holder.decline.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                HashMap<String, Object> hashMap = new HashMap<>();
                hashMap.put("approved", "declined");

                FirebaseFirestore.getInstance().collection("User").document(user.getId()).update(hashMap);
                Toast.makeText(mContext, "Doctor Denied", Toast.LENGTH_SHORT).show();

            }
        });

        holder.itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                Intent intent = new Intent(mContext, ChatActivity2.class);
                Bundle bundle = new Bundle();
                bundle.putString("USER_ID", (user.getId()));
                bundle.putString("USER_NAME", (user.getName()));
                intent.putExtras(bundle);
                view.getContext().startActivity(intent);


            }

        });
```

Figure 9-2 - Approve/Decline Doctors Adapter

This source code illustrates the functionalities available for the Administrator based on all the Doctors that are non-approved, the administrator has the ability to chat with the Doctor in order to inform him the reason his request has been rejected and also has the ability to provide photographic evidence for declining his application. On the other hand, the user on the list will be displayed with an Approve or Decline button that gives the Administrator the ability to act accordingly. If the Administrator has approved the Doctor, they will be upgraded from limited functionalities to the full functional doctor, that allows him to receive requests and approve them along with the ability to see their patients and their data.

9.1.2 Doctor

The doctor is comprised of multiple functions that will be illustrated below. In the non-approved state, the Doctor has the single ability to Edit his profile which also gives him the ability to do all the functions of a patient by changing his type.

9.1.2.1 Edit Profile

```
image_profile.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                openImage();

            }
        });
        Button button_submit = view.findViewById(R.id.button_submit);
        button_submit.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!validateForm()) {
                    return;
                } else {

                    document = FirebaseFirestore.getInstance().collection("User").document(Fuser.getUid());
                    current_user.setName(editText_update_name.getText().toString());
                    current_user.setSurname(editText_update_surname.getText().toString());
                    current_user.setSearch(editText_update_name.getText().toString().toLowerCase());
                    if (RadioGroupUpdate.getCheckedRadioButtonId() == R.id.ChoicePatient) {
                        current_user.setType("Patient");
                    } else if (RadioGroupUpdate.getCheckedRadioButtonId() == R.id.ChoiceDoctor) {
                        current_user.setType("Doctor");
                    }
                    document.set(current_user);
                    name.setText(current_user.getName());
                    hideItems();
                    Toast.makeText(getActivity(), "User information updated", Toast.LENGTH_SHORT).show();

                }

            }
        });
```

Figure 9-3 - Edit Profile

As shown in this figure the “Edit profile” Fragment has the function to modify the profile picture along with the data of the patient and his type. Fail-safe methods have been implemented whether the user has correctly filled the input forms and recalls the HideItems function in order to change how the application operates since the user has changed his type.

Once the Doctor has been approved, he gains multiple functionalities that enable him to examine his patient’s data more appropriately.

9.1.2.2 Communication/Chat

Chat is required by the Doctors and the Patients in order to establish communication between the two individuals, this will enable them to interact with one another in case the application indicates the Patient is not following the instructions provided by the doctor, or the patient is having problems with his leg or complications after his surgery. There are currently two activities comprising the Chat function, one of them houses the two fragments that will be displayed to each user. The fragments are My Patients in case of the Doctor, and My Doctors in case of the Patient. And the second fragment is the fragment that illustrates all the previous communications established.

9.1.2.2.1 Chat Activity

```
collection = FirebaseFirestore.getInstance().collection("Chats");

     collection.addSnapshotListener(new EventListener<QuerySnapshot>() {
         @Override
         public void onEvent(@javax.annotation.Nullable QuerySnapshot queryDocumentSnapshots, @javax.annotation.Nullable FirebaseFirestoreException e) {
             final ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(getSupportFragmentManager());
        int unread = 0;
             assert queryDocumentSnapshots != null;
             for (DocumentSnapshot document1 : queryDocumentSnapshots.getDocuments() ){
                 ChatModel chatModel = document1.toObject(ChatModel.class);
                 assert chatModel != null;
                 if (chatModel.getReceiver().equals(firebaseUser.getUid()) && !chatModel.isIsseen()){
                unread++;
            }
        }

        if (unread == 0 ){
            viewPagerAdapter.AddFragment(new ChatsFragment(), "Chats");
        } else if (unread == 1){
            viewPagerAdapter.AddFragment(new ChatsFragment(), "("+unread+") Chat");
        } else
            {viewPagerAdapter.AddFragment(new ChatsFragment(), "("+unread+") Chats");
        }
       document = FirebaseFirestore.getInstance().collection("User"). document(firebaseUser.getUid());
        document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
            @Override
            public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                final UserModel userModel = task.getResult().toObject(UserModel.class);
                assert userModel != null;
                if(userModel.getType().equals("Patient")){
                    viewPagerAdapter.AddFragment(new MyDoctorsChat(), "Doctors");
                    viewPagerAdapter.notifyDataSetChanged();
                }else if (userModel.getType().equals("Doctor")) {
                    viewPagerAdapter.AddFragment(new MyPatientsChat(), "Patients");
                    viewPagerAdapter.notifyDataSetChanged();


                }
            }

        });


        viewPager.setAdapter(viewPagerAdapter);

        tabLayout.setupWithViewPager(viewPager);

         }
     });
    }
```

Figure 9-4 Chat Activity populated

The figure above indicates how the two fragments are displayed based on the type of user which will keep everything neatly aligned with showing the appropriate information.

9.1.2.2.1.1 Chat - My Patients

The chat application has multiple functionalities ranging from notification for new messages to the online status being displayed and whether the patient has seen those messages, the following figure illustrates how these functions are feasible.

```
private void gettheusersonrequest(){
    mUsers.clear();
    myPatientsAdapterchat = new MyPatientsAdapterChat(getContext(), mUsers, false);
    recyclerView.setAdapter(myPatientsAdapterchat);
    for (int y = 0; y < theids.size();y++) {
        assert  document != null;
            document = FirebaseFirestore.getInstance().collection("User").document(theids.get(y));


            document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    final UserModel userModel = task.getResult().toObject(UserModel.class);
                    assert userModel != null;
                    if (mUsers.contains(userModel)) {
                        return;
                    } else {
                        mUsers.add(userModel);
                        Log.d(TAG, "this is what it passes for display " + userModel.getId());

                    }

                    myPatientsAdapterchat = new MyPatientsAdapterChat(getContext(), mUsers, false );

                    recyclerView.setAdapter(myPatientsAdapterchat);


                }


            });
        }


}
```

Figure 9-5 - Display all my patients

This application displays approved patients by finding all the available patients that have made requests to the specified doctor. A function is utilised prior calling the "getusersonrequest" in order to get all the users that are approved by this Doctor and are based under his collection, therefore minimising the read and write required by the application and also minimising the amount of time to process the list.

9.1.2.2.1.2 My Chat List

Followed by the chat list implementation there is a requirement for the application to show all the current and previous messages.

```
    private void chatList() {
        mUsers = new ArrayList<>();

        collection = FirebaseFirestore.getInstance().collection("User");
                collection.addSnapshotListener(new EventListener<QuerySnapshot>() {
                    @Override
                    public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                        mUsers.clear();

                        assert queryDocumentSnapshots != null;
                        for (DocumentSnapshot snapshots : queryDocumentSnapshots.getDocuments()) {
                            UserModel userModel = snapshots.toObject(UserModel.class);
                            for (String id : usersList) {
                                assert userModel != null;
                                if (userModel.getId().equals(id)) {
                                    if (mUsers.size() != 0) {
                                        for (UserModel user1 : mUsers) {
                                            if (!userModel.getId().equals(user1.getId())) {
                                                mUsers.add(userModel);
                                            }
                                        }
                                    } else {
                                        mUsers.add(userModel);
                                    }
                                }
                            }
                        }
                        userAdapter = new UserAdapter(getContext(), mUsers, true);
                        recyclerView.setAdapter(userAdapter);
                    }
                });
            }


}
```

Figure 9-6 - Chats List

This function is utilised in order to populate the chat list with all the users there has been communication with, to show to the user that there has been a previously established communication with this user. This function populates that list and passes it to the userAdapter which will then display the Last message received as well as the user that has been contacted. This minimises the unnecessary information for the Doctor with keeping the Chat list to the minimal, only showing the people that matter and enabling them to communicate with one another.

```
    final UserModel user = mUsers.get(position);
    holder.username.setText(user.getUsername());
    if (user.getImageURL().equals("default")) {
        holder.profile_image.setImageResource(R.mipmap.ic_launcher);
    } else {
        Glide.with(mContext).load(user.getImageURL()).into(holder.profile_image);
    }

    if (ischat) {
        lastMessage(user.getId(), holder.last_msg);
    } else {
        holder.last_msg.setVisibility(View.GONE);
    }

    if (ischat) {
        if (user.getStatus().equals("online")) {
            holder.img_on.setVisibility(View.VISIBLE);
            holder.img_off.setVisibility(View.GONE);
        } else if (user.getStatus().equals("offline")) {
            holder.img_on.setVisibility(View.GONE);
            holder.img_off.setVisibility(View.VISIBLE);
        }

    } else {
        holder.img_on.setVisibility(View.GONE);
        holder.img_off.setVisibility(View.GONE);
    }


    holder.itemView.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            Intent intent = new Intent(mContext, MessageActivity.class);
            intent.putExtra("userid", user.getId());
            mContext.startActivity(intent);
        }

    });
}
```

Figure 9-7 - Display the chat List

This function populates the list with all the users that there was established communication in previous times. It also sets the user online and offline depending on their status in the application.

```
private void lastMessage(final String userid, final TextView last_msg) {
      TheLastMessage = "default";
      final FirebaseUser firebaseUser = FirebaseAuth.getInstance().getCurrentUser();
      query = FirebaseFirestore.getInstance().collection("Chats").orderBy("timestamp", Query.Direction.ASCENDING);
              query.addSnapshotListener(new EventListener<QuerySnapshot>() {
                  @Override
                  public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                      assert queryDocumentSnapshots != null;
                      for (DocumentSnapshot document : queryDocumentSnapshots.getDocuments()) {
                          ChatModel chatModel = document.toObject(ChatModel.class);
                          assert firebaseUser != null;
                          assert chatModel != null;
                          if (chatModel.getReceiver().equals(firebaseUser.getUid()) && chatModel.getSender().equals(userid)||
                          chatModel.getReceiver().equals(userid) && chatModel.getSender().equals(firebaseUser.getUid())){
                      TheLastMessage = chatModel.getMessage();
                  }
              }

              if ("default".equals(TheLastMessage)) {
                  last_msg.setText("No Message");
              } else {
                  last_msg.setText(TheLastMessage);
              }
              TheLastMessage = "default";

                  }
              });
          }
```

Figure 9-8 - Display the Last message from that user

This function implements showing the last message received from those specified users.

9.1.2.2.2 Message Activity

This activity is utilised when the user sends messages to the other users whether they are their patients or their doctor. Shown below is a figure that illustrates the various functions called when a message has been sent.

```
01. private void sendMessage(String sender, final String receiver, String message) {
02.
03.       CollectionReference reference = FirebaseFirestore.getInstance().collection("Chats");
04.
05.       timestamp = System.currentTimeMillis();
06.       HashMap<String, Object> hashMap = new HashMap<>();
07.       hashMap.put("timestamp", timestamp);
08.       hashMap.put("sender", sender);
09.       hashMap.put("receiver", receiver);
10.       hashMap.put("message", message);
11.       hashMap.put("isseen", false);
12.       reference.document().set(hashMap);
13.
14.
15.
16.
17.
18.       final String msg = message;
19.
20.       document = FirebaseFirestore.getInstance().collection("User").document(fuser.getUid());
21.               document.addSnapshotListener(new EventListener<DocumentSnapshot>() {
22.                   @Override
23.                   public void onEvent(@Nullable DocumentSnapshot documentSnapshot, @Nullable FirebaseFirestoreException e) {
24.                       assert documentSnapshot != null;
25.                       UserModel userModel = documentSnapshot.toObject(UserModel.class);
26.                       if (notify) {
27.                           assert userModel != null;
28.                           sendNotification(receiver, userModel.getUsername(), msg);
29.                           Log.d(TAG,"it apparently sent nots "+    "+receiver+"    "+userModel.getUsername()+"  "+ msg);
30.                       }
31.                       notify = false;
32.
33.                   }
34.               });
35.           }
```

Figure 9-9 - Send Message Function

The send message function is responsible for posting to the database the message that has been sent to the other user and to enable the notification to be sent simultaneously to the user if the user is not online.

```
01. private void readMesagges(final String myid, final String userid, final String imageurl) {
02.       mchat = new ArrayList<>();
03.       query = FirebaseFirestore.getInstance().collection("Chats").orderBy("timestamp", Query.Direction.ASCENDING);
04.               query.addSnapshotListener(new EventListener<QuerySnapshot>() {
05.                   @Override
06.                   public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
07.                       mchat.clear();
08.                       assert queryDocumentSnapshots != null;
09.                       for (DocumentSnapshot queryDocumentSnapshot1 :queryDocumentSnapshots.getDocuments()) {
10.                           ChatModel chatModel = queryDocumentSnapshot1.toObject(ChatModel.class);
11.                           assert chatModel != null;
12.                           if (chatModel.getReceiver().equals(myid) && chatModel.getSender().equals(userid) ||
13.                                   chatModel.getReceiver().equals(userid) && chatModel.getSender().equals(myid)) {
14.                               mchat.add(chatModel);
15.                               Log.d(TAG, "Current data: " + queryDocumentSnapshot1.getData());
16.                           }
17.
18.
19.                           messageAdapter = new MessageAdapter(MessageActivity.this, mchat, imageurl);
20.                           messageAdapter.notifyDataSetChanged();
21.                           recyclerView.setAdapter(messageAdapter);
22.
23.                       }
24.                   }
25.               });
26.           }
```

Figure 9-10 populate Chat View

This function is made in order to populate with all the messages that had been sent or received by the currently logged in user and populate those messages in an orderly fashion by posting them through an adapter to organise them accordingly.

9.1.2.3 Approve Requests

Doctors have the ability to approve requests from patients in order to come under their supervision, this will enable the Doctors to only see the data of patients that are under their supervision, and also enables the patient to provide their approval for the Doctor to see their data.

```
    private void requestids() {
        theids = new ArrayList<>();
        Query query1 = FirebaseFirestore.getInstance().collection("User").document(fuser.getUid()).collection("ReceivedRequestIDs").whereEqualTo("status", "pending");
        query1.addSnapshotListener(new EventListener<QuerySnapshot>() {

            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                assert queryDocumentSnapshots != null;
                for (DocumentSnapshot queryDocumentSnapshots1 : queryDocumentSnapshots.getDocuments()) {

                    final RequestIDsModel requestIDsModel = queryDocumentSnapshots1.toObject(RequestIDsModel.class);
                    assert requestIDsModel != null;
                    requestusers = requestIDsModel.getSender();
                    requestIDs = requestIDsModel.getRequestID();

                    if (theids.contains(requestusers)){return;} else {
                        theids.add(requestusers);
                        Log.d(TAG, "the ids we got are" + theids);

                    }


                    }
                    gettheusersonrequest();

            }
        });
    }

//

private void gettheusersonrequest(){
        mUsers.clear();
    pushthedata();

    for (int y = 0; y < theids.size();y++){
            document = FirebaseFirestore.getInstance().collection("User").document(theids.get(y));
            document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    final UserModel userModel = task.getResult().toObject(UserModel.class);
                    assert userModel != null;
                    if (mUsers.contains(userModel)) {
                        return;
                    } else {
                        mUsers.add(userModel);
                        Log.d(TAG, "this is what it passes for display " + userModel.getId());

                    }


                    pushthedata();


                }


            });
                }

            }


    private void pushthedata(){
        approveRequestAdapter = new ApproveRequestAdapter(getContext(), mUsers);
        recyclerView.setAdapter(approveRequestAdapter);
    }
```

Figure 9-11 Get all Requests

The following functions sum up to all the requests that have been made by the users to this specific doctor, the doctor will be able to see all the requests that have been made by the users to him in order to approve several of them to come under his supervision or even decline some of them in order to focus on some specialised cases. Therefore, giving both the Patient and the doctor complete control of who sees their data as well as giving the doctor the power to choose what patients they can have. The adapter is also responsible to identify the information of the Patient available as in their name and surname along with their email address, this is for the doctor to further understand their relationship and make an educated choice of approving or decline. This is easily done by clicking the patient item which will pop up all the relevant information available.

9.1.2.4 My Patients

“My patients” function has similar functionality as the one showing the patients in the Chat List as shown by “Figure 5 6 - Display all my patients” this function is utilised in order to display all my patients with a slight alteration on the OnItemClick of the users. When an individual, clicks on the user-item a Popup will be displayed that implements several

functions. The function ranges from Date Picker, the number of steps made within a day, the elevation graphs and the total time of elevation. These are all implemented on the adapter of the patients which is illustrated to the administrator and is also able to see the data for each of the Patients he is supervising.

```
final View popupView = LayoutInflater.from(mContext).inflate(R.layout.popuprofilepatient, null);
                popupWindow = new PopupWindow(popupView, WindowManager.LayoutParams.MATCH_PARENT, WindowManager.LayoutParams.WRAP_CONTENT);
                final Button datedialog = popupView.findViewById(R.id.datedialog);
                datechosen = popupView.findViewById(R.id.datechosen);
                TextView name = popupView.findViewById(R.id.popupname);
                TextView surname = popupView.findViewById(R.id.popupsurname);
                TextView type = popupView.findViewById(R.id.popuptype);
                TextView email = popupView.findViewById(R.id.popupemail);
                totalelevation = popupView.findViewById(R.id.totalelevationtime);

                CircleImageView profpic = popupView.findViewById(R.id.popup_profile_image);
                graph = popupView.findViewById(R.id.patientview);

                patientsteps = popupView.findViewById(R.id.patientstepsmade);
                date = System.currentTimeMillis();
                final SimpleDateFormat sdf = new SimpleDateFormat("dd-MM-yyyy");
                dateString = sdf.format(date);
                Log.d(TAG, "the date is " + dateString);

                name.setText(user.getName());
                surname.setText(user.getSurname());
                type.setText(user.getType());
                email.setText(user.getUsername());
                if (user.getImageURL().equals("default")) {
                    profpic.setImageResource(R.mipmap.ic_launcher);
                } else {
                    Glide.with(mContext).load(user.getImageURL()).into(profpic);
                }


                popupWindow.setBackgroundDrawable(new ColorDrawable(
                        android.graphics.Color.TRANSPARENT));
                popupWindow.setFocusable(true);
                popupWindow.setBackgroundDrawable(new ColorDrawable());
                int location[] = new int[2];

                // Get the View's(the one that was clicked in the Fragment) location
                view.getLocationOnScreen(location);
                popupWindow.showAtLocation(view, Gravity.NO_GRAVITY,
                        location[0], location[1] + view.getHeight());
                popupWindow.setOutsideTouchable(true);


                datedialog.setOnClickListener(new View.OnClickListener() {
                    @Override
                    public void onClick(View view) {
                        showDatePickerDialog();



                    }
                });
```

Figure 9-12 - Popup

The popup is displayed as soon as a user clicks on his patients. This function creates the popup and sets the users information on it such as the name the email and surname. It also initialises the view of the popup and all the different items that are to be populated.

```
final Query query4 = FirebaseFirestore.getInstance().collection("User").document(user.getId()).collection("Steps");
query4.addSnapshotListener(new EventListener<QuerySnapshot>() {
    @Override
    public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
        for (DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments()) {
            MyStepsModel myStepsModel = documentSnapshot.toObject(MyStepsModel.class);
            assert myStepsModel != null;
            if (myStepsModel.getDate().equals(dateString)) {
                patientsteps.setText("Today patient made: " + myStepsModel.getStepsmade() + " Steps");


            } else patientsteps.setText("Today patient made: " + 0 + " Steps");

        }
    }

});
```

Figure 9-13 - Retrieve daily steps

This function retrieves the steps that had been done by the user on a specified day which allows the doctor to determine if the appropriate exercises have been made in order to comprehend in what stage of recovery is the patient and also whether he is following the instruction of the doctor.

```
Elevationline = new LineGraphSeries<>();
                graph.addSeries(Elevationline);
                graph.setTitle("Elevation");
                Elevationline.setTitle("Up/Down");
                Elevationline.setColor(Color.CYAN);
                graph.getLegendRenderer().setVisible(true);
                graph.getLegendRenderer().setAlign(LegendRenderer.LegendAlign.BOTTOM);
                graph.getGridLabelRenderer().setHumanRounding(false);
                graph.getViewport().setYAxisBoundsManual(true);
                graph.getViewport().setXAxisBoundsManual(true);
                graph.getViewport().setMinY(0);
                graph.getViewport().setMaxY(1);
                graph.getViewport().setScalable(true);
                graph.getViewport().setScrollable(true);
                graph.getGridLabelRenderer().setNumHorizontalLabels(3); // only 4 because of the space
                final SimpleDateFormat sdf2 = new SimpleDateFormat("hh:mm:ss.SSS");
                graph.getGridLabelRenderer().setLabelFormatter(new DefaultLabelFormatter() {
                    @Override
                    public String formatLabel(double value, boolean isValueX) {
                        if (isValueX) {
                            Log.d(TAG, "it goes in " + sdf2.format(new Date((long) value)));

                            return sdf2.format(new Date((long) value));
                        } else {
                            return super.formatLabel(value, isValueX);
                        }


                    }
                });

                query = FirebaseFirestore.getInstance().collection("User").document(user.getId()).collection("Elevation")
                .document(dateString).collection("Readings").orderBy("timestamp", Query.Direction.ASCENDING);

                query.addSnapshotListener(new EventListener<QuerySnapshot>() {
                    @Override
                    public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                        assert queryDocumentSnapshots != null;
                        for (DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments()) {
                            final MyElevationModel myElevationModel = documentSnapshot.toObject(MyElevationModel.class);
                            allelevations.add(myElevationModel);
                            Log.d(TAG, "The models i got is:" + myElevationModel);
                            assert myElevationModel != null;

                        }

                       reduceelevations();
                        for (int y = 0; y < reducedelevations.size(); y++) {
                            Elevationline.appendData(new DataPoint(reducedelevations.get(y).getTimestamp(), reducedelevations.get(y).getStatus()), true, 999999999);
                            graph.getViewport().setMinX(reducedelevations.get(0).getTimestamp());
                            graph.getViewport().setMaxX(reducedelevations.get(reducedelevations.size() - 1).getTimestamp());
                            graph.onDataChanged(false, false);
                        }
```

Figure 9-14 populate graph based on retrieved data

The graph is then populated and displayed to the user based on the formatting shown above. The display will be comprised of time on the x-axis and status on the y-axis, the way of time collecting has been made into milliseconds in order to accurately calculate the total time of the patient having the leg elevated as accurately as possible. In order to alleviate any values on the graph that might have duplicate timestamps. Even though the possibility of such occurrence being minimal, a function has been implemented to alleviate duplicate values being plotted on the graph in order to make the graph more understandable and more visually appealing as well as making the graph application more robust and error-free.

```
        for (int y = 0; y < allelevations.size(); y++) {
            if ((allelevations.get(y).getStatus()) == 1) {
                Log.d(TAG, "It got the up");
                Long current = allelevations.get(y).getTimestamp();
                if (y+1 < allelevations.size()) {
                    //if (allelevations.contains(allelevations.get(y + 1))) {
                    if (allelevations.get(y + 1).getStatus() == 1) {
                        Log.d(TAG, "this is up again " + allelevations.get(y + 1).getTimestamp());
                        Long next = allelevations.get(y + 1).getTimestamp();

                        totaltimeinmsecs += (next - current);
                    } else {
                        Long down = allelevations.get(y + 1).getTimestamp();
                        totaltimeinmsecs += down-current;
                    }

                }else {
                    Long currenttime = System.currentTimeMillis();
                    Long lastknownvalue = allelevations.get((allelevations.size() - 1)).getTimestamp();
                    totaltimeinmsecs += (currenttime - lastknownvalue);

                }

            }
        }
        Log.d(TAG, "The total time is: " + totaltimeinmsecs);
        int hours = (int) (totaltimeinmsecs / 3600000);
        int remainder = (int) (totaltimeinmsecs - hours * 3600000);
        int mins = remainder / 60000;
        remainder = remainder - mins * 60000;
        int secs = remainder / 1000;
        remainder = remainder - secs * 1000;
        int msecs = remainder;

        totalelevation.setText("Total Elevation in Seconds is: " +
        String.format("%02d",hours) + ":" +String.format("%02d",mins) + ":" + String.format("%02d",secs)+ "." + String.format("%03d",msecs));


    }
});
```

Figure 9-15 Total elevation time

This function calculates the total time that the leg has been elevated it gets the values that are currently stored in the database as soon as the values return to 0 which indicates the leg is down a timestamp is collected which when compared to the previous value enables us to receive the total difference which is the time the leg was elevated.

These functions are repeated on the moment the individual decides that need to visit a previous date to see his data. The patient and doctor both can see their current and past values and see their progress of how they have been doing on the elevation and whether they are complying with the doctors' instructions.

9.1.3 Patient

9.1.3.1 BLE Connection

The Bluetooth LE connection is utilised in order to connect to the Sensor Tag, it utilises the sensors of the mobile device and based on the devices that are available in the surrounding area that are openly using Bluetooth LE and are broadcasting their information, are than captured by this fragment and displayed in order to identify the Sensor Tag and receive the appropriate data. Shown below are the functions within the fragment that implement this functionality.

```
if (Build.VERSION.SDK_INT >= 21) {

    if(Build.VERSION.SDK_INT>=23){
        requestPermissions(new String[]{Manifest.permission.ACCESS_COARSE_LOCATION}, PERMISSION_REQUEST_COARSE_LOCATION);
    }
    mScanCallback = new ScanCallback() {
        @Override
        public void onScanResult(int callbackType, ScanResult result) {
            super.onScanResult(callbackType, result);
            mLeDeviceListAdapter.addDevice(result.getDevice());
            mLeDeviceListAdapter.updateRssiValue(result.getDevice(), result.getRssi());
            mLeDeviceListAdapter.notifyDataSetChanged();
        }
    };
} else {
    mLeScanCallback = new BluetoothAdapter.LeScanCallback() {
        @Override
        public void onLeScan(BluetoothDevice bluetoothDevice, int i, byte[] bytes) {
            mLeDeviceListAdapter.addDevice(bluetoothDevice);
            mLeDeviceListAdapter.notifyDataSetChanged();
        }
    };
}
// initialize bluetooth manager & adapter
BluetoothManager manager = (BluetoothManager) getActivity().getSystemService(Context.BLUETOOTH_SERVICE);
mBluetoothAdapter = manager.getAdapter();
```

Figure 9-16 - Start BLE session/Request permission

In this function, the application will request the necessary permissions in order to enable Bluetooth to operate all based on the version of the mobile operating system. At the point of discovering a device, the function will then inform the adapter to list it along with the relevant information such as the devices address, its name and RSSI.

```
listView = view.findViewById(R.id.BleDevices);
mLeDeviceListAdapter=new LeDeviceListAdapter(getContext());
listView.setAdapter(mLeDeviceListAdapter);
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
        Intent intent = new Intent(getContext(), SensorTagConnect.class);
        intent.putExtra("TheDevice", mLeDeviceListAdapter.getDevice(position));
        startActivity(intent);
        if (mBluetoothAdapter.isEnabled()) {
            if (Build.VERSION.SDK_INT < 21) {
                mBluetoothAdapter.stopLeScan(mLeScanCallback);
            } else {
                if (mLeScanner != null) {
                    mLeScanner.stopScan(mScanCallback);

                }
            }
        }


    }
});

startScan.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Log.d(TAG, "Pressed start button");
        if (Build.VERSION.SDK_INT < 21) {
            mBluetoothAdapter.startLeScan(mLeScanCallback);
        } else {
            // request BluetoothLeScanner if it hasn't been initialized yet
            if (mLeScanner == null) mLeScanner = mBluetoothAdapter.getBluetoothLeScanner();
            // start scan in low latency mode
            mLeScanner.startScan(new ArrayList<ScanFilter>(), new ScanSettings.Builder().setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY).build(), mScanCallback);
        }

        // Used for Low Energy Bluetooth
        scanLeDevice(true);


    }
});

stopScan.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        Log.d(TAG, "Pressed stop button");
        if (mBluetoothAdapter.isEnabled()) {
            if (Build.VERSION.SDK_INT < 21) {
                mBluetoothAdapter.stopLeScan(mLeScanCallback);
            } else {
                if (mLeScanner!= null) {
                    mLeScanner.stopScan(mScanCallback);

                }
                }
        }
        scanLeDevice(false);
        mLeDeviceListAdapter.clear();
        mLeDeviceListAdapter.notifyDataSetInvalidated();
        mLeDeviceListAdapter.notifyDataSetChanged();

    }
});
```

Figure 9-17 - Start/Stop BLE scan

```
private void scanLeDevice(final boolean enable) {
        if (enable) {
            Log.d(TAG, "Starting scan");
            mScanning = true;
            if (Build.VERSION.SDK_INT >= 21) {

                // start scan in low latency mode
                mLeScanner.startScan(new ArrayList<ScanFilter>(), new ScanSettings.Builder().setScanMode(ScanSettings.SCAN_MODE_LOW_LATENCY).build(), mScanCallback);
            }
            else{
                mBluetoothAdapter.startLeScan(mLeScanCallback);
            }

        } else {
            Log.d(TAG, "Stopping scan and refreshing list");
            mScanning = false;
            if (Build.VERSION.SDK_INT >= 21) {
                if (mLeScanner != null) {
                    mLeScanner.stopScan(mScanCallback);
                }
            }else{
                mBluetoothAdapter.stopLeScan(mLeScanCallback);
            }
        }
    }
```

Figure 9-18 Scanning Function

This function is required in order to enable the scanning and utilisation of BLE to operate.

9.1.3.2 Search Doctors

Search Doctors enables the Patient to communicate with the doctors in order to further understand their speciality and to ask whether they would like to become their supervisors in order to set some terms they would like. In the following figures, the search function is illustrated.

```
EditText search_users = view.findViewById(R.id.search_users);
        search_users.addTextChangedListener(new TextWatcher() {
            @Override
            public void beforeTextChanged(CharSequence s, int start, int count, int after) {

            }

            @Override
            public void onTextChanged(CharSequence s, int start, int before, int count) {
                searchUsers(s.toString().toLowerCase());
            }

            @Override
            public void afterTextChanged(Editable s) {

            }
        });


    private void searchUsers (String s) {
        final FirebaseUser Fuser = FirebaseAuth.getInstance().getCurrentUser();
        final Query query = FirebaseFirestore.getInstance().collection("User").orderBy("search")
                .startAt(s)
                .endAt(s+"\uf8ff");
        query.get().addOnCompleteListener(new OnCompleteListener<QuerySnapshot>() {
            @Override
            public void onComplete(@NonNull final Task<QuerySnapshot> task) {
               query.addSnapshotListener(new EventListener<QuerySnapshot>() {
                    @Override
                    public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                        mUsers.clear();
                         for (QueryDocumentSnapshot queryDocumentSnapshots1: task.getResult()) {
                             UserModel userModel = queryDocumentSnapshots1.toObject(UserModel.class);
                             assert Fuser != null;
                             if (!userModel.getId().equals(Fuser.getUid())&& userModel.getType().equals("Doctor") && userModel.getApproved().equals("approved")) {
                                 mUsers.add(userModel);
                             }
                         }


                         userAdapter = new UserAdapter(getContext(), mUsers, false);
                         recyclerView.setAdapter(userAdapter);
                    }

                });
            }
        });
    }
```

Figure 9-19 Search Doctors

This function takes into consideration the users input and reorders the list based on the item the user is searching on and populates the list of doctors accordingly.

9.1.3.3 Edit Profile

Edit Profile follows the same procedure as the Edit Profile in the doctor function as shown by "Figure 9 - 3 - Edit Profile".

9.1.3.4 Chat

Chat also follows the same procedure as indicated by the doctor although instead of displaying the "my Patients" Fragment it illustrates the "My Doctors" fragment. Which is clearly illustrated on "Figure 9-24 Approved Doctors & agreed on supervision"

9.1.3.5 Make Request to Doctors

This application will illustrate all doctors that are currently approved by the administrator and have not yet approved the patient to come under their supervision. This will populate a list item based on the current requests of the user and will only show the doctors that their request is not yet been approved, which will enable the items to be shown available for the patient in order to make their request. Shown below are the functions required for the requests to take place.

```
01. private void requestmade() {
02.
03.     final Query query = FirebaseFirestore.getInstance().collection("User").whereEqualTo("type", "Doctor").whereEqualTo("approved", "approved");
04.     query.addSnapshotListener(new EventListener<QuerySnapshot>() {
05.         @Override
06.         public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
07.             mUsers.clear();
08.             assert queryDocumentSnapshots != null;
09.             for (DocumentSnapshot documentSnapshot : queryDocumentSnapshots.getDocuments()) {
10.                 final UserModel userModel = documentSnapshot.toObject(UserModel.class);
11.                 Log.d(TAG, "Current data from USER QUERY: " + documentSnapshot.getData());
12.                 assert userModel != null;
13.                 if (!userModel.getId().equals(fuser.getUid())) {
14.                     mUsers.add(userModel);// put all doctors in}
15.
16.
17.                     final Query query = FirebaseFirestore.getInstance().collection("Requests");
18.                     query.addSnapshotListener(new EventListener<QuerySnapshot>() {
19.                         @Override
20.                         public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
21.                             assert queryDocumentSnapshots != null;
22.                             for (DocumentSnapshot documentSnapshot1 : queryDocumentSnapshots.getDocuments()) {
23.                                 final RequestsModel requestsModel = documentSnapshot1.toObject(RequestsModel.class);
24.                                 Log.d(TAG, "Current data from the Requests is : " + documentSnapshot1.getData());
25.
26.                                 if (requestsModel.getDoctorId().equals(userModel.getId()) && (requestsModel.getPatientId().equals(fuser.getUid()))
27.                                         && (((requestsModel.getStatus().equals("pending")) //made request
28.                                         || (requestsModel.getStatus().equals("approved")))) || (requestsModel.getDoctorId().equals(fuser.getUid())))) {
29.                                     Log.d(TAG, "it will remove this guy(s) from the model " + userModel.getId());
30.                                     mUsers.remove(userModel);
31.
32.                                 }
33.
34.                             }
35.
36.                             makeRequestAdapter = new MakeRequestAdapter(getContext(), mUsers);
37.                             recyclerView.setAdapter(makeRequestAdapter);
38.                         }
39.                     });
40.
41.                 }
42.             }
43.         }
44.     });
45. }
```

Figure 9-20 Available Doctors

All available doctors will be implemented here and then passed on to the adapter in order to enable the user to make their requests accordingly. Shown below is the adapter functions to enable requests to be made.

```
holder.makeRequest.setOnClickListener(new View.OnClickListener() {
  @Override
  public void onClick(View v) {
      String current_user = FirebaseAuth.getInstance().getUid();
      final RequestsModel newrequest;
      long unixTime = System.currentTimeMillis() /1000;
      assert current_user != null;
      DocumentReference newrequestid =  FirebaseFirestore.getInstance().collection("Requests").document();
      newrequest = new RequestsModel(newrequestid.getId(), current_user, user.getId(), unixTime , "pending",null);
      newrequestid.set(newrequest);
      HashMap<String, Object> requesthashmap = new HashMap<>();
      requesthashmap.put("RequestID", newrequestid.getId());
      requesthashmap.put("timestamp", unixTime);
      requesthashmap.put("status", "pending");
      requesthashmap.put("sender", current_user);
      requesthashmap.put("receiver", user.getId());
        if (FirebaseFirestore.getInstance().collection("User").document(current_user).collection("RequestIDs").document(user.getId()).get().isSuccessful()) {
            FirebaseFirestore.getInstance().collection("User").document(current_user).collection("RequestIDs").document(user.getId()).update(requesthashmap);
            FirebaseFirestore.getInstance().collection("User").document(user.getId()).collection("ReceivedRequestIDs").document(current_user).update(requesthashmap);
        }else
        {FirebaseFirestore.getInstance().collection("User").document(current_user).collection("RequestIDs").document(user.getId()).set(requesthashmap);
            FirebaseFirestore.getInstance().collection("User").document(user.getId()).collection("ReceivedRequestIDs").document(current_user).set(requesthashmap);}



      //FirebaseFirestore.getInstance().collection("Requests").document().set(newrequest);

      Toast.makeText(mContext, "Request to "+user.getName()+ " made", Toast.LENGTH_SHORT).show();
```

Figure 9-21 Make a request function

By pressing the request button on the available doctor, it will then trigger a write to the database that will trigger a request to be made to that doctor.

9.1.3.6 My Requests

My requests utilises a function that is being used by the doctors in order to see all the requests that have been sent to them, on this occasion the function has been slightly altered in order to see all the requests the user has made to any doctor and passes them to the adapter in order to display their status. The functions utilised here are illustrated below.

```
    private void requestids() {
        theids = new ArrayList<>();
        Query query1 = FirebaseFirestore.getInstance().collection("User").document(fuser.getUid()).collection("RequestIDs");
        query1.addSnapshotListener(new EventListener<QuerySnapshot>() {

            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                assert queryDocumentSnapshots != null;
                for (DocumentSnapshot queryDocumentSnapshots1 : queryDocumentSnapshots.getDocuments()) {

                    final RequestIDsModel requestIDsModel = queryDocumentSnapshots1.toObject(RequestIDsModel.class);
                    assert requestIDsModel != null;
                    mydoctorsrequest = requestIDsModel.getReceiver();


                    if (theids.contains(mydoctorsrequest)){return;} else {
                        theids.add(mydoctorsrequest);
                        Log.d(TAG, "the ids we got are" + theids);


                    }


                }
                gettheusersonrequest();

            }
        });
    }

//

    private void gettheusersonrequest(){
        mUsers.clear();
        myRequestsAdapter = new MyRequestsAdapter(getContext(), mUsers, java.util.Collections.emptyList());
        recyclerView.setAdapter(myRequestsAdapter);
        for (int y = 0; y < theids.size();y++) {
            assert  document != null;
                document = FirebaseFirestore.getInstance().collection("User").document(theids.get(y));


                document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                    @Override
                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                        final UserModel userModel = task.getResult().toObject(UserModel.class);
                        assert userModel != null;
                        if (mUsers.contains(userModel)|| userModel.getApproved().equals("False") ||  userModel.getType().equals("Patient")) {
                            return;
                        } else {
                            mUsers.add(userModel);
                            Log.d(TAG, "this is what it passes for display " + userModel.getId());

                        }

                        myRequestsAdapter = new MyRequestsAdapter(getContext(), mUsers, (theids));

                        recyclerView.setAdapter(myRequestsAdapter);


                    }


                });
            }
```

Figure 9-22 My requests

This function will pass all the users requests to the adapter which will then gather the status of those requests and then display its status accordingly to the doctors' answer. Illustrated below is the functions of the adapter.

```
query = FirebaseFirestore.getInstance().collection("User").document(user.getId()).collection("ReceivedRequestIDs")
.whereEqualTo("receiver", user.getId()).whereEqualTo("sender",fuser.getUid());
query.addSnapshotListener(new EventListener<QuerySnapshot>() {
                                @Override
                                public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {

                                    for (DocumentSnapshot queryDocumentSnapshots1 : queryDocumentSnapshots.getDocuments()) {
                                        final RequestIDsModel requestIDsModel = queryDocumentSnapshots1.toObject(RequestIDsModel.class);
                                        if (requestIDsModel.getSender().equals(fuser.getUid())&& requestIDsModel.getReceiver().equals(user.getId())){
                                            holder.status.setText(requestIDsModel.getStatus());
                                        }
                                    }
                                }
                            });
//holder.status.setText(statuses.get(position).toString());
Log.d(TAG, "this is what the postion is on the adapter " + position+"with this user "+ user.getName());
```

Figure 9-23 Request Status

9.1.3.7 My Doctors

My Doctors function is utilised in order to illustrate to the user all the doctors that have agreed to input them as being their patient. Shown below is the function that retrieves all the approved doctors and passes them to the adapter to be displayed.

```
    private void requestids() {
        theids = new ArrayList<>();
        thestatuses = new ArrayList<>();
        Query query1 = FirebaseFirestore.getInstance().collection("User").document(fuser.getUid()).collection("RequestIDs").whereEqualTo("status", "approved");
        query1.addSnapshotListener(new EventListener<QuerySnapshot>() {

            @Override
            public void onEvent(@Nullable QuerySnapshot queryDocumentSnapshots, @Nullable FirebaseFirestoreException e) {
                assert queryDocumentSnapshots != null;
                for (DocumentSnapshot queryDocumentSnapshots1 : queryDocumentSnapshots.getDocuments()) {

                    final RequestIDsModel requestIDsModel = queryDocumentSnapshots1.toObject(RequestIDsModel.class);
                    assert requestIDsModel != null;
                    if (requestIDsModel.getStatus().equals("approved")) {
                        mydoctors = requestIDsModel.getReceiver();
                    }

                    if (theids.contains(mydoctors)){return;} else {
                        theids.add(mydoctors);
                        Log.d(TAG, "the ids we got are" + theids);
                        Log.d(TAG, "the statuses we got are" + thestatuses);


                    }


                }
                gettheusersonrequest();

            }
        });
    }

//

    private void gettheusersonrequest(){
        mUsers.clear();
        myDoctorsAdapter = new MyDoctorsAdapter(getContext(), mUsers);
        recyclerView.setAdapter(myDoctorsAdapter);
        for (int y = 0; y < theids.size();y++) {
            assert  document != null;
            document = FirebaseFirestore.getInstance().collection("User").document(theids.get(y));


            document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                @Override
                public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                    final UserModel userModel = task.getResult().toObject(UserModel.class);
                    assert userModel != null;
                    if (mUsers.contains(userModel)|| userModel.getApproved().equals("False") ||  userModel.getType().equals("Patient")) {
                        return;
                    } else {

                        mUsers.add(userModel);
                        Log.d(TAG, "this is what it passes for display " + userModel.getId());

                    }

                    myDoctorsAdapter = new MyDoctorsAdapter(getContext(), mUsers );

                    recyclerView.setAdapter(myDoctorsAdapter);


                }


            });
        }


    }
```

Figure 9-24 Approved Doctors & agreed on supervision

9.1.3.8 My Data

Follows the same functionality as illustrated by the OnItemClick Functionality in My Patients in the doctor, although converted into a Fragment format as the requirement is only for the user that is currently logged in who is a patient.

9.2 Authentication/ Hide Items

```
public void onClick(View v) {
    int i = v.getId();
    if (i == R.id.signin) {
        // Get username, password from EditText
        final String username = editText_username.getText().toString();
        String password = editText_password.getText().toString();
        // Validate if username, password is filled
        if (validateEntry(username, password)) {
            showProgressDialog();
            mAuth.signInWithEmailAndPassword(username, password)
                    .addOnCompleteListener(this, new OnCompleteListener<AuthResult>() {
                        @Override
                        public void onComplete(@NonNull final Task<AuthResult> task) {
                            if (task.isSuccessful()) {
                                firebaseuser = mAuth.getCurrentUser();
                                assert firebaseuser != null;
                                document = FirebaseFirestore.getInstance().collection("User").document(firebaseuser.getUid());
                                document.get().addOnCompleteListener(new OnCompleteListener<DocumentSnapshot>() {
                                    @Override
                                    public void onComplete(@NonNull Task<DocumentSnapshot> task) {
                                UserModel userModel = task.getResult().toObject(UserModel.class);
                                Log.d(TAG, "the id that we got is  " + firebaseuser.getUid());
                                assert userModel != null;
                                session.createUserLoginSession(firebaseuser.getDisplayName(), firebaseuser.getEmail(), userModel.getApproved());
                                updateUI(firebaseuser);

                                    }
                                });



                                FirebaseInstanceId.getInstance().getInstanceId().addOnCompleteListener(new OnCompleteListener<InstanceIdResult>() {
                                    @Override
                                    public void onComplete(@NonNull Task<InstanceIdResult> task) {
                                        if (!task.isSuccessful()) {
                                            Log.w(TAG, "getInstanceId failed", task.getException());
                                            return;
                                        }

                                        String refreshToken = (task.getResult()).getToken();
                                        MyFirebaseInstanceIDService.Companion.updateToken(refreshToken);

                                    }
                                });

                            } else {
                                Toast.makeText(LoginActivity.this, "Authentication failure: Either username or password is wrong",
                                        Toast.LENGTH_SHORT).show();
                                updateUI(null);
                            }
                            hideProgressDialog();
                        }
                    });
        } else {
            Toast.makeText(this, "Please fill both username and password fields", Toast.LENGTH_SHORT).show();

        }
    }
```

Figure 9-25 - Login Function

The above source code has been utilised to get what the user has attempted to input to the login function and will attempt to sign in and authenticate the user if he exists. Fail-safe methods have been implemented in case the user has provided the wrong input or if the user exists.

If the user has forgotten their password, a separate Activity has been implemented in order to change their current password on the application. This comprises the user receiving an e-mail from the developers account that redirects them to a “change password website” which is implemented by Firebase and enables them to change their password. On the other hand, if the user knows all his details and exists on the database, they will be authenticated successfully and will be redirected to the Main activity page where the user will have the ability to carry out his allowed functions as clearly shown in the UML diagram in “Figure 5 - 1 Authentication”.

```
public void hideItems() {
        Log.d(TAG, "Hide items");
        if (current_user != null) {
            if (!session.isUserInfoFull())
                session.updateSession(current_user.getType(),current_user.getApproved());
            if (session.getUserDetails().get(KEY_TYPE).equals("Doctor") && session.getUserDetails().get(KEY_APPROVED).equals("approved")) {
                Log.d(TAG, "IT WENT INSIDEEEEEEE doctor 1");
                navigationView.getMenu().findItem(R.id.nav_scan_ble).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_searchdoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_profileedit).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_chat).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_doctorApprove).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_makerequest).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myrequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_ApproveRequests).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_myPatients).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_MyDoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_logout).setVisible(true);

            } else if  (session.getUserDetails().get(KEY_TYPE).equals("Doctor") && (!session.getUserDetails().get(KEY_APPROVED).equals("approved"))) {
                Log.d(TAG, "IT WENT INSIDEEEEEEE doctor 1");
                navigationView.getMenu().findItem(R.id.nav_scan_ble).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_searchdoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_profileedit).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_chat).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_doctorApprove).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_makerequest).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myrequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_ApproveRequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myPatients).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_MyDoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_logout).setVisible(true);

            }
            if (session.getUserDetails().get(KEY_TYPE).equals("Patient")) {
                Log.d(TAG, "IT WENT INSIDEEEEEEE patient 1");
                navigationView.getMenu().findItem(R.id.nav_scan_ble).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_searchdoctors).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_profileedit).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_chat).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_doctorApprove).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_makerequest).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_myrequests).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_ApproveRequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myPatients).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_MyDoctors).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_logout).setVisible(true);
            }
            if (session.getUserDetails().get(KEY_TYPE).equals("Administrator")) {
                Log.d(TAG, "IT WENT INSIDEEEEEEE Administrator 1");
                navigationView.getMenu().findItem(R.id.nav_scan_ble).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_searchdoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_profileedit).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_chat).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_doctorApprove).setVisible(true);
                navigationView.getMenu().findItem(R.id.nav_makerequest).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myrequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_ApproveRequests).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_myPatients).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_MyDoctors).setVisible(false);
                navigationView.getMenu().findItem(R.id.nav_logout).setVisible(true);
            }
        } else {
```

Figure 9-26 Hide Items function

Shown above is the source code for the differentiation of users.

9.3 Sensor Tag-Connect

In order to retrieve the values of the sensor tag, the open-source version of their application has been utilised, by minimising the functions utilised by the application to the ones required and then making the data to be broadcasted throughout the application. This will enable the application to be operating in the background without keeping the phone in constant operative/on mode and enables the data to be retrieved once and transmitted to multiple locations at the same time which will then enable all the function to operate simultaneously.

Although for the prototype, it was decided to stop the functions running when the application is paused, or the fragments are switched in order to maintain the fact that the user can see their data live. This is done in order to reduce power consumption and to enable the user to see all his visible movements based on the application. Shown below are the functions utilised to retrieve the data from the Sensor Tag.

There is a Service implemented that automatically detects all the available services on the connected device before populating the application with the correct fragments. This service utilises the connection in order to determine the Gatt services available on the device and is also responsible for connection and disconnection on these services along with the utilisation of those services such as modifying the speed of retrieving data. As soon as services are discovered, it broadcasts that this device has the available Gatt profiles that will enable this application to operate.

```
01.  public boolean connect(final String address) {              //33333333333
02.          if (mBluetoothAdapter == null || address == null) {
03.              //Log.w(TAG, "BluetoothAdapter not initialized or unspecified address.");
04.              return false;
05.          }
06.
07.          final BluetoothDevice device = mBluetoothAdapter.getRemoteDevice(address);
08.          int connectionState = mBluetoothManager.getConnectionState(device, BluetoothProfile.GATT);
09.
10.          if (connectionState == STATE_DISCONNECTED) {
11.
12.              // Previously connected device.  Try to reconnect. Current address equals previously connected address
13.              if (mBluetoothDeviceAddress != null && address.equals(mBluetoothDeviceAddress) && mBluetoothGatt != null) {
14.                  //Log.d(TAG, "Trying to use an existing mBluetoothGatt for connection.");
15.                  if (mBluetoothGatt.connect()) {
16.                      mConnectionState = STATE_CONNECTING;
17.                      return true;
18.                  } else {
19.                      //Log.w(TAG, "GATT reconnect failed");
20.                      return false;
21.                  }
22.              }
23.
24.              // Gets object for BLE device from its address
25.              if (device == null) {
26.                  //Log.w(TAG, "Device not found.  Unable to connect.");
27.                  return false;
28.              }
29.
30.              mBluetoothGatt = device.connectGatt(this, false, mGattCallback);
31.              //Log.d(TAG, "Trying to create a new connection.");
32.              mBluetoothDeviceAddress = address;
33.              mConnectionState = STATE_CONNECTING;
34.              return true;
35.
36.          } else {
37.              //Log.w(TAG, "Attempt to connect in state: " + connectionState);
38.              return false;
39.          }
40.
41.      }
```

Figure 9-27 Connect to Gatt Server on BLE Device

After this function there are multiple failsafe procedures that detect the state of the connection and whether there have been changes and acts according to the change.

```
@Override
        public void onConnectionStateChange(BluetoothGatt gatt, int status, int newState) {
            String intentAction;

            if (mBluetoothGatt == null) {
                //Log.e(TAG, "mBluetoothGatt not created!");
                return;
            }
                BluetoothDevice device = gatt.getDevice();
                String address = device.getAddress();

            if (newState == BluetoothProfile.STATE_CONNECTED) {
                //Log.i(TAG, "Connected to GATT server.");
                mConnectionState = STATE_CONNECTED;
                broadcastUpdate(ACTION_GATT_CONNECTED);


                /*
                 * Attempts to discover services after successful connection. Will report result through
                 * {@code BluetoothGattCallback#onServicesDiscovered(android.bluetooth.BluetoothGatt, int)} callback.
                 */
                //Log.i(TAG, "Attempting to start service discovery:");
                if (mBluetoothGatt.discoverServices()) {
                    //Log.i(TAG, "Has services");
                } else {
                    //Log.e(TAG, "NO SERVICES DISCOVERED");
                }

            } else if (newState == BluetoothProfile.STATE_DISCONNECTED) {
                //Log.i(TAG, "Disconnected from GATT server.");
                mConnectionState = STATE_DISCONNECTED;
                broadcastUpdate(ACTION_GATT_DISCONNECTED);
            } else {
                //Log.e(TAG, "Not connected or disconnected. Something went wrong");
            }
        }
```

Figure 9-28 Connection change Detection

```
public void onServicesDiscovered(BluetoothGatt gatt, int status) {
          BluetoothDevice device = gatt.getDevice();

          if (status == BluetoothGatt.GATT_SUCCESS) {
              broadcastUpdate(ACTION_GATT_SERVICES_DISCOVERED);
          } else {
              //Log.w(TAG, "onServicesDiscovered received: " + status);
          }
      }


      @Override
      public void onCharacteristicRead(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
         //Log.i(TAG, "Callback: characteristic read from " + characteristic.getUuid().toString() + "\nStatus: " + status);
          broadcastUpdate(ACTION_DATA_READ, characteristic, status);
      }

      @Override
      public void onCharacteristicWrite(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic, int status) {
          //Log.i(TAG, "Callback: Characteristic write from  " + characteristic.getUuid().toString() + "\nStatus: " + status);
          broadcastUpdate(ACTION_DATA_WRITE, characteristic, status);
      }


      @Override
      public void onCharacteristicChanged(BluetoothGatt gatt, BluetoothGattCharacteristic characteristic) {
          //Log.i(TAG, "Callback: characteristic changed " + characteristic.getUuid().toString());
          broadcastUpdate(ACTION_DATA_NOTIFY, characteristic, BluetoothGatt.GATT_SUCCESS);
      }


      @Override
      public void onDescriptorRead(BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status) {
          //Log.i(TAG, "Callback: Descriptor read from " + descriptor.getUuid().toString() + "\nStatus: " + status);
      }

      @Override
      public void onDescriptorWrite(BluetoothGatt gatt, BluetoothGattDescriptor descriptor, int status) {
          //Log.i(TAG, "Callback: Descriptor write from " + descriptor.getUuid().toString() + "\nStatus: " + status);

      }
  };
```

Figure 9-29 Services identified on BLE Device

After those services had been discovered a broadcast is sent to notify that the services had been found, a secondary broadcast is also implemented when those services are altered by any modifications we have made through the application.

```
01.  private void broadcastUpdate(final String action) {
02.        final Intent intent = new Intent(action);
03.        sendBroadcast(intent);
04.    }
05.
06.
07.    private void broadcastUpdate(final String action, final BluetoothGattCharacteristic characteristic, final int status) {
08.        final Intent intent = new Intent(action);
09.        intent.putExtra(IntentNames.EXTRA_UUID, characteristic.getUuid().toString());
10.        intent.putExtra(IntentNames.EXTRA_DATA, characteristic.getValue());
11.        intent.putExtra(IntentNames.EXTRA_STATUS, status);
12.        sendBroadcast(intent);
13.    }
```

Figure 9-30 Broadcasts

```
01.  public void changePeriod(BluetoothGattCharacteristic periodCharacteristic, byte p) {
02.
03.          byte[] val = new byte[1];
04.          val[0] = p;
05.
06.          periodCharacteristic.setValue(val);
07.          mBluetoothGatt.writeCharacteristic(periodCharacteristic);
08.          //Log.i(TAG, "Changing period of service " + periodCharacteristic.getUuid().toString());
09.      }
```

Figure 9-31 Altering Services

As previously stated, speed and wake on shake are functions available on the sensor tag for reduction of battery usage and for the speed of the data being streamed. Shown above is how speed is manipulated to send slower and faster. This function is utilised in a fragment that utilises this service in order to manipulate speed.

In order to determine which services, we are looking for we need to inform the application what are the various UUIDs of those services. Shown below are the ones being utilised.

```
01.  UUID_ACC_SERV = fromString("f000aa10-0451-4000-b000-000000000000"),
02.  UUID_ACC_DATA = fromString("f000aa11-0451-4000-b000-000000000000"),
03.  UUID_ACC_CONF = fromString("f000aa12-0451-4000-b000-000000000000"), // 0: disable, 1: enable
04.  UUID_ACC_PERI = fromString("f000aa13-0451-4000-b000-000000000000"), // Period in tens of milliseconds
05.
06.
07.
08.  UUID_MAG_SERV = fromString("f000aa30-0451-4000-b000-000000000000"),
09.  UUID_MAG_DATA = fromString("f000aa31-0451-4000-b000-000000000000"),
10.  UUID_MAG_CONF = fromString("f000aa32-0451-4000-b000-000000000000"), // 0: disable, 1: enable
11.  UUID_MAG_PERI = fromString("f000aa33-0451-4000-b000-000000000000"), // Period in tens of milliseconds
12.
13.
14.
15.  UUID_GYR_SERV = fromString("f000aa50-0451-4000-b000-000000000000"),
16.  UUID_GYR_DATA = fromString("f000aa51-0451-4000-b000-000000000000"),
17.  UUID_GYR_CONF = fromString("f000aa52-0451-4000-b000-000000000000"), // 0: disable, bit 0: enable x, bit 1: enable y, bit 2: enable z
18.  UUID_GYR_PERI = fromString("f000aa53-0451-4000-b000-000000000000"), // Period in tens of milliseconds
19.
20.
21.  UUID_MOV_SERV = fromString("f000aa80-0451-4000-b000-000000000000"),
22.  UUID_MOV_DATA = fromString("f000aa81-0451-4000-b000-000000000000"),
23.  UUID_MOV_CONF = fromString("f000aa82-0451-4000-b000-000000000000"), // 0: disable, bit 0: enable x, bit 1: enable y, bit 2: enable z
24.  UUID_MOV_PERI = fromString("f000aa83-0451-4000-b000-000000000000"), // Period in tens of milliseconds
```

Figure 9-32 UUIDS

The fact that the Sensor Tag does not differentiate the data and utilises the Sensors in such a way that movement data is sent instead of raw sensor data of the Gyroscope, Accelerometer and Magnetometer means that it requires conversion to take place in order to extract those values from the movement matrix received. This reduces the number of transmissions the Sensor Tag requires, although it requires a conversion method which can translate those values back to their original format. A function has been utilised by the open-source Sensor Tag application to carry out this function.

```
MOVEMENT_ACC(UUID_MOV_SERV, UUID_MOV_DATA, UUID_MOV_CONF, (byte) 3) {
        @Override
        public Point3D convert(final byte[] value) {
            // Range 8G
            final float SCALE = (float) 4096.0;

            int x = (value[7] << 8) + value[6];
            int y = (value[9] << 8) + value[8];
            int z = (value[11] << 8) + value[10];

            return new Point3D(((x / SCALE) * -1), y / SCALE, ((z / SCALE) * -1));
        }
    },
    MOVEMENT_GYRO(UUID_MOV_SERV, UUID_MOV_DATA, UUID_MOV_CONF, (byte) 3) {
        @Override
        public Point3D convert(final byte[] value) {


            final float SCALE = (float) 128.0;

            int x = (value[1] << 8) + value[0];
            int y = (value[3] << 8) + value[2];
            int z = (value[5] << 8) + value[4];
            return new Point3D(x / SCALE, y / SCALE, z / SCALE);

        }
    },
    MOVEMENT_MAG(UUID_MOV_SERV, UUID_MOV_DATA, UUID_MOV_CONF, (byte) 3) {
        @Override
        public Point3D convert(final byte[] value) {

            final float SCALE = (float) (32768 / 4912);
            if (value.length >= 18) {
                int x = (value[13] << 8) + value[12];
                int y = (value[15] << 8) + value[14];
                int z = (value[17] << 8) + value[16];
                return new Point3D(x / SCALE, y / SCALE, z / SCALE);
            } else return new Point3D(0, 0, 0);
        }
        }
```

Figure 9-33 - Movement Data conversion

In order to enable this application to function, a separate Activity has been created, which houses all the fragments that are required for this application to display the data. This Activity utilises all the previous services stated, in order to establish the communication of the Sensor Tag and to commence retrieving information from the device, in order to start retrieving data for measuring elevation. It is responsible for broadcasting through all the housed Fragments the information that it is retrieving.

```
01. private final BroadcastReceiver mGattUpdateReceiver = new BroadcastReceiver() {
02.     @Override
03.     public void onReceive(Context context, Intent intent) {
04.         final String action = intent.getAction();
05.
06.         if (BleService.ACTION_GATT_CONNECTED.equals(action))
07.         {
08.             Toast.makeText(SensorTagConnect.this, "Connected",Toast.LENGTH_SHORT).show();
09.
10.         } else if (BleService.ACTION_GATT_DISCONNECTED.equals(action))
11.         {
12.             Toast.makeText(SensorTagConnect.this, "Disconnected",Toast.LENGTH_SHORT).show();
13.
14.         } else if (BleService.ACTION_GATT_SERVICES_DISCOVERED.equals(action))
15.         {
16.             //Log.i(TAG, "Services discovered");
17.             setUpGattServices(mBleService.getSupportedGattServices());
18.
19.         } else if (BleService.ACTION_DATA_READ.equals(action))
20.         {
21.             //Log.d(TAG, "Data read");
22.
23.         } else if (BleService.ACTION_DATA_WRITE.equals(action))
24.         {
25.             //Log.d(TAG, "Data written");
26.
27.         } else if (BleService.ACTION_DATA_NOTIFY.equals(action))
28.         {
29.             /*
30.              * Data was changed and the appropriate fragment will be notified
31.              */
32.             String uuidStr = intent.getStringExtra(IntentNames.EXTRA_UUID);
33.             byte[] value = intent.getByteArrayExtra(IntentNames.EXTRA_DATA);
34.
35.             sendNotificationsToFragments(uuidStr, value);
36.
37.         } else
38.         {
39.             //Log.e(TAG, "Unknown received");
40.         }
41.     }
42. };
```

Figure 9-34 Receive Broadcasts from Device

This function retrieves broadcasts from the application which are related to the status of the device and what action has been done, for example, changes in services would trigger this broadcast just like connection status changes and all the services that have been identified will utilise this broadcast to inform the Activity and it's corresponding Fragments based on their subscription with the appropriate information.

```
01. private void sendNotificationsToFragments(String uuidStr, byte[] value) {
02.         Intent notifyIntent;
03.
04.
05.         if (uuidStr.compareTo(SensorTagGatt.UUID_MOV_DATA.toString()) == 0) {
06.             //Log.d(TAG, "Notify intent to motion");
07.             notifyIntent = new Intent(IntentNames.ACTION_MOV_CHANGE);
08.             notifyIntent.putExtra(IntentNames.EXTRAS_MOV_DATA, value);
09.             sendBroadcast(notifyIntent);
10.
11.         } else {
12.             //Log.e(TAG, "Notified something that isn't set up for");
13.         }
14.     }
```

Figure 9-35 Broadcast to Fragments

```
View.OnClickListener onClickListener = new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            switch(v.getId()) {
                case R.id.buttonconnect:
                    //Log.d(TAG, "connect pressed");
                    if (!mConnected) {
                        Log.d (TAG, mDeviceName+ "address2"+ mDeviceAddress);
                        mBleService.connect(mDeviceAddress);
                        mConnected = true;
                    } else {
                        Toast.makeText(SensorTagConnect.this, "Already connected", Toast.LENGTH_SHORT).show();
                    }
                    break;

                case R.id.buttondisconnect:
                    //Log.d(TAG, "disconnect pressed");
                    if (mConnected) {
                        mBleService.disconnect(mDeviceAddress);
                        mConnected = false;
                    } else {
                        Toast.makeText(SensorTagConnect.this, "Already disconnected", Toast.LENGTH_SHORT).show();
                    }
                    break;

                default:
                    //Log.e(TAG, "Error in clicking");
                    break;
            }
        }
    };
```

Figure 9-36 Connect/Disconnect Device

The most resource-intensive function is the setup of the services and also to initialise them, therefore in order to alleviate the hogging of the main thread a background thread is utilised in order to keep the application functional and also displays to the user the progress of these services identified.

```
private void setUpGattServices(final List<BluetoothGattService> gattServices) {
        if (gattServices == null) return;

        List<BluetoothGattCharacteristic> charList = new ArrayList<BluetoothGattCharacteristic>();


        // Loops through available Gatt services to find number of characteristics
        for (int i = 0; i < gattServices.size(); i++) {
            BluetoothGattService s = gattServices.get(i);
            List<BluetoothGattCharacteristic> c = s.getCharacteristics();
            if (c.size() > 0) {
                for (int jj = 0; jj < c.size(); jj++) {
                    charList.add(c.get(jj));
                }
            }
        }

        // Sets up UI progress dialog to display progress of background thread
        progressDialog = new ProgressDialog(SensorTagConnect.this);
        progressDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        progressDialog.setIndeterminate(true);
        progressDialog.setTitle("Discovering Services");
        progressDialog.setMessage("");
        progressDialog.setMax(100);
        progressDialog.setProgress(0);
        progressDialog.show();

        Thread worker = new Thread(new Runnable() {
            @Override
            public void run() {

                int nrNotificationsOn = 0;
                int maxNotifications;
                int servicesDiscovered = 0;
                int totalCharacteristics = 0;


                for (BluetoothGattService s : gattServices) {
                    List<BluetoothGattCharacteristic> chars = s.getCharacteristics();
                    totalCharacteristics += chars.size();
                }
                if (totalCharacteristics == 0) {
                    //Something bad happened, we have a problem
                    runOnUiThread(new Runnable() {
                        @Override
                        public void run() {
                            progressDialog.hide();
                            progressDialog.dismiss();
                            AlertDialog.Builder alertDialogBuilder = new AlertDialog.Builder(SensorTagConnect.this);
                            alertDialogBuilder.setTitle("Error !");
                            alertDialogBuilder.setMessage(gattServices.size() + " Services found, but no characteristics found, device will be disconnected !");
                            alertDialogBuilder.setPositiveButton("Retry", new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                }
                            });
                            alertDialogBuilder.setNegativeButton("Disconnect",new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which) {
                                    mBleService.disconnect(mDeviceAddress);
                                }
                            });
                            AlertDialog a = alertDialogBuilder.create();
                            a.show();
                        }
                    });
                    return;
                }
```

Figure 9-37 Progress Dialog and identify services

As soon as these functions operate then the second part of the function operates which enables all those services and initialises them in order to retrieve the Sensor information.

```
final int final_totalCharacteristics = totalCharacteristics;
              runOnUiThread(new Runnable() {
                  @Override
                  public void run() {
                      progressDialog.setIndeterminate(false);
                      progressDialog.setTitle("Generating GUI");
                      progressDialog.setMessage("Found a total of " + gattServices.size() + " services with a total of " +
                              final_totalCharacteristics + " characteristics on this device" );

                  }
              });


              for (int ii = 0; ii < gattServices.size(); ii++) {
                  BluetoothGattService service = gattServices.get(ii);
                  List<BluetoothGattCharacteristic> chars = service.getCharacteristics();
                  if (chars.size() == 0) {
                      //Log.e(TAG, "No characteristics found for this service !!!");
                      continue;
                  }
                  servicesDiscovered++;

                  final float serviceDiscoveredcalc = (float) servicesDiscovered;
                  final float serviceTotalcalc = (float) gattServices.size();

                  runOnUiThread(new Runnable() {
                      @Override
                      public void run() {
                          progressDialog.setProgress((int) ((serviceDiscoveredcalc / (serviceTotalcalc - 1)) * 100));
                      }
                  });


                  String serviceUUID = service.getUuid().toString();
                  //Log.d("DeviceActivity", "Configuring service with uuid : " + serviceUUID);

                   if (serviceUUID.compareTo(SensorTagGatt.UUID_ACC_SERV.toString()) == 0)
                  {
                      //Log.d(TAG, "ACCEL FOUND");
                      //see movement


                  } else if (serviceUUID.compareTo(SensorTagGatt.UUID_MAG_SERV.toString()) == 0)
                  {
                      //Log.d(TAG, "MAGNETOMETER FOUND");
                      //see movement

                  }  else if (serviceUUID.compareTo(SensorTagGatt.UUID_GYR_SERV.toString()) == 0)
                  {
                      Log.d(TAG, "GYROSCOPE FOUND");
                      //see movement

                  } else if (serviceUUID.compareTo(SensorTagGatt.UUID_MOV_SERV.toString()) == 0)
                  {
                      Log.d(TAG, "MOTION SENSOR FOUND");
                      mBleService.enableNotifications(service, SensorTagGatt.UUID_MOV_DATA);
                      safeSleep();
                      mBleService.enableMotionService(service, SensorTagGatt.UUID_MOV_CONF, true);
                      safeSleep();


                  } else {
                      //Log.e(TAG, "Unknown service");
                  }

              }
```

Figure 9-38 ForEach service implementation

```
69.             final int numOfFragments = 6;
70.             runOnUiThread(new Runnable() {
71.                 @Override
72.                 public void run() {
73.                     progressDialog.setTitle("Enabling Services");
74.                     progressDialog.setMax(numOfFragments);
75.                     progressDialog.setProgress(0);
76.
77.                     // Create the adapter that will return a fragment for each data in database
78.                     mSectionsPagerAdapter = new SectionsPagerAdapter(getSupportFragmentManager());
79.
80.                     // Set up the ViewPager with the sections adapter.
81.                     mViewPager = findViewById(R.id.container);
82.                     mViewPager.setAdapter(mSectionsPagerAdapter);
83.
84.                     //two fragments on either side; not applicable for small datasets
85.                     mViewPager.setOffscreenPageLimit(2);
86.                 }
87.             });
88.             for (int i = 0; i < numOfFragments; i++) {
89.
90.                 runOnUiThread(new Runnable() {
91.                     @Override
92.                     public void run() {
93.                         progressDialog.setProgress(progressDialog.getProgress() + 1);
94.                     }
95.                 });
96.
97.             }
98.             runOnUiThread(new Runnable() {
99.                 @Override
100.                public void run() {
101.
102.                    progressDialog.hide();
103.                    progressDialog.dismiss();
104.                }
105.            });
106.
107.            mBleService.changeIO(mBleService.getCharacteristicFromUUID(SensorTagGatt.UUID_TST_DATA.toString()), new byte[]{0});
108.
109.
110.
111.        }
112.    });
113.    worker.start();
114.
115. }
```

Figure 9-39 ForEach service implementation 2

This function is executed as soon as the services have been identified. This enables the initialisation of these services and enables read/write actions to be traversed to the sensor through the Main Activity.

All these values are then submitted to the fragments of the application, the original fragment is the one that will enable the user to modify the speed and the wake on shake function. Shown below is the registering of the fragment to the adapter that displays the fragments that require to receive those broadcasts and instantiates them when the Bluetooth sensor services have been found.

```
01. public static MotionFragment newInstance(int position) {
02.      Bundle args = new Bundle();
03.      MotionFragment fragment = new MotionFragment();
04.      args.putInt(FRAGMENT_POSITION, position);
05.      fragment.setArguments(args);
06.
07.      return fragment;
08.  }
```

Figure 9-40 Adapter Subscription

```
SeekBar.OnSeekBarChangeListener onSeekBarChangeListener = new SeekBar.OnSeekBarChangeListener() {
        @Override
        public void onProgressChanged(SeekBar seekBar, int progress, boolean fromUser) {
            periodLength.setText("Sensor period (currently : " + ((progress * 10) + periodMinVal) + "ms)");
        }

        @Override
        public void onStartTrackingTouch(SeekBar seekBar) {

        }

        @Override
        public void onStopTrackingTouch(SeekBar seekBar) {
            //Log.d(TAG, "Period Stop");
            int period = periodMinVal + (seekBar.getProgress() * 10);

            if (period > 2450) period = 2450;
            if (period < 100) period = 100;
            byte p = (byte)((period / 10) + 10);

            //Log.d(TAG, "Period characteristic set to: " + period);
            mBleService.changePeriod(mBleService.getCharacteristicFromUUID(SensorTagGatt.UUID_MOV_PERI.toString()), p);
        }
    };

    /**
     * Handles switch clicks which enable/disable service or turns on the wake on shake feature.
     */
    CompoundButton.OnCheckedChangeListener onCheckedChangeListener = new CompoundButton.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView, boolean isChecked) {

            if (!isChecked) {

                switch (buttonView.getId()) {
                    case R.id.wakeOnShakeSwitch:
                        mBleService.enableMotionService(mThis, SensorTagGatt.UUID_MOV_CONF, false);
                        break;

                    default:
                        break;
                }

            } else {
                switch (buttonView.getId()) {
                    case R.id.wakeOnShakeSwitch:
                        mBleService.enableMotionService(mThis, SensorTagGatt.UUID_MOV_CONF, true);
                        break;

                    default:
                        break;

                }
            }
```

Figure 9-41 - Speed Seek bar and WakeOnShake

These are the functions utilised to alter those values on the sensor which are directly linking to the BLE service.

```
private final BroadcastReceiver motionUpdateReceiver = new BroadcastReceiver() {
        @Override
        public void onReceive(Context context, Intent intent) {

            final String action = intent.getAction();

            if (IntentNames.ACTION_MOV_CHANGE.equals(action)) {

                //Log.i(TAG, "***************** MOTION sensed *****************");
                byte[] value = intent.getByteArrayExtra(IntentNames.EXTRAS_MOV_DATA);
                Point3D v;

                v = SensorConversion.MOVEMENT_ACC.convert(value);
                accelData.setText(Html.fromHtml(String.format("<font color=#FF0000>X:%.2fG</font>," +
                        "<font color=#00967D>Y:%.2fG</font>, <font color=#00000>Z:%.2fG</font>", v.x,v.y,v.z)));


                v = SensorConversion.MOVEMENT_GYRO.convert(value);
                gyroData.setText(Html.fromHtml(String.format("<font color=#FF0000>X:%.2f°/s</font>, " +
                        "<font color=#00967D>Y:%.2f°/s</font>, <font color=#00000>Z:%.2f°/s</font>", v.x, v.y, v.z)));

                v = SensorConversion.MOVEMENT_MAG.convert(value);
                magData.setText(Html.fromHtml(String.format("<font color=#FF0000>X:%.2fuT</font>, " +
                        "<font color=#00967D>Y:%.2fuT</font>, <font color=#00000>Z:%.2fuT</font>", v.x, v.y, v.z)));
            }

        }
    };
```

Figure 9-42 Broadcast Receiver from Activity

The data are manipulated accordingly once the broadcast has been received and then are outputted on the users' mobile device screen as (Accelerometer x, y, z), (Gyroscope x, y, z), (Magnetometer x, y, z).

9.4 Algorithms

9.4.1 Madgwicks

In order to utilise the Madgwicks' algorithm, there is a need to convert the raw data we are receiving into quaternions and then by utilising the data into converting those quaternions into Euler Angles which are then usable data for the application to display and enable us to detect the angles of the sensor accordingly. Shown in Figure 9 - 44 is the fragment that implements this function.

In order to get the appropriate data, we require to pass our current sensor raw data to the algorithm and would also require modification in order to retrieve angles.

Shown below in Figure 9- 43 is the data passed to the algorithm and the algorithm class being initialised.

```
01.  private Madgwick_c madgwick_c = new Madgwick_c(250f, 0.1f);
```

Figure 9-43 Madgwicks Algorithm Initialisation

The two values are the beta value required by the algorithm and the sampling rate that is going to pass the algorithm at.

```
01.  private void Madgwick3() {
02.
03.
04.      madgwick_c.MadgwickAHRSupdate((float) gyro[0], (float) gyro[1], (float) gyro[2],
05.      (float) accel[0], (float) accel[1], (float) accel[2],
06.      (float) magnet[0], (float) magnet[1], (float) magnet[2]);
07.
08.      if (seriesAccx.size() > HISTORY_SIZE) {
09.          seriesAccx.removeFirst();
10.          seriesAccy.removeFirst();
11.          seriesAccz.removeFirst();
12.      }
13.
14.      lpPitch = madgwick_c.MadgPitch;
15.      lpRoll = madgwick_c.MadgRoll;
16.      lpYaw = madgwick_c.MadgYaw;
17.
18.      seriesAccx.addLast(null, lpPitch);
19.      seriesAccy.addLast(null, lpRoll);
20.      seriesAccz.addLast(null, lpYaw);
21.
22.
23.      plot.post(new Runnable() {
24.          public void run() {
25.
26.
27.              labelaccx.setText(Double.toString( madgwick_c.MadgPitch));
28.              labelaccy.setText(Double.toString(madgwick_c.MadgRoll));
29.              labelaccz.setText(Double.toString(madgwick_c.MadgYaw));
30.              labeldt.setText(Double.toString(madgwick_c.sampleFreq));
31.
32.              plot.redraw();
33.          }
34.      });
35.
36.  }
```

Figure 9-44 Passing data to Algorithm

This function is utilised in order to pass data to the algorithm and to plot the data being returned into a graph.

```
qDot1 = 0.5f * (-q1 * gx - q2 * gy - q3 * gz);
qDot2 = 0.5f * (q0 * gx + q2 * gz - q3 * gy);
qDot3 = 0.5f * (q0 * gy - q1 * gz + q3 * gx);
qDot4 = 0.5f * (q0 * gz + q1 * gy - q2 * gx);
```

Figure 9-45 Rate of change of Gyroscope

The function is followed after the normalisation of the magnetometer and the accelerometer which can then be fed to the function to determine the reference of the earth based on the magnetic field as shown below.

```
hx = mx * q0q0 - _2q0my * q3 + _2q0mz * q2 + mx * q1q1 + _2q1 * my * q2 + _2q1 * mz * q3 - mx * q2q2 - mx * q3q3;
        hy = _2q0mx * q3 + my * q0q0 - _2q0mz * q1 + _2q1mx * q2 - my * q1q1 + my * q2q2 + _2q2 * mz * q3 - my * q3q3;
        _2bx = (float) sqrt(hx * hx + hy * hy);
        _2bz = -_2q0mx * q2 + _2q0my * q1 + mz * q0q0 + _2q1mx * q3 - mz * q1q1 + _2q2 * my * q3 - mz * q2q2 + mz * q3q3;
        _4bx = 2.0f * _2bx;
        _8bx = 2.0f * _4bx;
        _4bz = 2.0f * _2bz;
        _8bz = 2.0f * _4bz;
```

Figure 9-46 Reference direction of Earth's magnetic field

```
    // Gradient decent algorithm corrective step
s0= (float) (-_2q2*(2*(q1q3 - q0q2) - ax)    +   _2q1*(2*(q0q1 + q2q3) - ay)
+  -_4bz*q2*(_4bx*(0.5 - q2q2 - q3q3) + _4bz*(q1q3 - q0q2) - mx)
+   (-_4bx*q3+_4bz*q1)*(_4bx*(q1q2 - q0q3) + _4bz*(q0q1 + q2q3) - my)
+   _4bx*q2*(_4bx*(q0q2 + q1q3) + _4bz*(0.5 - q1q1 - q2q2) - mz));

s1= (float) (_2q3*(2*(q1q3 - q0q2) - ax) +   _2q0*(2*(q0q1 + q2q3) - ay)
+   -4*q1*(2*(0.5 - q1q1 - q2q2) - az)    +   _4bz*q3*(_4bx*(0.5 - q2q2 - q3q3)
+ _4bz*(q1q3 - q0q2) - mx)   + (_4bx*q2+_4bz*q0)*(_4bx*(q1q2 - q0q3)
+ _4bz*(q0q1 + q2q3) - my)   +   (_4bx*q3-_8bz*q1)*(_4bx*(q0q2 + q1q3)
+ _4bz*(0.5 - q1q1 - q2q2) - mz));

s2= (float) (-_2q0*(2*(q1q3 - q0q2) - ax)    +     _2q3*(2*(q0q1 + q2q3) - ay)
+   (-4*q2)*(2*(0.5 - q1q1 - q2q2) - az) +   (-_8bx*q2-_4bz*q0)*(_4bx*(0.5 - q2q2 - q3q3)
+ _4bz*(q1q3 - q0q2) - mx)+(_4bx*q1+_4bz*q3)*(_4bx*(q1q2 - q0q3)
+ _4bz*(q0q1 + q2q3) - my)+(_4bx*q0-_8bz*q2)*(_4bx*(q0q2 + q1q3)
+ _4bz*(0.5 - q1q1 - q2q2) - mz));

s3= (float) (_2q1*(2*(q1q3 - q0q2) - ax) +   _2q2*(2*(q0q1 + q2q3) - ay)
+(-_8bx*q3+_4bz*q1)*(_4bx*(0.5 - q2q2 - q3q3) + _4bz*(q1q3 - q0q2) - mx)
+(-_4bx*q0+_4bz*q2)*(_4bx*(q1q2 - q0q3) + _4bz*(q0q1 + q2q3) - my)
+(_4bx*q1)*(_4bx*(q0q2 + q1q3) + _4bz*(0.5 - q1q1 - q2q2) - mz));
```

Figure 9-47 Gradient descent algorithm

After the data has been passed through the gradient descent algorithm it then requires to be normalised and then passed on to a function to apply the step algorithm results with the beta value that has been fed to the function to output Quaternions with the orientation values.

```
    qDot1 -= beta * s0;
    Log.d(TAG, " qdot1: "+qDot1+" is it the qdot1?");

    qDot2 -= beta * s1;
    qDot3 -= beta * s2;
    qDot4 -= beta * s3;
}

q0 += qDot1 * (1.0f / sampleFreq);
q1 += qDot2 * (1.0f / sampleFreq);
q2 += qDot3 * (1.0f / sampleFreq);
q3 += qDot4 * (1.0f / sampleFreq);
```

Figure 9-48 - Apply Step and Integrate rate of change to Quaternions

Which are then re-normalised and passed on to the algorithm for conversion to Euler's angles.

```
01. private void quat_2_euler()
02. {
03. 
04.    //  roll (x-axis rotation)
05.     double sinr_cosp = +2.0 * (q0* q1 + q2 * q3);
06.     double cosr_cosp = +1.0 - 2.0 * (q1 * q1 + q2 * q2);
07.     MadgRoll =(float)  Math.toDegrees (atan2(sinr_cosp, cosr_cosp));
08. 
09.     // pitch (y-axis rotation)
10.     double sinp = +2.0 * (q0 * q2 - q3 * q1);
11.     if (abs(sinp) >= 1)
12.         MadgPitch = (float)  Math.toDegrees (copySign(PI / 2, sinp)); // use 90 degrees if out of range
13.     else
14.         MadgPitch = (float)  Math.toDegrees (asin(sinp));
15. 
16.     // yaw (z-axis rotation)
17.     double siny_cosp = +2.0 * (q0 * q3 + q1 * q2);
18.     double cosy_cosp = +1.0 - 2.0 * (q2 * q2 + q3 * q3);
19.     MadgYaw = (float)  Math.toDegrees (atan2(siny_cosp, cosy_cosp));
20. 
21. 
22. 
23. 
24. }
```

Figure 9-49 Quaternion to Euler's

An implementation of the inverse square root is used throughout the implementation in order to increase the accuracy of the Quaternions shown.

```
01. private int instability_fix = 1;
02. 
03. 
04. 
05.     private float invSqrt(float number){
06.      if (instability_fix == 0)
07.     {
08. 
09.         int i;
10.         float x2, y;
11.      float threehalfs = 1.5F;
12. 
13.         x2 = number * 0.5F;
14.         i  = 0x5f3759df - (Float.floatToIntBits(number)>> 1);
15.         y  = intBitsToFloat( i);
16.         y  = y * ( threehalfs - ( x2 * y * y ) );   // 1st iteration
17.         y  = y * ( threehalfs - ( x2 * y * y ) );   // 2nd iteration, this can be removed
18. 
19.         return y;
20.     }
21.         else if (instability_fix == 1)
22.         {
23. 
24.            int i = 0x5F1F1412 - (Float.floatToIntBits(number) >> 1);
25.             float tmp = Float.intBitsToFloat( i);
26.             return tmp * (1.69000231f - 0.714158168f * number * tmp * tmp);
27.         }
28.         else
29.         {
30.             return (float) (1.0f / sqrt(number));
31.         }
32.     }
```

Figure 9-50 Inverse Square Root

9.4.2 Mahony

The Mahony algorithm follows a similar procedure as Madgwicks algorithm. It normalises the accelerometer and the magnetometer data, it then detects the reference direction of Earth's magnetic field and then the difference being here is the utilisation of the function that estimates the direction of the gravity and magnetic field by the function shown below.

```
halfvx = q1q3 - q0q2;
halfvy = q0q1 + q2q3;
halfvz = q0q0 - 0.5f + q3q3;
halfwx = bx * (0.5f - q2q2 - q3q3) + bz * (q1q3 - q0q2);
halfwy = bx * (q1q2 - q0q3) + bz * (q0q1 + q2q3);
halfwz = bx * (q0q2 + q1q3) + bz * (0.5f - q1q1 - q2q2);
```

Figure 9-51 Direction of Gravity and Magnetic field

It then identified the error is the sum of the cross-multiplication between estimated direction and measured direction of field vectors.

```
halfex = (ay * halfvz - az * halfvy) + (my * halfwz - mz * halfwy);
halfey = (az * halfvx - ax * halfvz) + (mz * halfwx - mx * halfwz);
halfez = (ax * halfvy - ay * halfvx) + (mx * halfwy - my * halfwx);
```

Figure 9-52 Error of cross of estimated direction with measure direction

```
if(twoKi > 0.0f) {
    integralFBx += twoKi * halfex * (1.0f / sampleFreq);    // integral error scaled by Ki
    integralFBy += twoKi * halfey * (1.0f / sampleFreq);
    integralFBz += twoKi * halfez * (1.0f / sampleFreq);
    gx += integralFBx;  // apply integral feedback
    gy += integralFBy;
    gz += integralFBz;
}
else {
    integralFBx = 0.0f; // prevent integral windup
    integralFBy = 0.0f;
    integralFBz = 0.0f;
}
```

Figure 9-53 Applies integral feedback

It then based on the values outputted in the previous functions it than computes and applies integral feedback.

```
gx += twoKp * halfex;
gy += twoKp * halfey;
gz += twoKp * halfez;
```

Figure 9-54 Apply proportion feedback

It will then apply the proportion of the feedback and then pass the output to be converted into quaternions and then normalise it and pass it to the function to convert the quaternions into Euler Angles.

9.4.3 Kalman Fusion

Kalman fusion is one of the most important algorithms that have been developed in the recent years, and thanks to its success it's being implemented into multiple locations that include aviation and all kinds of orientations and Attitude, Heading Reference System algorithms will most likely implement Kalman filters and similar algorithms in order to have a commercially and robust application. Shown below is the implementation of the Kalman Filter.

All the values required to be initialised are shown below

```
01. public static final int TIME_CONSTANT = 100; //In milliseconds
02. private Timer kalmanTimer = new Timer();
03.
04. private float[] gyro = new float[3];               // angular speeds from gyro
05. private float[] accel = new float[3];              // accelerometer vector
06. private float[] magnet = new float[3];             // magnetic field vector
07. private float[] accMagOrientation = new float[3];  // orientation angles from accel and magnet
08. private float[] rotationMatrix = new float[9];     // accelerometer and magnetometer based rotation matrix
09.
10. private float accel_roll;
11. private float accel_pitch;
12.
13. private float kalman_roll;
14. private float kalman_pitch;
15.
16. private static final float accelStd = 0.01f;   //Standard deviation of accelerometer readings - Used to initialize Pk
17. private static final float gyroStd = 0.01f;    //Standard deviation of gyro readings
18.
19. private static final float Q_accel = 0.001f;    //process noise variance for accelerometer - Used to set Q
20. private static final float Q_gyro = 0.003f;     //process noise variance for gyro
21.
22. private static final float R_accel = 0.03f;    //measurement noise variance for accelerometer - Used to set R
23. private static final float R_gyro = 0.03f;     //measurement noise variance for gyro
24.
25. private float[] A = {1, TIME_CONSTANT/1000f, 0, 1};
26.
27. private float[] wk = {Q_accel, Q_gyro};     //process noise
28. private float[] vk = {R_accel, R_gyro};     //measurement noise
29.
30. //Pitch variables
31. private float[] Xk_pitch = {0, 0};        //angle and angular speed
32.
33. private float[] Zk_pitch = new float[2];      //measured values
34.
35. private float[] Pk_pitch = {accelStd*accelStd, accelStd*gyroStd,
36.         accelStd*gyroStd,  gyroStd*gyroStd}; //process error covariance
37.
38. private float[] R_pitch = {vk[0]*vk[0], vk[0]*vk[1],
39.         vk[1]*vk[0], vk[1]*vk[1]};      //measurement error covariance
40.
41. private float[] Q_pitch = {wk[0]*wk[0], wk[0]*wk[1],
42.         wk[1]*wk[0], wk[1]*wk[1]};      //process noise covariance
43.
44. private float[] K_pitch = new float[4];      //Kalman gain
45.
46. //Roll variables
47. private float[] Xk_roll = {0, 0};        //angle and angular speed
48.
49. private float[] Zk_roll = new float[2];      //measured values
50.
51. private float[] Pk_roll = {accelStd*accelStd, accelStd*gyroStd,
52.         accelStd*gyroStd,  gyroStd*gyroStd}; //process error covariance
53.
54. private float[] R_roll = {vk[0]*vk[0], vk[0]*vk[1],
55.         vk[1]*vk[0], vk[1]*vk[1]};      //measurement error covariance
56.
57. private float[] Q_roll = {wk[0]*wk[0], wk[0]*wk[1],
58.         wk[1]*wk[0], wk[1]*wk[1]};      //process noise covariance
59.
60. private float[] K_roll = new float[4];      //Kalman gain
61.
62. private float[] I = {1,0,0,1};
```

Figure 9-55 Initialisation of variables

As soon as the broadcast values have been received of all the various data, they are then implemented within their variables

```
01. calculateAccMagOrientation();
02.                 accel_roll = (float) Math.atan2(accel[0], Math.sqrt(Math.pow(accel[1], 2) + Math.pow(accel[2], 2)));
03.                 //Log.d("Roll: ", Float.toString(accel_roll));
04.                 accel_pitch = (float) Math.atan2(accel[1], Math.sqrt(Math.pow(accel[0], 2) + Math.pow(accel[2], 2)));
```

Figure 9-56 assigning roll and pitch

The requirements are based on the white paper which has been released by MIT [26]. There is a requirement to take the previous state and include it within the function in order to properly predict the values that will be displayed next. The following image shows the mathematical functions required in order to operate the Kalman filter ranging from multiplication, division, addition subtraction and transposing all for the data being outputted through the algorithms of google as well as the raw data of the sensor.

```
01.  public float[] dot(float[] A, float[] B){
02.         if (A.length == 4 && B.length == 2){
03.             float[] C = new float[2];
04.
05.             C[0] = A[0]*B[0] + A[1]*B[1];
06.             C[1] = A[2]*B[0] + A[3]*B[1];
07.
08.             return C;
09.         }
10.         else if (A.length == 4 && B.length == 4){
11.             float[] C = new float [4];
12.
13.             C[0] = A[0]*B[0] + A[1]*B[2];
14.             C[1] = A[0]*B[1] + A[1]*B[3];
15.             C[2] = A[2]*B[0] + A[3]*B[2];
16.             C[3] = A[2]*B[1] + A[3]*B[3];
17.
18.             return C;
19.         }
20.         return null;
21.     }
22.
23.     public float[] transpose(float[] A){
24.         if (A.length == 4){
25.             float[] C = new float[4];
26.
27.             C[0] = A[0];
28.             C[1] = A[2];
29.             C[2] = A[1];
30.             C[3] = A[3];
31.
32.             return C;
33.         }
34.         return null;
35.     }
36.
37.
38.     public float[] divide(float[] A, float[] B){
39.         if (A.length == 4 && B.length == 4){
40.             float[] C = new float [4];
41.
42.             C[0] = A[0] / B[0];
43.             C[1] = A[1] / B[1];
44.             C[2] = A[2] / B[2];
45.             C[3] = A[3] / B[3];
46.
47.             return C;
48.         }
49.         return null;
50.     }
```

```
01.  public float[] add(float[] A, float[] B){
02.      if (A.length == 2 && B.length == 2){
03.          float[] C = new float[2];
04.
05.          C[0] = A[0]+B[0];
06.          C[1] = A[1]+B[1];
07.
08.          return C;
09.      }
10.      else if (A.length == 4 && B.length == 4){
11.          float[] C = new float [4];
12.
13.          C[0] = A[0]+B[0];
14.          C[1] = A[1]+B[1];
15.          C[2] = A[2]+B[2];
16.          C[3] = A[3]+B[3];
17.
18.          return C;
19.      }
20.      return null;
21.  }
22.
23.  public float[] subtract(float[] A, float[] B){
24.      if (A.length == 2 && B.length == 2){
25.          float[] C = new float[2];
26.
27.          C[0] = A[0]-B[0];
28.          C[1] = A[1]-B[1];
29.
30.          return C;
31.      }
32.      else if (A.length == 4 && B.length == 4){
33.          float[] C = new float [4];
34.
35.          C[0] = A[0]-B[0];
36.          C[1] = A[1]-B[1];
37.          C[2] = A[2]-B[2];
38.          C[3] = A[3]-B[3];
39.
40.          return C;
41.      }
42.      return null;
43.  }
```

Figure 9-57 Mathematical functions

As well as running the Accelerometer/Magnetometer Orientation function which follows the same procedure as the previous algorithm with the Google functions in order to retrieve the matrixes.

The function to calculate Kalman is displayed below.

```
class calculateKalman extends TimerTask {
      public void run() {

          Xk_pitch = add(dot(A, Xk_pitch), wk);

          Pk_pitch = dot(dot(A, Pk_pitch), transpose(A));
          Pk_pitch = add(Pk_pitch, Q_pitch);

          K_pitch = divide(Pk_pitch, add(Pk_pitch, R_pitch));

          Zk_pitch[0] = 1.0f * accel_pitch;
          Zk_pitch[1] = gyro[0];

          Xk_pitch = add(Xk_pitch, dot(K_pitch, subtract(Zk_pitch, Xk_pitch)));

          Pk_pitch = dot(subtract(I, K_pitch), Pk_pitch);

          kalman_pitch = Xk_pitch[0];


          Xk_roll = add(dot(A, Xk_roll), wk);

          Pk_roll = dot(dot(A, Pk_roll), transpose(A));
          Pk_roll = add(Pk_roll, Q_roll);

          K_roll = divide(Pk_roll, add(Pk_roll, R_roll));

          Zk_roll[0] = -1.0f * accel_roll;
          Zk_roll[1] = gyro[1];

          Xk_roll = add(Xk_roll, dot(K_roll, subtract(Zk_roll, Xk_roll)));

          Pk_roll = dot(subtract(I, K_roll), Pk_roll);

          kalman_roll = Xk_roll[0];
      }
  }
```

Figure 9-58 - Kalman Fusion

XK_pitch indicates angular and angle of the sensor which adds the values of the matrix with time it's previous XK pitch values and then the processes of the noise of accelerometer and gyroscope.

PK_pitch multiples the matrix with the time variance along with itself and then multiplied the result with the transposed version of the matrix. Which will then be added along with the value identified and the processes noise covariance.

K_pitch will than divide PK_Pitch with the addition of itself and the measurement error covariance.

ZK_pitchfirst are the values that had been measured by the algorithm the first value of the array being the accelerometer pitch and second being the gyro.x.

XK_pitch will then be utilised again in order to add itself along with a multiplication of K_pitch with the subtraction of ZK_Pitch and XK_Pitch

Which will then link to PK_Pitch which will multiply the subtraction of I with K_Pitch with PK_Pitch and at this point the first value of the array XK_Pitch is the value of the Kalman Pitch. The functions are implemented in order to successfully predict the state of the sensor in the angle of pitch. It utilises this algorithm in order to implement angles and determine orientation based on the data being provided, what is the next state of the movement of the sensor. The gain is also implemented to minimise the mean square error. By utilising the matrix in such a way to predict the next state enables us to have a minimal error algorithm that can be utilised in detection of angles seamlessly.

9.4.4 Google Fusion algorithms

9.4.4.1 Accelerometer/Magnetometer Orientation

The following function is utilised in order to calculate Accelerometer and Magnetometer orientation.

```
01. public void calculateAccMagOrientation() {
02.
03.     if(SensorManager.getRotationMatrix(rotationMatrix, null, accel, magnet)) {
04.         SensorManager.getOrientation(rotationMatrix, accMagOrientation);
05.     }
06. }
```

Figure 9-59 - Rotation Matrix and Orientation

These functions that have been made available by the SensorManager in Google, retrieve Rotation Matrixes and Orientation based on the data of the accelerometer and the magnetometer which output a rotation matrix which is passed onto the getOrientation function along with the Accelerometer and Magnetometer Orientation.

9.4.4.2 Gyroscope Orientation

The following function enables us to retrieve gyroscope Orientation. Which is then outputted as a secondary orientation function within our algorithm in order to distinguish the difference between how each fusion operates better.

```
01.  public void gyroFunction(float[] gyrovalues, Long eventtimestamp) {
02.
03.          if (accMagOrientation == null)
04.              return;
05.
06.          // initialisation of the gyroscope based rotation matrix
07.          if(initState) {
08.              float[] initMatrix = new float[9];
09.              initMatrix = getRotationMatrixFromOrientation(accMagOrientation);
10.              float[] test = new float[3];
11.              SensorManager.getOrientation(initMatrix, test);
12.              gyroMatrix = matrixMultiplication(gyroMatrix, initMatrix);
13.              initState = false;
14.          }
15.
16.          // copy the new gyro values into the gyro array
17.          // convert the raw gyro data into a rotation vector
18.          float[] deltaVector = new float[4];
19.          if(timestamp != 0) {
20.              final float dT = (eventtimestamp - timestamp) * NS2S;
21.              System.arraycopy(gyrovalues, 0, gyro, 0, 3);
22.              getRotationVectorFromGyro(gyro, deltaVector, dT / 2.0f);
23.          }
24.
25.          timestamp = eventtimestamp;
26.
27.          // convert rotation vector into rotation matrix
28.          float[] deltaMatrix = new float[9];
29.          SensorManager.getRotationMatrixFromVector(deltaMatrix, deltaVector);
30.
31.          // apply the new rotation interval on the gyroscope based rotation matrix
32.          gyroMatrix = matrixMultiplication(gyroMatrix, deltaMatrix);
33.
34.          // get the gyroscope based orientation from the rotation matrix
35.          SensorManager.getOrientation(gyroMatrix, gyroOrientation);
36.      }
```

Figure 9-60 Gyro Function

The above functions are further elaborated in the following figures:

```
01.  private float[] getRotationMatrixFromOrientation(float[] o) {
02.      float[] xM = new float[9];
03.      float[] yM = new float[9];
04.      float[] zM = new float[9];
05.
06.      float sinX = (float)Math.sin(o[1]);
07.      float cosX = (float)Math.cos(o[1]);
08.      float sinY = (float)Math.sin(o[2]);
09.      float cosY = (float)Math.cos(o[2]);
10.      float sinZ = (float)Math.sin(o[0]);
11.      float cosZ = (float)Math.cos(o[0]);
12.
13.      // rotation about x-axis (pitch)
14.      xM[0] = 1.0f; xM[1] = 0.0f; xM[2] = 0.0f;
15.      xM[3] = 0.0f; xM[4] = cosX; xM[5] = sinX;
16.      xM[6] = 0.0f; xM[7] = -sinX; xM[8] = cosX;
17.
18.      // rotation about y-axis (roll)
19.      yM[0] = cosY; yM[1] = 0.0f; yM[2] = sinY;
20.      yM[3] = 0.0f; yM[4] = 1.0f; yM[5] = 0.0f;
21.      yM[6] = -sinY; yM[7] = 0.0f; yM[8] = cosY;
22.
23.      // rotation about z-axis (yaw)
24.      zM[0] = cosZ; zM[1] = sinZ; zM[2] = 0.0f;
25.      zM[3] = -sinZ; zM[4] = cosZ; zM[5] = 0.0f;
26.      zM[6] = 0.0f; zM[7] = 0.0f; zM[8] = 1.0f;
27.
28.      // rotation order is y, x, z (roll, pitch, yaw)
29.      float[] resultMatrix = matrixMultiplication(xM, yM);
30.      resultMatrix = matrixMultiplication(zM, resultMatrix);
31.      return resultMatrix;
32.  }
```

Figure 9-61 Orientation to Matrix conversion

At the initialisation stage, it requires the Accelerometer and Magnetometer data retrieved from the Acc/Mag orientation function which is then converted back to matrixes and used as initialisation values so that there are no null values in the function. The values in the second iteration are then replaced by the gyro data after initialisation.

```
01.  private void getRotationVectorFromGyro(float[] gyroValues,
02.                                         float[] deltaRotationVector,
03.                                         float timeFactor)
04.    {
05.        float[] normValues = new float[3];
06.
07.        // Calculate the angular speed of the sample
08.        float omegaMagnitude =
09.                (float)Math.sqrt(gyroValues[0] * gyroValues[0] +
10.                        gyroValues[1] * gyroValues[1] +
11.                        gyroValues[2] * gyroValues[2]);
12.
13.        // Normalize the rotation vector if it's big enough to get the axis
14.        if(omegaMagnitude > EPSILON) {
15.            normValues[0] = gyroValues[0] / omegaMagnitude;
16.            normValues[1] = gyroValues[1] / omegaMagnitude;
17.            normValues[2] = gyroValues[2] / omegaMagnitude;
18.        }
19.
20.        float thetaOverTwo = omegaMagnitude * timeFactor;
21.        float sinThetaOverTwo = (float)Math.sin(thetaOverTwo);
22.        float cosThetaOverTwo = (float)Math.cos(thetaOverTwo);
23.        deltaRotationVector[0] = sinThetaOverTwo * normValues[0];
24.        deltaRotationVector[1] = sinThetaOverTwo * normValues[1];
25.        deltaRotationVector[2] = sinThetaOverTwo * normValues[2];
26.        deltaRotationVector[3] = cosThetaOverTwo;
27.    }
```

Figure 9-62 – Vector from Gyroscope values

This function is also based on Googles' Documentation [27] that indicates us how to retrieve rotation vectors from the gyroscope raw data. It enables us to determine the rotation of the device since the values from its previous measurement. This function is called within the overall Gyro Function which is shown above although the data being passed are not in the appropriate format, in order to manipulate these values accordingly we needed to convert the vector into matrixes and then we would be able to carry out matrix multiplication which would make them suitable. Therefore, a Google function that translates from Vector to matrixes was implemented to translate those rotations into matrixes for them to be applied in our function.

```
01.  private float[] matrixMultiplication(float[] A, float[] B) {
02.          float[] result = new float[9];
03.
04.          result[0] = A[0] * B[0] + A[1] * B[3] + A[2] * B[6];
05.          result[1] = A[0] * B[1] + A[1] * B[4] + A[2] * B[7];
06.          result[2] = A[0] * B[2] + A[1] * B[5] + A[2] * B[8];
07.
08.          result[3] = A[3] * B[0] + A[4] * B[3] + A[5] * B[6];
09.          result[4] = A[3] * B[1] + A[4] * B[4] + A[5] * B[7];
10.          result[5] = A[3] * B[2] + A[4] * B[5] + A[5] * B[8];
11.
12.          result[6] = A[6] * B[0] + A[7] * B[3] + A[8] * B[6];
13.          result[7] = A[6] * B[1] + A[7] * B[4] + A[8] * B[7];
14.          result[8] = A[6] * B[2] + A[7] * B[5] + A[8] * B[8];
15.
16.          return result;
17.      }
```

Figure 9-63 Matrix multiplication

It also establishes the currently new rotation (deltamatrix) onto the gyroscope-based rotation matrix (gyromatrix) and then retrieves the orientation of the gyro.

9.4.5 Elevation algorithm

9.4.5.1 Stationary Detection algorithm

```
    if (((((Math.round(f1currentroll) == 0)|| (Math.round(f1currentroll) == 1))
|| ((Math.round(f1meanroll) == 0)|| (Math.round(f1meanroll) == 1)))
    || (((Math.round(f2currentroll) == 0)|| (Math.round(f2currentroll) == 1))
|| ((Math.round(f2meanroll) == 0)|| (Math.round(f2meanroll) == 1)))
    || (((Math.round(f3currentroll) == 0)|| (Math.round(f3currentroll) == 1))
|| ((Math.round(f3meanroll) == 0)|| (Math.round(f3meanroll) == 1))))
        //negative values
    || ((( Math.round(f1currentroll) == -1) || (Math.round(f1meanroll) == -1))
    || ( ( Math.round(f2currentroll) == -1) || (Math.round(f2meanroll) == -1))
    || ( ( Math.round(f3currentroll) == -1) || (Math.round(f3meanroll) == -1))))
    {
        if ((((f1currentroll < f1meanroll + 3) && (f1currentroll > f1meanroll - 3))
|| ((f1currentroll > f1meanroll + 3) && (f1currentroll < f1meanroll - 3)))
        || (((f2currentroll < f2meanroll + 3) && (f2currentroll > f2meanroll - 3))
|| ((f2currentroll > f2meanroll + 3) && (f2currentroll < f2meanroll - 3)))
        || (((f3currentroll < f3meanroll + 3) && (f3currentroll > f3meanroll - 3))
|| ((f3currentroll > f3meanroll + 3) && (f3currentroll < f3meanroll - 3)))){

                Log.d(TAG, "the roll is stationary now ");
                rollstationary=true;
            }else
            {
                Log.d(TAG, "the roll is not stationary yet moves to second class ");

            }
        }else
    {
        if    ((((f1currentroll < f1meanroll * 1.03) && (f1currentroll > f1meanroll * 0.97))
|| ((f1currentroll > f1meanroll * 1.03) && (f1currentroll < f1meanroll * 0.97)))
            || (((f2currentroll < f2meanroll * 1.03) && (f2currentroll > f2meanroll * 0.97))
||((f2currentroll > f2meanroll * 1.03) && (f2currentroll < f2meanroll * 0.97)))
            || (((f3currentroll < f3meanroll * 1.03) && (f3currentroll > f3meanroll * 0.97))
||((f3currentroll > f3meanroll * 1.03) && (f3currentroll < f3meanroll * 0.97))))
        {
            Log.d(TAG, "the roll is stationary now ");
            rollstationary=true;
        }else
        {
            Log.d(TAG, "the roll is not stationary failed ");

        }

        }
```

Figure 9-64 Roll Fusion Stationary Algorithm

```
104.
105.     if (((((Math.round(f1currentyaw) == 0)|| (Math.round(f1currentyaw) == 1))
106. || ((Math.round(f1meanyaw) == 0)|| (Math.round(f1meanyaw) == 1)))
107.     || (((Math.round(f2currentyaw) == 0)|| (Math.round(f2currentyaw) == 1))
108. || ((Math.round(f2meanyaw) == 0)|| (Math.round(f2meanyaw) == 1)))
109.     || (((Math.round(f3currentyaw) == 0)|| (Math.round(f3currentyaw) == 1))
110. || ((Math.round(f3meanyaw) == 0)|| (Math.round(f3meanyaw) == 1))))
111.             //negative values
112.     || ((( Math.round(f1currentyaw) == -1) || (Math.round(f1meanyaw) == -1))
113.     || ( ( Math.round(f2currentyaw) == -1) || (Math.round(f2meanyaw) == -1))
114.     || ( ( ( Math.round(f3currentyaw) == -1) || (Math.round(f3meanyaw) == -1))))
115.     {
116.
117.
118.      if ((((f1currentyaw < f1meanyaw + 3) && (f1currentyaw > f1meanyaw - 3))
119. || ((f1currentyaw > f1meanyaw + 3) && (f1currentyaw < f1meanyaw - 3)))
120.     || (((f2currentyaw < f2meanyaw + 3) && (f2currentyaw > f2meanyaw - 3))
121. || ((f2currentyaw > f2meanyaw + 3) && (f2currentyaw < f2meanyaw - 3)))
122.     || (((f3currentyaw < f3meanyaw + 3) && (f3currentyaw > f3meanyaw - 3))
123. || ((f3currentyaw > f3meanyaw + 3) && (f3currentyaw < f3meanyaw - 3)))){
124.
125.          Log.d(TAG, "the yaw is stationary now ");
126.          yawstationary=true;
127.      }else
128.      {
129.          Log.d(TAG, "the yaw is not stationary yet moves to second class ");
130.
131.      }
132.     }else
133.
134.
135.     {
136.         if ((((f1currentyaw < f1meanyaw * 1.03) && (f1currentyaw > f1meanyaw * 0.97))
137. || ((f1currentyaw > f1meanyaw * 1.03) && (f1currentyaw < f1meanyaw * 0.97)))
138.             ||(((f2currentyaw < f2meanyaw * 1.03) && (f2currentyaw > f2meanyaw * 0.97))
139. ||((f2currentyaw > f2meanyaw * 1.03) && (f2currentyaw < f2meanyaw * 0.97)))
140.             ||(((f3currentyaw < f3meanyaw * 1.03) && (f3currentyaw > f3meanyaw * 0.97))
141. ||((f3currentyaw > f3meanyaw * 1.03) && (f3currentyaw < f3meanyaw * 0.97))))
142.         {
143.             Log.d(TAG, "the yaw is stationary now ");
144.             yawstationary=true;
145.         }else
146.         {
147.             Log.d(TAG, "the yaw is not stationary failed");
148.
149.         }
150.     }
151.
152.
153.     if (rollstationary && pitchstationary && yawstationary)
154.     {
155.         Log.d(TAG, "finally stationary ");
156.         stationary=true;
157.         Log.d(TAG, "stationary: "+(stationary));
158.     }
159.     else {
160.         stationary = false;
161.         Log.d(TAG, "stationary: "+(stationary));
162.
163.
164.
165.
166.
167.     }
168.
169. }
```

Figure 9-65 Yaw Fusion Stationary Algorithm

9.4.5.2 The initialisation of angle functions and retrieving raw data.

```
int howmanytogetforcheckingup = 8;


      int up = 0;
      int listsize = f1pitchvalues.size();

      if (listsize > howmanytogetforcheckingup) {
          int startfrom = listsize - howmanytogetforcheckingup;

          for (int y = startfrom; y < listsize; y++) {
              if (y + 7 < listsize) {
                  float f1pitch1 = f1pitchvalues.get(y);
                  float f1pitch2 = f1pitchvalues.get(y + 1);
                  float f1pitch3 = f1pitchvalues.get(y + 2);
                  float f1pitch4 = f1pitchvalues.get(y + 3);
                  float f1pitch5 = f1pitchvalues.get(y + 4);
                  float f1pitch6 = f1pitchvalues.get(y + 5);
                  float f1pitch7 = f1pitchvalues.get(y + 6);
                  float f1pitch8 = f1pitchvalues.get(y + 7);

                  float f1roll1 = f1rollvalues.get(y);
                  float f1roll2 = f1rollvalues.get(y + 1);
                  float f1roll3 = f1rollvalues.get(y + 2);
                  float f1roll4 = f1rollvalues.get(y + 3);
                  float f1roll5 = f1rollvalues.get(y + 4);
                  float f1roll6 = f1rollvalues.get(y + 5);
                  float f1roll7 = f1rollvalues.get(y + 6);
                  float f1roll8 = f1rollvalues.get(y + 7);


                  float f1yaw1 = f1yawvalues.get(y);
                  float f1yaw2 = f1yawvalues.get(y + 1);
                  float f1yaw3 = f1yawvalues.get(y + 2);
                  float f1yaw4 = f1yawvalues.get(y + 3);
                  float f1yaw5 = f1yawvalues.get(y + 4);
                  float f1yaw6 = f1yawvalues.get(y + 5);
                  float f1yaw7 = f1yawvalues.get(y + 6);
                  float f1yaw8 = f1yawvalues.get(y + 7);

                  float f2pitch1 = f2pitchvalues.get(y);
                  float f2pitch2 = f2pitchvalues.get(y + 1);
                  float f2pitch3 = f2pitchvalues.get(y + 2);
                  float f2pitch4 = f2pitchvalues.get(y + 3);
                  float f2pitch5 = f2pitchvalues.get(y + 4);
                  float f2pitch6 = f2pitchvalues.get(y + 5);
                  float f2pitch7 = f2pitchvalues.get(y + 6);
                  float f2pitch8 = f2pitchvalues.get(y + 7);

                  float f2roll1 = f2rollvalues.get(y);
                  float f2roll2 = f2rollvalues.get(y + 1);
                  float f2roll3 = f2rollvalues.get(y + 2);
                  float f2roll4 = f2rollvalues.get(y + 3);
                  float f2roll5 = f2rollvalues.get(y + 4);
                  float f2roll6 = f2rollvalues.get(y + 5);
                  float f2roll7 = f2rollvalues.get(y + 6);
                  float f2roll8 = f2rollvalues.get(y + 7);

                  float f2yaw1 = f2yawvalues.get(y);
                  float f2yaw2 = f2yawvalues.get(y + 1);
                  float f2yaw3 = f2yawvalues.get(y + 2);
                  float f2yaw4 = f2yawvalues.get(y + 3);
                  float f2yaw5 = f2yawvalues.get(y + 4);
                  float f2yaw6 = f2yawvalues.get(y + 5);
                  float f2yaw7 = f2yawvalues.get(y + 6);
                  float f2yaw8 = f2yawvalues.get(y + 7);

                  float f3pitch1 = f3pitchvalues.get(y);
                  float f3pitch2 = f3pitchvalues.get(y + 1);
                  float f3pitch3 = f3pitchvalues.get(y + 2);
                  float f3pitch4 = f3pitchvalues.get(y + 3);
                  float f3pitch5 = f3pitchvalues.get(y + 4);
                  float f3pitch6 = f3pitchvalues.get(y + 5);
                  float f3pitch7 = f3pitchvalues.get(y + 6);
                  float f3pitch8 = f3pitchvalues.get(y + 7);

                  float f3roll1 = f3rollvalues.get(y);
                  float f3roll2 = f3rollvalues.get(y + 1);
                  float f3roll3 = f3rollvalues.get(y + 2);
                  float f3roll4 = f3rollvalues.get(y + 3);
                  float f3roll5 = f3rollvalues.get(y + 4);
                  float f3roll6 = f3rollvalues.get(y + 5);
                  float f3roll7 = f3rollvalues.get(y + 6);
                  float f3roll8 = f3rollvalues.get(y + 7);

                  float f3yaw1 = f3yawvalues.get(y);
                  float f3yaw2 = f3yawvalues.get(y + 1);
                  float f3yaw3 = f3yawvalues.get(y + 2);
                  float f3yaw4 = f3yawvalues.get(y + 3);
                  float f3yaw5 = f3yawvalues.get(y + 4);
                  float f3yaw6 = f3yawvalues.get(y + 5);
                  float f3yaw7 = f3yawvalues.get(y + 6);
                  float f3yaw8 = f3yawvalues.get(y + 7);
```

Figure 9-66 initialisation values

The initialisation of all the various raw data has been implemented in the above function which utilises an iterative function in order to store the latest 7 values received by the sensor.

```
boolean f1pitchuptodown = (f1pitch1 > f1pitch2) && (f1pitch2 > f1pitch3) && (f1pitch3 > f1pitch4) && (f1pitch4 > f1pitch5) &&
                          (f1pitch5 > f1pitch6) && (f1pitch6 > f1pitch7) && (f1pitch7 > f1pitch8)
                          && (Math.abs(f1pitch8 - f1pitch1) > (f1pitch8 / 10) || (Math.abs(f1pitch1 - f1pitch8) > (f1pitch1 / 10)));
                  boolean f1pitchdowntoup = (f1pitch1 < f1pitch2) && (f1pitch2 < f1pitch3) && (f1pitch3 < f1pitch4) && (f1pitch4 < f1pitch5) &&
                          (f1pitch5 < f1pitch6) && (f1pitch6 < f1pitch7) && (f1pitch7 < f1pitch8)
                          && (Math.abs(f1pitch8 - f1pitch1) > (f1pitch8 / 10) || (Math.abs(f1pitch1 - f1pitch8) > (f1pitch1 / 10)));

                  boolean f2pitchuptodown = (f2pitch1 > f2pitch2) && (f2pitch2 > f2pitch3) && (f2pitch3 > f2pitch4) && (f2pitch4 > f2pitch5) &&
                          (f2pitch5 > f2pitch6) && (f2pitch6 > f2pitch7) && (f2pitch7 > f2pitch8)
                          && (Math.abs(f2pitch8 - f2pitch1) > (f2pitch8 / 10) || (Math.abs(f2pitch1 - f2pitch8) > (f2pitch1 / 10)));
                  boolean f2pitchdowntoup = (f2pitch1 < f2pitch2) && (f2pitch2 < f2pitch3) && (f2pitch3 < f2pitch4) && (f2pitch4 < f2pitch5) &&
                          (f2pitch5 < f2pitch6) && (f2pitch6 < f2pitch7) && (f2pitch7 < f2pitch8)
                          && (Math.abs(f2pitch8 - f2pitch1) > (f2pitch8 / 10) || (Math.abs(f2pitch1 - f2pitch8) > (f2pitch1 / 10)));

                  boolean f3pitchuptodown = (f3pitch1 > f3pitch2) && (f3pitch2 > f3pitch3) && (f3pitch3 > f3pitch4) && (f3pitch4 > f3pitch5) &&
                          (f3pitch5 > f3pitch6) && (f3pitch6 > f3pitch7) && (f3pitch7 > f3pitch8)
                          && (Math.abs(f3pitch8 - f3pitch1) > (f3pitch8 / 10) || (Math.abs(f3pitch1 - f3pitch8) > (f3pitch1 / 10)));
                  boolean f3pitchdowntoup = (f3pitch1 < f3pitch2) && (f3pitch2 < f3pitch3) && (f3pitch3 < f3pitch4) && (f3pitch4 < f3pitch5) &&
                          (f3pitch5 < f3pitch6) && (f3pitch6 < f3pitch7) && (f3pitch7 < f3pitch8)
                          && (Math.abs(f3pitch8 - f3pitch1) > (f3pitch8 / 10) || (Math.abs(f3pitch1 - f3pitch8) > (f3pitch1 / 10)));


                  boolean f1rolluptodown = (f1roll1 > f1roll2) && (f1roll2 > f1roll3) && (f1roll3 > f1roll4) && (f1roll4 > f1roll5) &&
                          (f1roll5 > f1roll6) && (f1roll6 > f1roll7) && (f1roll7 > f1roll8)
                          && (Math.abs(f1roll8 - f1roll1) > (f1roll8 / 10) || (Math.abs(f1roll1 - f1roll8) > (f1roll1 / 10)));
                  boolean f1rolldowntoup = (f1roll1 < f1roll2) && (f1roll2 < f1roll3) && (f1roll3 < f1roll4) && (f1roll4 < f1roll5) &&
                          (f1roll5 < f1roll6) && (f1roll6 < f1roll7) && (f1roll7 < f1roll8)
                          && (Math.abs(f1roll8 - f1roll1) > (f1roll8 / 10) || (Math.abs(f1roll1 - f1roll8) > (f1roll1 / 10)));
                  boolean f1rolluptodownflip = (f1roll1 < f1roll2) && (f1roll2 > f1roll3) && (f1roll3 > f1roll4) && (f1roll4 < f1roll5) && (f1roll4 < -140 && f1roll5 > 140) &&
                          (f1roll5 > f1roll6) && (f1roll6 > f1roll7) && (f1roll7 > f1roll8)
                          && (Math.abs(f1roll8 - f1roll1) > (f1roll8 / 10) || (Math.abs(f1roll1 - f1roll8) > (f1roll1 / 10)));
                  boolean f1rolldowntoupflip = (f1roll1 < f1roll2) && (f1roll2 < f1roll3) && (f1roll3 < f1roll4) && (f1roll4 > f1roll5) && (f1roll4 > 140 && f1roll5 < -140) &&
                          (f1roll5 < f1roll6) && (f1roll6 < f1roll7) && (f1roll7 < f1roll8)
                          && (Math.abs(f1roll8 - f1roll1) > (f1roll8 / 10) || (Math.abs(f1roll1 - f1roll8) > (f1roll1 / 10)));

                  boolean f2rolluptodown = (f2roll1 > f2roll2) && (f2roll2 > f2roll3) && (f2roll3 > f2roll4) && (f2roll4 > f2roll5) &&
                          (f2roll5 > f2roll6) && (f2roll6 > f2roll7) && (f2roll7 > f2roll8)
                          && (Math.abs(f2roll8 - f2roll1) > (f2roll8 / 10) || (Math.abs(f2roll1 - f2roll8) > (f2roll1 / 10)));
                  boolean f2rolldowntoup = (f2roll1 < f2roll2) && (f2roll2 < f2roll3) && (f2roll3 < f2roll4) && (f2roll4 < f2roll5) &&
                          (f2roll5 < f2roll6) && (f2roll6 < f2roll7) && (f2roll7 < f2roll8)
                          && (Math.abs(f2roll8 - f2roll1) > (f2roll8 / 10) || (Math.abs(f2roll1 - f2roll8) > (f2roll1 / 10)));
                  boolean f2rolluptodownflip = (f2roll1 > f2roll2) && (f2roll2 > f2roll3) && (f2roll3 > f2roll4) && (f2roll4 < f2roll5) && (f2roll4 < -140 && f2roll5 > 140) &&
                          (f2roll5 > f2roll6) && (f2roll6 > f2roll7) && (f2roll7 > f2roll8)
                          && (Math.abs(f2roll8 - f2roll1) > (f2roll8 / 10) || (Math.abs(f2roll1 - f2roll8) > (f2roll1 / 10)));
                  boolean f2rolldowntoupflip = (f2roll1 < f2roll2) && (f2roll2 < f2roll3) && (f2roll3 < f2roll4) && (f2roll4 > f2roll5) && (f2roll4 > 140 && f2roll5 < -140) &&
                          (f2roll5 < f2roll6) && (f2roll6 < f2roll7) && (f2roll7 < f2roll8)
                          && (Math.abs(f2roll8 - f2roll1) > (f2roll8 / 10) || (Math.abs(f2roll1 - f2roll8) > (f2roll1 / 10)));

```

Figure 9-67 possible movements of the Angles 1

```

                  boolean f3rolluptodown = (f3roll1 > f3roll2) && (f3roll2 > f3roll3) && (f3roll3 > f3roll4) && (f3roll4 > f3roll5) &&
                          (f3roll5 > f3roll6) && (f3roll6 > f3roll7) && (f3roll7 > f3roll8)
                          && (Math.abs(f3roll8 - f3roll1) > (f3roll8 / 10) || (Math.abs(f3roll1 - f3roll8) > (f3roll1 / 10)));
                  boolean f3rolldowntoup = (f3roll1 < f3roll2) && (f3roll2 < f3roll3) && (f3roll3 < f3roll4) && (f3roll4 < f3roll5) &&
                          (f3roll5 < f3roll6) && (f3roll6 < f3roll7) && (f3roll7 < f3roll8)
                          && (Math.abs(f3roll8 - f3roll1) > (f3roll8 / 10) || (Math.abs(f3roll1 - f3roll8) > (f3roll1 / 10)));
                  boolean f3rolluptodownflip = (f3roll1 > f3roll2) && (f3roll2 > f3roll3) && (f3roll3 > f3roll4) && (f3roll4 < f3roll5) && (f3roll4 < -140 && f3roll5 > 140) &&
                          (f3roll5 > f3roll6) && (f3roll6 > f3roll7) && (f3roll7 > f3roll8)
                          && (Math.abs(f3roll8 - f3roll1) > (f3roll8 / 10) || (Math.abs(f3roll1 - f3roll8) > (f3roll1 / 10)));
                  boolean f3rolldowntoupflip = (f3roll1 < f3roll2) && (f3roll2 < f3roll3) && (f3roll3 < f3roll4) && (f3roll4 > f3roll5) && (f3roll4 > 140 && f3roll5 < -140) &&
                          (f3roll5 < f3roll6) && (f3roll6 < f3roll7) && (f3roll7 < f3roll8)
                          && (Math.abs(f3roll8 - f3roll1) > (f3roll8 / 10) || (Math.abs(f3roll1 - f3roll8) > (f3roll1 / 10)));


                  boolean f1yawuptodown = (f1yaw1 > f1yaw2) && (f1yaw2 > f1yaw3) && (f1yaw3 > f1yaw4) && (f1yaw4 > f1yaw5) &&
                          (f1yaw5 > f1yaw6) && (f1yaw6 > f1yaw7) && (f1yaw7 > f1yaw8)
                          && (Math.abs(f1yaw8 - f1yaw1) > (f1yaw8 / 10) || (Math.abs(f1yaw1 - f1yaw8) > (f1yaw1 / 10)));

                  boolean f1yawdowntoup = (f1yaw1 < f1yaw2) && (f1yaw2 < f1yaw3) && (f1yaw3 < f1yaw4) && (f1yaw4 < f1yaw5) &&
                          (f1yaw5 < f1yaw6) && (f1yaw6 < f1yaw7) && (f1yaw7 < f1yaw8)
                          && (Math.abs(f1yaw8 - f1yaw1) > (f1yaw8 / 10) || (Math.abs(f1yaw1 - f1yaw8) > (f1yaw1 / 10)));

                  boolean f1yawuptodownflip = (f1yaw1 > f1yaw2) && (f1yaw2 > f1yaw3) && (f1yaw3 > f1yaw4) && (f1yaw4 < f1yaw5) && (f1yaw4 < -140 && f1yaw5 > 140) &&
                          (f1yaw5 > f1yaw6) && (f1yaw6 > f1yaw7) && (f1yaw7 > f1yaw8)
                          && (Math.abs(f1yaw8 - f1yaw1) > (f1yaw8 / 10) || (Math.abs(f1yaw1 - f1yaw8) > (f1yaw1 / 10)));
                  boolean f1yawdowntoupflip = (f1yaw1 < f1yaw2) && (f1yaw2 < f1yaw3) && (f1yaw3 < f1yaw4) && (f1yaw4 > f1yaw5) && (f1yaw4 > 140 && f1yaw5 < -140) &&
                          (f1yaw5 < f1yaw6) && (f1yaw6 < f1yaw7) && (f1yaw7 < f1yaw8)
                          && (Math.abs(f1yaw8 - f1yaw1) > (f1yaw8 / 10) || (Math.abs(f1yaw1 - f1yaw8) > (f1yaw1 / 10)));

                  boolean f2yawuptodown = (f2yaw1 > f2yaw2) && (f2yaw2 > f2yaw3) && (f2yaw3 > f2yaw4) && (f2yaw4 > f2yaw5) &&
                          (f2yaw5 > f2yaw6) && (f2yaw6 > f2yaw7) && (f2yaw7 > f2yaw8)
                          && (Math.abs(f2yaw8 - f2yaw1) > (f2yaw8 / 10) || (Math.abs(f2yaw1 - f2yaw8) > (f2yaw1 / 10)));
                  boolean f2yawdowntoup = (f2yaw1 < f2yaw2) && (f2yaw2 < f2yaw3) && (f2yaw3 < f2yaw4) && (f2yaw4 < f2yaw5) &&
                          (f2yaw5 < f2yaw6) && (f2yaw6 < f2yaw7) && (f2yaw7 < f2yaw8)
                          && (Math.abs(f2yaw8 - f2yaw1) > (f2yaw8 / 10) || (Math.abs(f2yaw1 - f2yaw8) > (f2yaw1 / 10)));
                  boolean f2yawuptodownflip = (f2yaw1 > f2yaw2) && (f2yaw2 > f2yaw3) && (f2yaw3 > f2yaw4) && (f2yaw4 < f2yaw5) && (f2yaw4 < -140 && f2yaw5 > 140) &&
                          (f2yaw5 > f2yaw6) && (f2yaw6 > f2yaw7) && (f2yaw7 > f2yaw8)
                          && (Math.abs(f2yaw8 - f2yaw1) > (f2yaw8 / 10) || (Math.abs(f2yaw1 - f2yaw8) > (f2yaw1 / 10)));
                  boolean f2yawdowntoupflip = (f2yaw1 < f2yaw2) && (f2yaw2 < f2yaw3) && (f2yaw3 < f2yaw4) && (f2yaw4 > f2yaw5) && (f2yaw4 > 140 && f2yaw5 < -140) &&
                          (f2yaw5 < f2yaw6) && (f2yaw6 < f2yaw7) && (f2yaw7 < f2yaw8)
                          && (Math.abs(f2yaw8 - f2yaw1) > (f2yaw8 / 10) || (Math.abs(f2yaw1 - f2yaw8) > (f2yaw1 / 10)));


                  boolean f3yawuptodown = (f3yaw1 > f3yaw2) && (f3yaw2 > f3yaw3) && (f3yaw3 > f3yaw4) && (f3yaw4 > f3yaw5) &&
                          (f3yaw5 > f3yaw6) && (f3yaw6 > f3yaw7) && (f3yaw7 > f3yaw8)
                          && (Math.abs(f3yaw8 - f3yaw1) > (f3yaw8 / 10) || (Math.abs(f3yaw1 - f3yaw8) > (f3yaw1 / 10)));
                  boolean f3yawdowntoup = (f3yaw1 < f3yaw2) && (f3yaw2 < f3yaw3) && (f3yaw3 < f3yaw4) && (f3yaw4 < f3yaw5) &&
                          (f3yaw5 < f3yaw6) && (f3yaw6 < f3yaw7) && (f3yaw7 < f3yaw8)
                          && (Math.abs(f3yaw8 - f3yaw1) > (f3yaw8 / 10) || (Math.abs(f3yaw1 - f3yaw8) > (f3yaw1 / 10)));
                  boolean f3yawuptodownflip = (f3yaw1 > f3yaw2) && (f3yaw2 > f3yaw3) && (f3yaw3 > f3yaw4) && (f3yaw4 < f3yaw5) && (f3yaw4 < -140 && f3yaw5 > 140) &&
                          (f3yaw5 > f3yaw6) && (f3yaw6 > f3yaw7) && (f3yaw7 > f3yaw8)
                          && (Math.abs(f3yaw8 - f3yaw1) > (f3yaw8 / 10) || (Math.abs(f3yaw1 - f3yaw8) > (f3yaw1 / 10)));
                  boolean f3yawdowntoupflip = (f3yaw1 < f3yaw2) && (f3yaw2 < f3yaw3) && (f3yaw3 < f3yaw4) && (f3yaw4 > f3yaw5) && (f3yaw4 > 140 && f3yaw5 < -140) &&
                          (f3yaw5 < f3yaw6) && (f3yaw6 < f3yaw7) && (f3yaw7 < f3yaw8)
                          && (Math.abs(f3yaw8 - f3yaw1) > (f3yaw8 / 10) || (Math.abs(f3yaw1 - f3yaw8) > (f3yaw1 / 10)));
```

Figure 9-68 possible movements of the Angles 2

The above functions are all the various modification that can happen to the fused data of the algorithm. These range from increase to decrease to complex increase and flips from positive 180 to negative 180 which are checked by using a separate function in order to successfully perceive that this action has occurred.

9.5 Testing/Results

9.5.1 Authentication Function

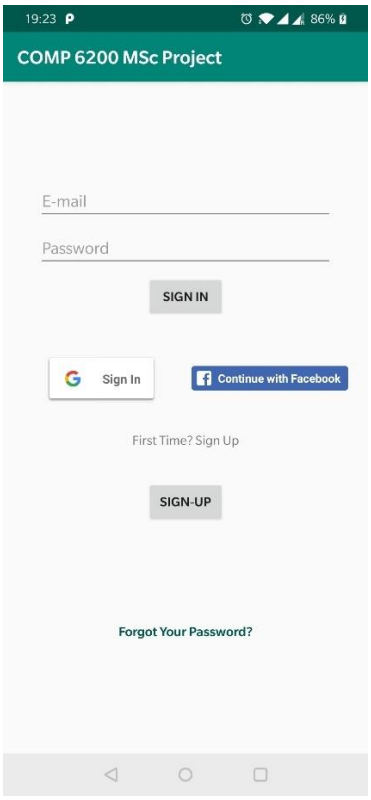

Figure 9-69 Sign In

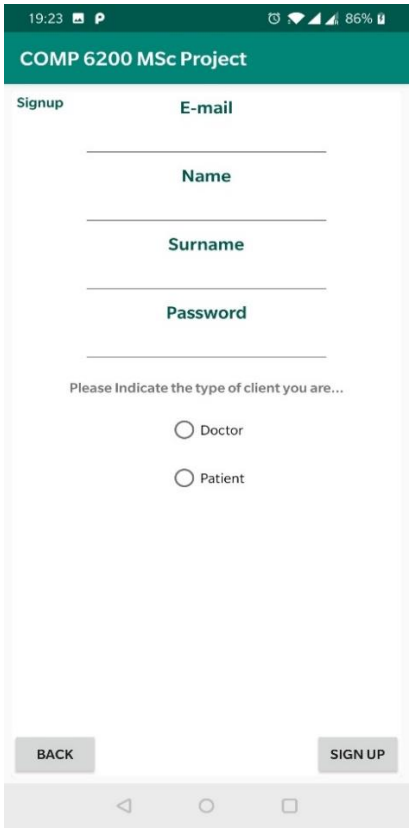

Figure 9-70 Signup

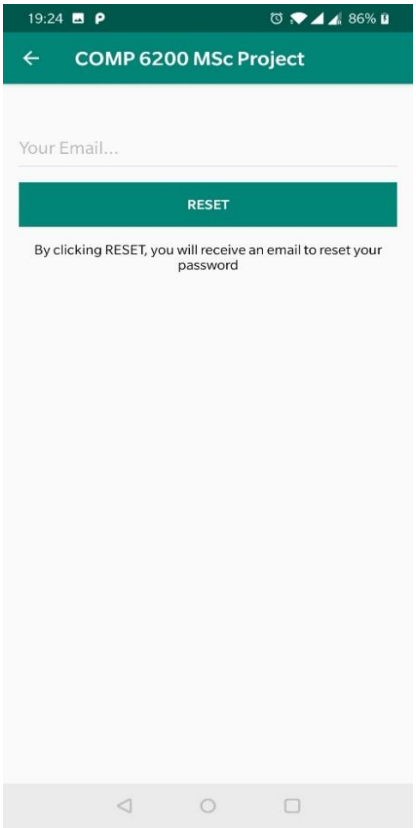

Figure 9-71 Reset Password

The above functions illustrate how the user can be authenticated and to alleviate any problems with forgotten passwords a remediation method of "forgotten password" has been implemented in order to enable users to regain access to their information.
The implementation is then divided based on the type of user.
Shown below is the Patients' Point of View.

9.5.2 Patients

The patient has multiple abilities as clearly shown below.

9.5.2.1 Initial Fragment

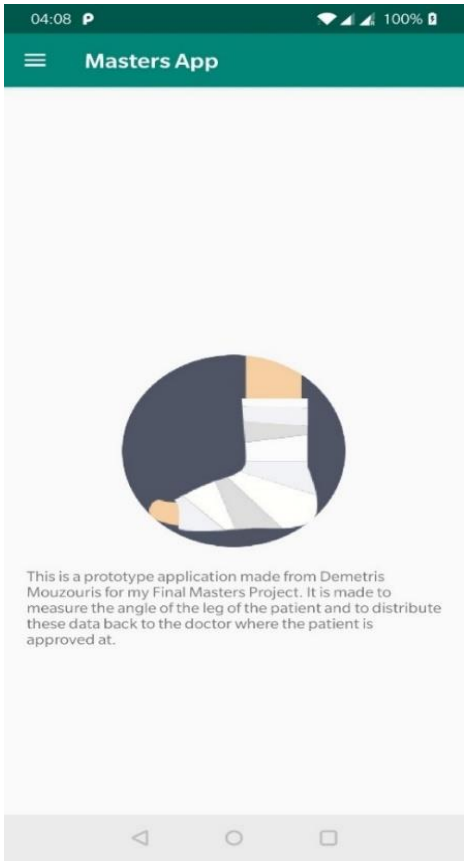

Figure 9-72 Initial Fragment

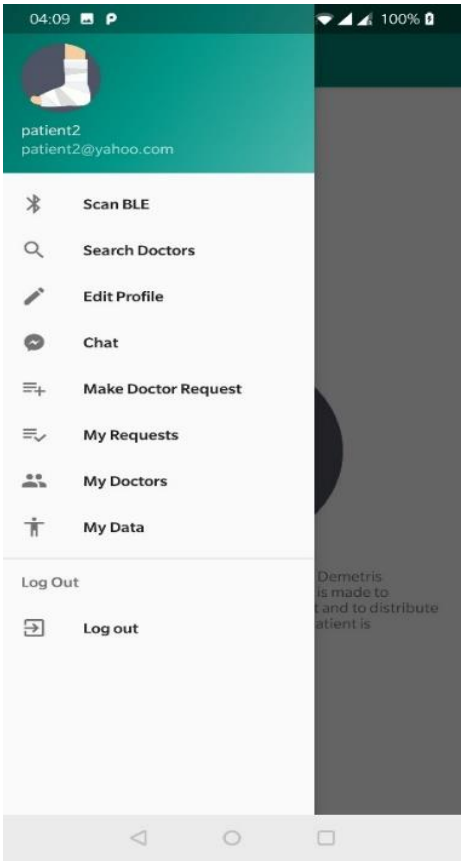

Figure 9-73 Navigation Bar

The above functions illustrate how the Patient is greeted by the application and is shown his navigation system.

9.5.2.2 Scan BLE

Figure 9-74 Found Devices

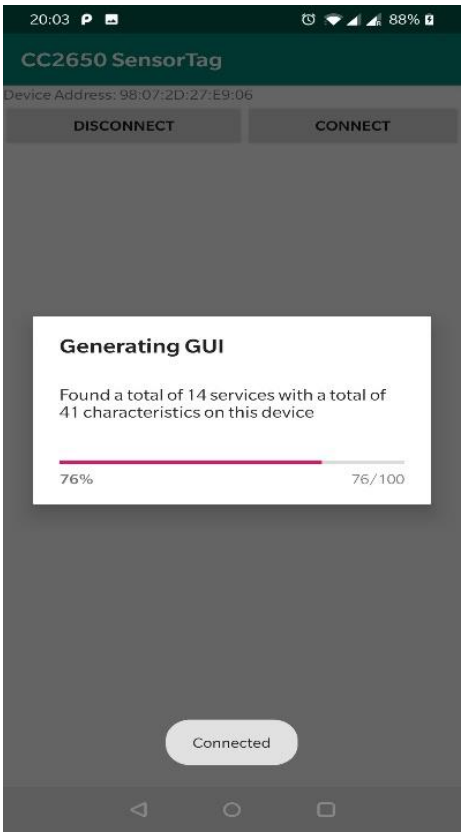

Figure 9-75 Connecting services of the device

The above function is the first fragment that enables the user to connect to the Sensor which will then enable them to collect the appropriate data and start utilising the elevation functions.

9.5.2.3 Search Doctors

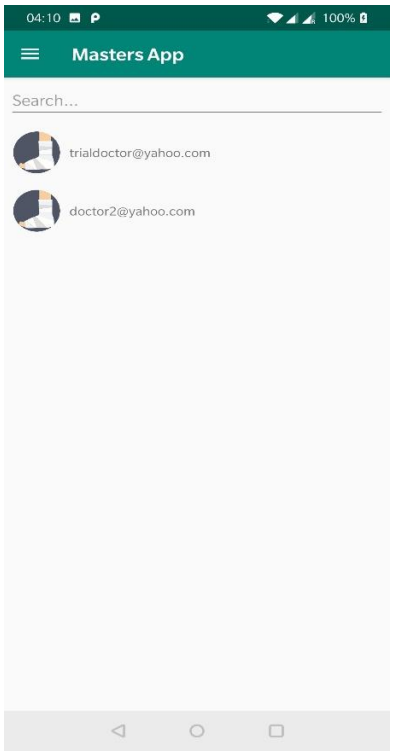

Figure 9-76 Search Doctors

9.5.2.4 Edit Profile

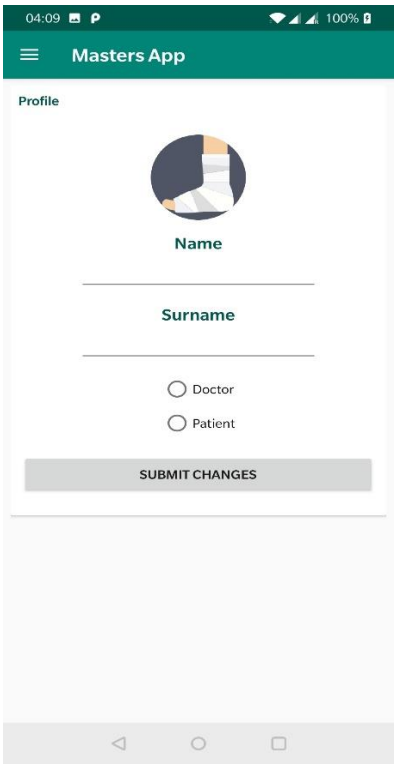

Figure 9-77 Edit Profile

Above are the functions that enable the patient to search all doctors and edit their profile. As illustrated by the Patient design section.

9.5.2.5 Chats/Communication

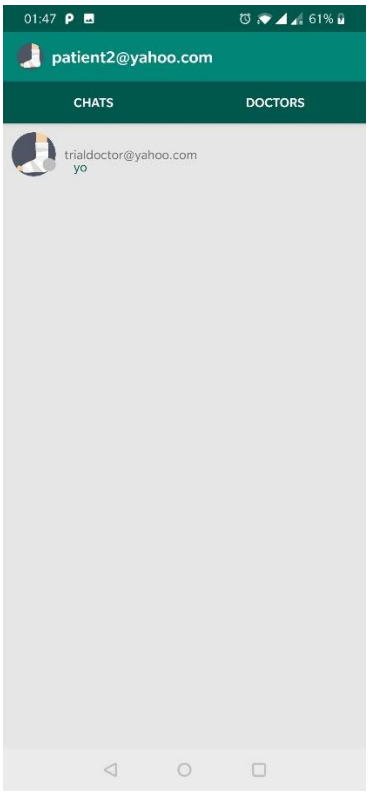

Figure 9-78 Initial Chat View

Figure 9-79 Chat

The above figures illustrate how the chat and communication section operates illustrating what is seen by the patient and how those individuals can communicate with one another.

9.5.2.6 Make Requests

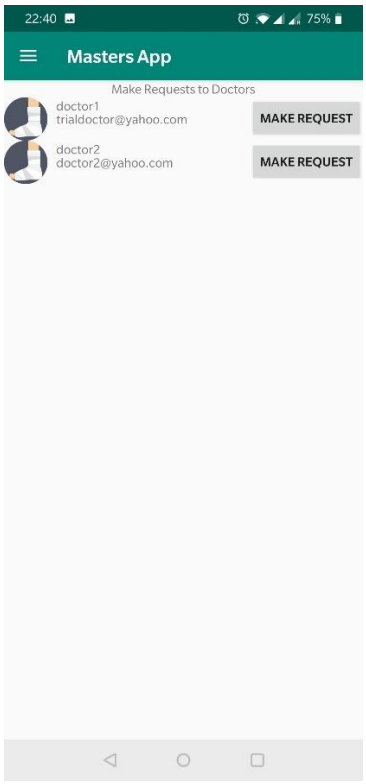

Figure 9-80 Make Requests

9.5.2.7 My Requests

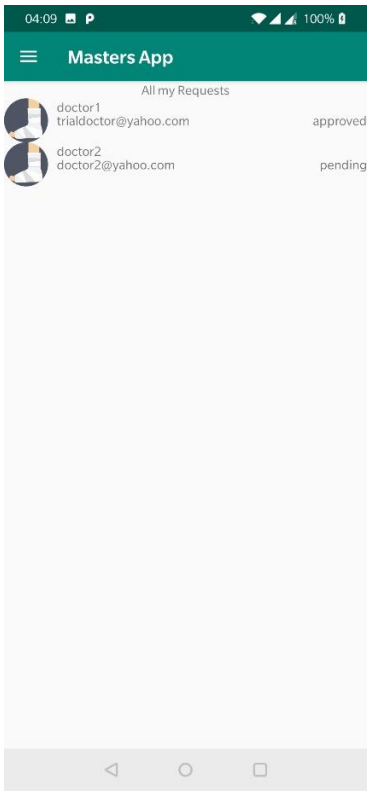

Figure 9-81 My Requests

In the above images, it indicates how the patient can make request to the approved doctors and how those requests status is shown based on the fragment Make requests and My requests.

9.5.2.8 My Doctors

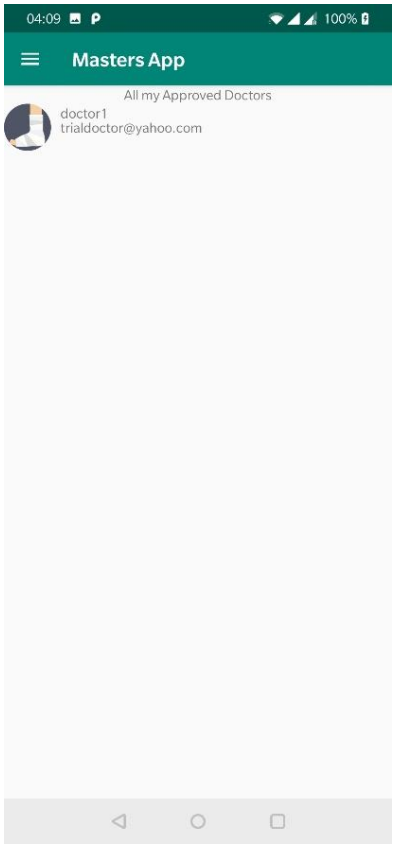

Figure 9-82 My Doctors

Figure 9-83 My Doctors Information

Shown above is the last function available to the Patient which illustrates their doctors that are supervising them as well as all their relevant information.

9.5.3 Doctors

Doctors have similar abilities available to them shown below is the greeting navigation bar.

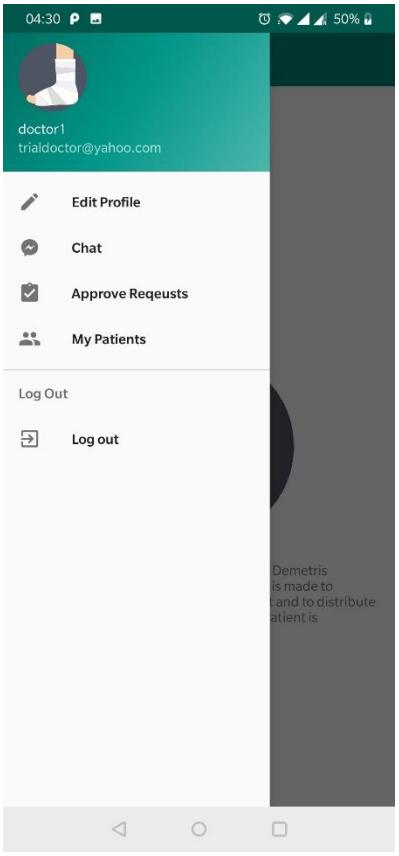

Figure 9-84 - Navigation Bar Doctor

9.5.3.1 Chats/Communication

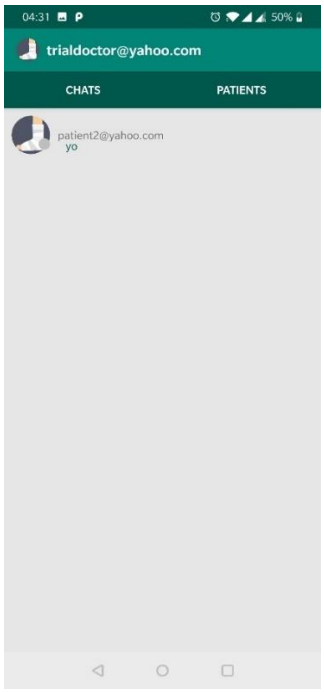

Figure 9-85 - Chats

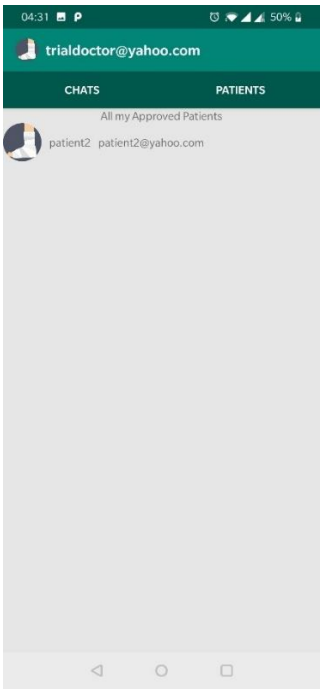

Figure 9-86 Approved patients as chat list

9.5.3.2 Approve/Decline Patients

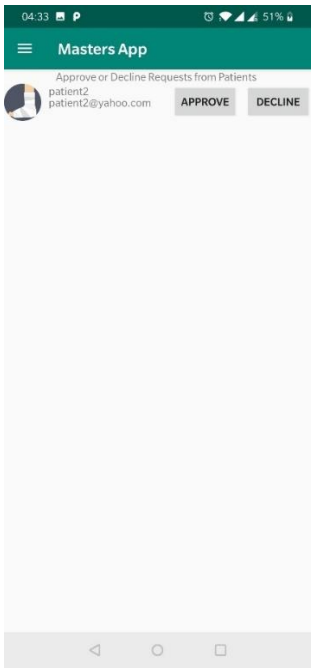

Figure 9-87 Approve/Decline Patients

In the above figures, an illustration of the operations of doctors are identified. The difference of chat is the chat list illustrated being the doctors' patients. These functions although are only available in the case when the doctor is approved, and his type is of "Doctor". In case the doctor is not approved the only thing illustrated would be the Edit profile function in order to enable the user to revert back to Patient.

Application Table of Content

Activities

Main Activity
ChatActivity
Chat
Message Activity
Login Activity
ResetPasswordActivity
SensorTagConnect
SignupActivity

Fragments

Approve Doctors
Approve Requests
Chats Fragment
Display Readings
DisplayReadings2
Initial Fragment
Kalman Filter
Make Requests
Motion Fragment
My Data
My Doctors
My Doctors Chat
My Patients
My Patients Chat
My Requests
Profile Fragment
Scan_BLE
Search Fragments

Additional Important Classes

Madgwick Algorithm
Mahony Algorithm
Step Detector – StepCounter2, Statistics Util
Sensor Tag – Sensor Conversion
Services – BLE Service

Chapter 10 Ergo Application

FEPS Ethics Committee

10.1 Ver 4

Refer to the *Instructions* and to the *Guide* documents for a glossary of the key phrases in **bold** and for an explanation of the information required in each section. The *Templates* document provides some text that may be helpful in preparing some of the required appendices.

Replace the highlighted text with the appropriate information.

Note that the size of the text entry boxes provided on this form does **not** indicate the expected amount of information; instead, refer to the *Instructions* and to the *Guide* documents in providing the complete information required in each section. Do **not** duplicate information from one text box to another. Do not otherwise edit this form.

Reference number: **ERGO/Error! Unknown document property name.**/50707	Submission version: 4	Date: 2019-08-12
Name of the **investigator**(s): Demetris Mouzouris		
Name of supervisor(s) (if student **investigator**(s)): Dr Neil White		
Title of study: Use of IoT to enhance rehabilitation monitoring in leg surgery patients		

Note that failure to follow the University's policy on Ethics may lead to disciplinary action concerning Misconduct or a breach of Academic Integrity.

By submitting this application, the investigator(s) undertake to:

- Conduct the study in accordance with University policies governing:
  Ethics (http://www.southampton.ac.uk/ris/policies/ethics.html);
  Data management (http://www.southampton.ac.uk/library/research/researchdata/);
  Health and Safety (http://www.southampton.ac.uk/healthandsafety);
  Academic Integrity (http://www.calendar.soton.ac.uk/sectionIV/academic-integrity-statement.html.
- Ensure the study Reference number ERGO/**Error! Unknown document property name.**/XXXX is prominently displayed on all advertising and study materials and is reported on all media and in all publications;
- Conduct the study in accordance with the information provided in the application, its appendices, and any other documents submitted;
- Submit the study for re-review (as an amendment through ERGO) or seek **Error! Unknown document property name.** EC advise if any changes, circumstances, or outcomes materially affect the study or the information given;
- Promptly advise an appropriate authority (Research Integrity and Governance team) of any adverse study outcomes;
- Submit an end-of-study form if required to do so.

Refer to the Instructions and Guide documents when completing this form and the Templates document when preparing the required appendices.

Study details

What are the aims and objectives of this study?
To develop an application that will measure the ankle incline of an individual in order to measure whether the leg has been elevated or not. As well as also to measure if the leg has been making movement measured in steps.

Background of the study (*a brief rationale for conducting the study)*
The study is required in order to identify whether there are any bugs within the application that will need to be eliminated. As well as also to understand what has the algorithm done in order to improve our readings. This also will help us to understand the accuracy of the application as well as how to better the whole implementation in order to get better results. The device that will be utilised in this experiment is the Sensor Tag cc2650 which was developed by the company Texas Instruments. This device has been developed as a general usage device that detects readings from its multiple sensors and is up to the developer to utilise the features provided by this generic device in order to tailor it to the users' needs. It was not specifically developed for this purpose. Although based on the fact that it is open source, we will utilise its features in order to tailor its readings to progress our needs.

Key research question *(Specify hypothesis if applicable*)
There is a need to transmit data from the patient to the doctor in order for the doctor to process what has the patient done and if he is following the instructions that have been given by the doctor at his own habitat.

Study design *(Give a **brief** outline of the study design and why it is being used*)
The project is used in order to aid medicinal practices that are currently outdated or non-existent where in our case the only way there was communication between the doctor and the patient would be a phone call although this application bridges the gap and also provides to the doctors and patients the one-to-one communication required at the comfort of the patients home. The design will be mostly the development of the application and the communication between the two devices (Sensor and Mobile phone). The collection process has been designed to be anonymised therefore data will be collected from real users although will be attached to fake users such as "TestSubject1", the process of testing will be the users attaching on their ankle the sensor using a strap or inputting the sensor in their socks and we will be collecting the data from the sensor on a handheld mobile device. In order to see how the data, correspond to the movement. And to enable us to understand the algorithm operation.

Pre-study

Characterise the proposed **participants**
The proposed participants would be healthy human beings that are able to elevate their legs on command, in order to measure readings. This study will not involve clinicians during the trial taking place.

Describe how **participants** will be approached
If any e-mail lists are used, including FEPS distribution lists, justify their use *here* *Potential participants will be approached on the University Campus, where I will have multiple copies of the PIS form that will state the relevant information that the user should know before taking part in the experiment. In case participants are interested in participating. They will then be given the consent form to read and to agree on the terms. And the experiment will start momentarily after they have agreed on participating. Which will then enable the experiment to commence. No information from the users will be required as this participation will not be required to be repeated for further readings, nor will participants be recruited if they have no available time momentarily after reading the PIS form.*

Describe how inclusion/exclusion criteria will be applied (if any)
Any minors will be excluded from the application testing. The minimum requirements would be individuals that are able to raise their leg at a predefined higher than 90degree angle.

Describe how **participants** will decide whether or not to take part
The participants have the ability to withdraw from the testing time at any given time. The participants will be asked if they could spare a few minutes of their time to take part in this experiment. By notifying the researcher that they do not want to participate in this experiment. Therefore, no data will be kept for individuals that would not want to participate.

Participant Information (Appendix (i))

Provide the **Participant Information** in the form that it will be given to **participants** as Appendix (i). All studies must provide **participant information**.

Consent Form/Information (Appendix (iii))

Provide the **Consent Form** (or the request for consent) in the form that it will be given to **participants** as Appendix (iii). All studies must obtain **participant** consent. Some studies may obtain verbal consent (and only present consent information), other studies will require written consent, as explained in the *Instructions, Guide,* and *Templates* documents.

During the study

Describe the study procedures as they will be experienced by the **participants**
The study will commence at the University of Southampton Campus. It will take approximately 15-20 minutes for the whole experiment to finalise. And participants will be required to do 2 tasks, walking and raising their leg 3 times where readings will be recorded throughout the period of the user's usage. There will be given a sensor that will be inputted within the participant's sock or will be strapped on the ankle of the participant. The sensor will then be turned on and a "dummy" user will be created while the sensor is on the participant. These data will be recorded and attached to the "dummy" user, therefore, keeping data anonymised which will then be studied for further improving the algorithm and the application.

Identify how, when, where, and what kind of data will be recorded (not just the formal research data, but including all other study data such as e-mail addresses and signed consent forms)
The data will all be anonymised, and dummy users will be inputted within the place of the participants. The data will be the angle of the sensor based on its location. And, whether the user has taken any footsteps. As well as Consent forms will be taken in order to allow us to carry out the experiment.

Participant questionnaire/data gathering methods (Appendix (ii))

As Appendix (ii), reproduce any and all **participant** questionnaires or data gathering instruments in the exact forms that they will be given to or experienced by **participants**. If conducting less formal data collection, or data collection that does not involve direct questioning or observation of participants (eg secondary data or "big data"), provide specific information concerning the methods that will be used to obtain the data of the study.

Post-study

Identify how, when, and where data will be stored, processed, and destroyed
If the Study Characteristic M.1 applies, provide this information in the **DPA Plan** as Appendix (iv) instead and do *not* provide explanation or information on this matter here

Study characteristics

(L.1) The study is funded by a commercial organisation: **No** (delete one) If 'Yes', provide details of the funder or funding agency *here.*

(L.2) There are **restrictions** upon the study: -**No** (delete one) If 'Yes', explain the nature and necessity of the **restrictions** *here.*

(L.3) Access to **participants** is through a third party: **No** (delete one) If 'Yes', provide evidence of your permission to contact them as (v) in the *Checklist* below. Do *not* provide an explanation or information on this matter here.

(M.1) **Personal data** is or *may be collected or processed: **No** (delete one) Data will be processed outside the UK: -**No** (delete one) If 'Yes' to either question, provide the **DPA Plan** as (iv) in the *Checklist* below. Do *not* provide information or explanation on this matter here. Note that using or recording e-mail addresses, telephone numbers, signed consent forms, or similar study-related **personal data** requires M.1 to be "Yes". (* Secondary data / "big data" may be *de*-anonymised, or may contain **personal data**. If so, answer 'Yes'.)

(M.2) There is an **inducement** to **participants**: **No** (delete one) If 'Yes', explain the nature and necessity of the inducement *here.*

(M.3) The study is **intrusive**: **No** (delete one) If 'Yes', provide the **Risk Management Plan,** the **Debrief Plan,** and Technical Details as (vi), (vii), and (ix) in the *Checklist* below, and explain *here* the nature and necessity of the intrusion(s).

(M.4) There is a **risk of harm** during the study: **No** (delete one) If 'Yes', provide the **Risk Management Plan**, the **Contact Information**, the **Debrief Plan**, and Technical Details as (vi), (vii), (viii), and (ix) in the *Checklist* below, and explain *here* the necessity of the risks.

(M.5) The true purpose of the study will be hidden from **participants**: **No** (delete one)
The study involves **deception** of **participants**: **No** (delete one)

If 'Yes' to either question, provide the **Debrief Plan** and Technical Details as (vii) and (ix) in the *Checklist* below, and explain *here* the necessity of the deception.

(M.6) **Participants** may be minors or otherwise have **diminished capacity**: **No** (delete one)

If 'Yes', AND if one or more Study Characteristics in categories M or H applies, provide the **Risk Management Plan,** the **Contact Information**, and Technical Details as (vi), (vii), & (ix) in the *Checklist* below, and explain *here* the special arrangements that will ensure informed consent.

(M.7) **Special category personal data** is collected or processed: **No** (delete one)

If 'Yes', provide the **DPA Plan** and Technical Details as (iv) and (ix) in the *Checklist* below. Do *not* provide an explanation or information on this matter here.

(H.1) The study involves: **invasive** equipment, material(s), or process(es); or **participants** who are not able to withdraw at any time and for any reason; or animals; or human tissue; or biological samples: **No** (delete one)

If 'Yes', provide Technical Details and further justifications as (ix) and (x) in the *Checklist* below. Do *not* provide explanation or information on these matters here. Note that the study will require separate approval by the Research Governance Office.

Technical details

If one or more Study Characteristics in categories M.3 to M.7 or H applies, provide the description of the technical details of the experimental or study design, the power calculation(s) which yield the required sample size(s), and how the data will be analysed, as separate appendices.

Checklist of Documents to upload

Please provide the following forms, *naming the files as explicitly* as possible, e.g., "Participant Information", "Questionnaire", "Consent Form", "DPA Plan", "Permission to contact", "Risk Management Plan", "Debrief Plan", "Contact Information", and/or "Technical details" as appropriate. Each document must specify the reference number in the form ERGO/**Error! Unknown document property name.**/XXXX, the document version number, and its date of last edit.

(i): **Participant Information** in the form that it will be given to **participants.**

(ii): Data collection method (eg for secondary data or "big data") / **Participant** Questionnaire in the form that it will be given to **participants.**

(iii): **Consent Form** (or consent information if no **personal data** is collected) in the form that it will be given to **participants.**

(iv): **DPA Plan.**

(v): Evidence of permission to contact (prospective) **participants** through any third party.

(vi): **Risk Management Plan**.

(vii): **Debrief Plan**.

(viii): **Contact Information**.

(ix): Technical details of the experimental or study design, the power calculation(s) for the required sample size(s), and how the data will be analysed.

(x): Further details and justifications in the case of: **invasive** equipment, material(s), or process(es); **participants** who are not able to withdraw at any time and for any reason; animals; human tissue; or biological samples.

10.2 Participant Information Sheet

Study Title: Use of IoT to enhance rehabilitation monitoring in leg surgery patients

Researcher: Demetris Mouzouris

ERGO number: 50707

You are being invited to take part in the above research study. To help you decide whether you would like to take part or not, it is important that you understand why the research is being done and what it will involve. Please read the information below carefully and ask questions if anything is not clear or you would like more information before you decide to take part in this research. You may like to discuss it with others, but it is up to you to decide whether or not to take part. If you are happy to participate you will be asked to agree on a consent form.

What is the research about?

This project is made for my Final Master's Project. I am Demetris Mouzouris and I am doing this research in order to better aid the communication between doctors and patients and also to give the ability to the doctor to monitor readings from the patient in order to see progress and if the patient is complying with the doctor's instructions. The expected outcome of this project is an application that will receive the data and based on the algorithms of the application it will then indicate elevation and footsteps. The sensor that will be utilised for this application is developed by the company Texas Instruments and will be utilised for the carrying out of this experiment.

Why have I been asked to participate?

There will be 8 participants, you have been chosen as you are an adult individual at an age above 18 who is able to raise the leg at an angle higher than 90 degrees

What will happen to me if I take part?

You will be inputting a sensor inside your sock or with a strap on your ankle that will measure the movements that you would be making. This data will be collected anonymously and inputted into a database through a handheld smartphone device in order to see how the algorithm has reacted and also to see how the accuracy is affected in unknown environments. This data will help us to understand how well the algorithm is doing and then also improve it. You will be giving us a consent form in order for you to participate that will enable us to use this data to better the project. Being at the university, at the time of agreeing to participate and the information sheet has been provided to you and decided you would like to participate in this experiment, you will then be informed of the two tasks that you will require to make and also the sensor will be attached on your ankle. You would not be required to attend again. The project is expected to last until the 5th of September 2019 although your involvement is only required for the 15-20 minutes of the experiment and your data will be kept anonymously in the database until the final date of the project submission.

Are there any benefits in my taking part?

This study will not benefit the individuals participating although would benefit the understanding and also aid in breaching the gap between home monitoring for patients. Benefiting doctors and patients to communicate.

Are there any risks involved?

There are currently minor risks with the handheld device as it involves a lithium battery and will be located to close proximity of the user although it is highly unlikely that there could be any damage from the sensor to the individual taking part in the experiment. There is the possibility of losing balance while being asked to raise the leg for a period of time, therefore, to alleviate this risk there will be chairs and holding poles in order to alleviate individuals from falling down.

What data will be collected?

The data that will be collected from the user is the readings that will be collected from the user's ankle, this in order will enable us to improve the algorithms being developed to calculate incline and footsteps. The data will be collected by the Researcher Demetris Mouzouris and they will be collected on a handheld smartphone device which will be operated by the researcher and assign the data into anonymous test subject users. The

data collected will be stored into two locations locally within the handheld device which will be protected through a pin and the file will also be password protected. From any user, as well as the cloud on a firebase database that is protected through encryption and also protected with a password in order to gain access to that database.

Will my participation be confidential?

Your participation and the information we collect about you during the course of the research will be kept strictly confidential.

Only members of the research team and responsible members of the University of Southampton may be given access to data for monitoring purposes and/or to carry out an audit of the study to ensure that the research is complying with applicable regulations. Individuals from regulatory authorities (people who check that we are carrying out the study correctly) may require access to your data. All of these people have a duty to keep your information, as a research participant, strictly confidential.

The data will be encrypted in both scenarios the local file will be password protected as well as the firebase database is encrypted and can also be accessed only by the researcher who will be operating on the data that will be collected. The user's data will also be anonymised and will be unable to track the user's data even with access to the consent forms.

People who have access to the data will be the Supervisor as well as the researcher.

Do I have to take part?

No, it is entirely up to you to decide whether or not to take part. If you decide you want to take part, you will need to agree on the consent form that will then enable you to take part in the experiment.

At any time, the participant has the ability to opt-out of the experiment even during the process of the experiment although the data collected up to the point of the experiment will still be kept for further processing in the application.

What happens if I change my mind?

You have the right to change your mind and withdraw at any time without giving a reason and without your participant rights *(or routine care if a patient)* being affected.

If you withdraw from the study, we will keep the information about you that we have already obtained for the purposes of achieving the objectives of the study only.

What will happen to the results of the research?

Your data will remain strictly confidential. Research findings made available in any reports or publications will not include information that can directly identify you without your specific consent.

The project will be written up in the final Dissertation which will utilise the data to prove the accuracy and operability of the sensor. The personal details of the users will not be collected therefore the users will not be directly mentioned in the dissertation.

Where can I get more information?

For further questions contact Demetris Mouzouris: dm1u18@soton.ac.uk

What happens if there is a problem?

If you have a concern about any aspect of this study, you should speak to the researchers who will do their best to answer your questions.

If you remain unhappy or have a complaint about any aspect of this study, please contact the University of Southampton Research Integrity and Governance Manager (023 8059 5058, rgoinfo@soton.ac.uk).

Contact: dm1u18@soton.ac.uk for any enquiries or if you are unsatisfied by any part of this study.

Data Protection Privacy Notice

The University of Southampton conducts research to the highest standards of research integrity. As a publicly-funded organisation, the University has to ensure that it is in the public interest when we use personally-identifiable information about people who have agreed to take part in the research. This means that when you agree to take part in a research study, we will use information about you in the ways needed, and for the purposes specified, to conduct and complete the research project. Under data protection law, 'Personal data' means any information that relates to and is capable of identifying a living individual. The University's data protection policy governing the use of personal data by the University can be found on its website (https://www.southampton.ac.uk/legalservices/what-we-do/data-protection-and-foi.page).

This Participant Information Sheet tells you what data will be collected for this project and whether this includes any personal data. Please ask the research team if you have any questions or are unclear what data is being collected about you.

Our privacy notice for research participants provides more information on how the University of Southampton collects and uses your personal data when you take part in one of our research projects and can be found at http://www.southampton.ac.uk/assets/sharepoint/intranet/ls/Public/Research%20and%20Integrity%20Privacy%20Notice/Privacy%20Notice%20for%20Research%20Participants.pdf

Any personal data we collect in this study will be used only for the purposes of carrying out our research and will be handled according to the University's policies in line with data protection law. If any personal data is used from which you can be identified directly, it will not be disclosed to anyone else without your consent unless the University of Southampton is required by law to disclose it.

Data protection law requires us to have a valid legal reason ('lawful basis') to process and use your Personal Data. The lawful basis for processing personal information in this research study is for the performance of a task carried out in the public interest. Personal data collected for research will not be used for any other purpose.

For the purposes of data protection law, the University of Southampton is the 'Data Controller' for this study, which means that we are responsible for looking after your information and using it properly. The University of Southampton will keep identifiable information about you for 10 years after the study has finished after which time any link between you and your information will be removed.

To safeguard your rights, we will use the minimum personal data necessary to achieve our research study objectives. Your data protection rights – such as to access, change, or transfer such information - may be limited, however, in order for the research output to be reliable and accurate. The University will not do anything with your personal data that you would not reasonably expect.

If you have any questions about how your personal data is used, or wish to exercise any of your rights, please consult the University's data protection webpage (https://www.southampton.ac.uk/legalservices/what-we-do/data-protection-and-foi.page) where you can make a request using our online form. If you need further assistance, please contact the University's Data Protection Officer (data.protection@soton.ac.uk).

The data collected will be anonymised, the data collected will be the readings from the sensor alone as well as consent forms that will not link to the users.

Thank you.

10.3 Consent Form

CONSENT FORM

Study title: Use of IoT to enhance rehabilitation monitoring in leg surgery patients

Researcher name: Demetris Mouzouris
ERGO number: 50707

Please initial the box(es) if you agree with the statement(s):

I have read and understood the information sheet (12*/08/2019 /version no. 4 of the participant information sheet*) and have had the opportunity to ask questions about the study.	
I agree to take part in this research project and agree for my data to be used for the purpose of this study.	
I understand my participation is voluntary and I may withdraw (at any time) for any reason without my participation rights being affected.	
I provide consent to the fact that the study will utilise the data from the sensor and not any personal data will be recorded and utilised throughout the course of the project.	

Name of participant (print name)……

Signature of participant……

Date……

Name of researcher (print name) ……

Signature of researcher ……

Date……

10.4 DPA Plan

Ethics reference number: **ERGO**/FEPS/**50707**	Version: 1	Date: 2019-06-26
Study Title: Use of IoT to enhance rehabilitation monitoring in leg surgery patients		
Investigator: Demetris Mouzouris		

The following is an exhaustive and complete list of all the data that will be collected (through questionnaires, interviews, extraction from records, etc) [The required data will be the Readings from the Sensor Device, and the consent of the user to participate in the test]

The data is relevant to the study purposes because they will enable the project to have a diverse and variety of data from different users that will then be utilised in order to improve the application's algorithm and also to see if there are any kind of bugs that will require us to fix. This will also enable us to identify how good can the algorithm cope into various scenarios that will be designed and tested in order to make sure the sensor and the application are collecting the appropriate data and also to output the correct results. The data is adequate because the data that will be required from the participants will be attached to fake profile in order to keep the data anonymised, therefore the only data required will be the data from the sensor and the consent of the user to take participation in the experiment and the data is not excessive because the data that will be collected are the minimum requirements in order for this experiment to take place.

The data will be processed fairly because the participants will have given anonymised consent.

The data's accuracy is ensured because prior to the test beginning the data will be generated and will have been cross-checked between a secondary individual which will assist with the experiment.

Data will be stored on the cloud database Firebase a google NoSQL database as well as locally on the handheld mobile devices. That will not be a breach of the policy due to the data being anonymised. The data will be held in accordance with the University policy on data retention.

Data files will be protected by Firebase Encryption Algorithms; Handheld devices data will be protected by a PIN that is utilised by the handheld device and by a password in order to gain access to the file.

The data will be destroyed by Demetris Mouzouris at 12:00 pm on the 5th of September 2019 through Deletion and removal of the database stores.

The data will be anonymised by Demetris Mouzouris. Consent forms will be linked to the data by Demetris Mouzouris

The data will be processed in accordance with the rights of the participants because they will have the right to access, correct, and/or withdraw their data at any time and for any reason. Participants will be able to exercise their rights by contacting the investigator (e-mail: dm1u18@soton.ac.uk) or the project supervisor (e-mail: nmw@ecs.soton.ac.uk).

Data Protection Act 2018 (DPA) best practice

If the study involves personal or sensitive data, explicitly explain how data will be collected, stored, analysed, held securely, and in turn destroyed. The DPA does not apply to anonymous data and a DPA Plan is not required in the case of such data.

The principles of the DPA are that personal data must be:

1. Processed fairly and lawfully.
2. Processed for specified purposes and in an appropriate way.
3. Adequate, relevant and not excessive for the purposes.
4. Accurate and up-to-date.
5. Not kept for longer than necessary.
6. Processed in accordance with the rights of data subjects (participants).
7. Protected by appropriate security, both practical and organisational.
8. Not transferred outside the European Economic Area (EEA) without adequate data protection controls.

Data is recorded information, whether stored electronically on a computer or in paper-based filing systems. Personal data is information about an identifiable living individual. It can be factual, such as the date of a person's interview, or an opinion, such someone's view on how the person has performed on a task. It obviously includes individuals' contact addresses or telephone numbers. (Less obviously, note that personal data is being processed where information is collected and analysed with the intention of distinguishing one individual from another and to take a particular action in respect of an individual. This can take place even if no obvious identifiers, such as names or addresses, are held.) Processing is any activity that involves data, including collecting, recording or retrieving, using, disclosing, organising, adapting, changing, updating, or destroying it.

The DPA identifies Sensitive Personal Data as:

a) the racial or ethnic origin of the data subject (participant);
b) his political opinions;
c) his religious beliefs or other beliefs of a similar nature;
d) whether he is a member of a trade union;
e) his physical or mental health or condition;
f) his sexual life;
g) the commission or alleged commission by him of any offence or
h) any proceedings for any offence committed or alleged to have been committed by him, the disposal of such proceedings and the sentence of court in such proceedings.

The processing of sensitive data must meet at least one of the 10 stricter conditions laid down in Schedule 3 of the DPA. It may be useful to know that condition 1 of this schedule permits processing of such data if the data subject has given his explicit consent, and condition 5 if the information has been made public as a result of steps deliberately taken by the data subject.

Keep in mind that the Police have a right of access to personal data held by the study for the purpose of safeguarding national security; preventing or detecting crime; prosecuting or apprehending offenders; assessing or collecting tax; or protecting the vital interests of the data subject or another.

Researchers are exempted: from the second data protection principle, meaning that personal data can be processed for purposes other than for which they were originally obtained; from the fifth data protection principle, meaning that personal data can be held indefinitely; and from the data subject's right of access to his personal data provided the data is processed for research purposes and the results do not identify data subjects. In addition, the Data Protection (Processing of Sensitive Personal Data) Order 2000 para.9 provides that processing in the course of maintaining archives for research purposes is permissible where the sensitive personal data are not used to take decisions about any person without their consent and no substantial damage or distress is caused to any person by the keeping of those data. These exemptions do NOT give a blanket exemption from all the Data Protection Principles to data provided and/or used for research purposes. Researchers wishing to use personal data should be aware that the Data Protection Principles still generally apply, notably the requirement to keep data secure1.

A study may seek to anonymise the data it keeps. Anonymisation involves the removal of participants' personal information (names; e-mail address; whatever data it is that might permit identification; etc) from the data such that what remains cannot be used to identify them. Note that audio and video recordings (and often transcriptions too) cannot easily be anonymised, so they should normally be treated as non-anonymous data. Anonymised data can usually be kept without security and can easily be passed to other investigators for specialist analysis.

The DPA requires access to be granted to participants to all of their data, if any part of that data allows their identification. If the data has been anonymised, two issues arise.

1. If the personal information has been removed from the data AND DESTROYED, then the DPA is no longer applicable, and the data can be kept without security. However, investigators should note that they will be unable to follow up or subsequently contact participants in any way, or associate individuals with particular data, and should not attempt to suggest they might do so.

2. If the personal information has been removed from the bulk of the data, but NOT destroyed (ie, is kept separately), then the DPA remains applicable. In this situation, the personal information needs to be (a) kept both separately and securely from the

[1] http://www.jisc.ac.uk/publications/generalpublications/2001/pub_dpacop_0101.aspx

anonymised data, and (b) to be linked or 'keyed' to the anonymised data, such keys to be similarly kept securely (and often kept with the personal information).

If personal data is collected, in the 'Participant Information', inform the participant of:

· the processes the study will take to ensure data security;

· their right to access and correct their data and their right to request removal of their data;

· the authority which will give them access to their data (provide the contact information).

If sensitive data is collected, or the study involves clinical studies, human tissue samples, invasive procedures, or young or vulnerable people, provide additional detail. In the 'Participant Information', inform the participant of:

· the separation of identifying data and the anonymisation process;

· the method of linking the consent form (if any) to the participant's data;

· the processes for the destruction of all study data (if appropriate).

The study should conform to the University policy on data management applicable:

http://www.southampton.ac.uk/library/research/researchdata/

Investigators may find the University's survey platform useful:

https://www.isurvey.soton.ac.uk/

Contacts

risethic@soton.ac.uk.

10.5 Risk Assessment

UNIVERSITY OF Southampton

RECORD OF RISK ASSESSMENT

Title of the risk assessment	Use of IoT to enhance rehabilitation monitoring in leg surgery patients
Date risk assessment carried out	in between 10/July/2019 - 05/September/2019
Describe the work being assessed	Movement monitoring, angle of the leg,ankle location, footsteps
Describe the location at which the work is being carried out	University of Southampton Highfield Campus
Where appropriate list the individuals doing the work and the dates/times when the work will be carried out	Demetris Mouzouris - in between 10/July/2019 - 05/September/2019
List any other generic or specific risk assessments or other documents that relate to this assessment-use hyperlinks if possible	
Name and post of risk assessor	
List the names and posts of those assisting in compiling this risk assessment	Demetris Mouzouris
Name, post and where required, signature of the responsible manager/supervisor approving the risk assessment	Researcher: Demetris Mouzouris, Supervisor: Professor Neil White
Reference number and version number of risk assessment	

Page 1 of 3 HS/UOS/FR/038/0

Assessment

Title of risk assessment	Use of IoT to enhance rehabilitation monitoring in leg surgery patients

Risk Acceptability	
1-3	Risk acceptable
4-6	Risk to be reduced if readily possible
7-14	Risk to be reduced if reasonably practicable
15-25	Risk unacceptable

Risk Matrix		Severity				
		very low	low	medium	high	very high
		1	2	3	4	5
Likelihood	certainty 5	5	10	15	20	25
	likely 4	4	8	12	16	20
	possible 3	3	6	9	12	15
	less likely 2	2	4	6	8	10
	improbable 1	1	2	3	4	5

ref	Task/Aspect of work	Hazard	Harm and how it could arise	Who could be affected?	Existing measures to control risk	Risk Factors: Overall Likelihood	Risk Factors: Overall Severity	Residual Risk score	Any changes or extra controls?
	Lifting the leg	Falling	Raising the leg and not providing apropriate support measures	The participant	Chairs and support poles will be provided to keep the individuals standing	2	1	2	no
	Sensor Reading, Lithium battery	Lithium leak	Physical damage to the device	The participant	Protecting cover on the senor	1	2	2	no

Post Risk Assessment Actions
Title of risk assessment Use of IoT to enhance rehabilitation monitoring in leg surgery patients

Have any of the specialist control measures listed below been identified as required during this risk assessment? - indicate yes or no - if yes then include details on the post assessment action list below.	**yes/no**
is any exposure monitoring required?	no
Is any occupational health monitoring required?	no
Are there any hazards or other factors that could affect pregnant or nursing mothers?	no

Is any specfic training required before people can carry out this work?	no
Site local safety induction	

Are any additional procedures or risk assessments required as a result of this asssessment?	no

	Post Assessment actions		
ref	**action**	**by whom**	**by when**

Chapter 11 References

[1] S. O. Madgwick, "An efficient orientation filter for inertial and," pp. 1-32, 2010.

[2] A. I. t. t. K. Filter, "Gary Bishop, Greg Welch," *SIGGRAPH,* pp. 1-81, 2001.

[3] T. H. J.-M. P. Robert Mahony, "Nonlinear Complementary Filters on the," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL, VOL. 53, NO. 5, JUNE 2008 1203,* pp. 1202-1216, 2008.

[4] C. Worsham and A. B. Jena, "Why Doctors Shouldn't Dismiss the Apple Watch's New ECG App," *Harvard Business Review,* pp. 1-6, 2018.

[5] NHS, "Hospital Admitted Patient Care Activity, 2017-18," NHS, 20 Sep 2018.

[6] M. Matthew Hoffman, "The Seven Most Common Sports Injuries," *What weekend warriors need to know about preventing and treating the seven most common sports injuries,* pp. 1-5, 2008.

[7] M. F. F. Benjamin Wedro, "What are the most common bones that are broken?," *What is a broken bone (fracture)?,* p. 1, 2019.

[8] A. Elsts, X. Fafoutis, P. Woznowski, E. Tonkin, G. Oikonomou, R. Piechocki and I. Craddock, "Enabling Healthcare in Smart Homes: The SPHERE IoT Network Infrastructure," *IEEE Communications Magazine ,* pp. 164 - 170, 2018.

[9] Microsoft, "Project Emma," Microsoft, 2017.

[10] E. Mohn, "Internet of Things," no. Salem Press Encyclopedia of Science, p. 2, 2018.

[11] A. Reisner, P. Shaltis, D. McCombie and H. Asada, "A Critical Appraisal of Opportunities for Wearable Medical Sensors," *Proceedings of the 26th Annual International Conference of the IEEE EMBS ,* pp. 2149-2152, 2004.

[12] J. Qi, P. Yang, G. Min, O. Amft, F. Dong and L. Xu, "Advanced internet of things for personalised healthcare," *Pervasive and Mobile Computing,* no. 41, pp. 132-149, 2017.

[13] A. Gatouillat, B. Massot, Y. Badr, E. Sejdic and C. Gehin, "Building IoT-EnabledWearable Medical Devices: An Application to a Wearable, Multiparametric, Cardiorespiratory Sensor," *In Proceedings of the 11th International Joint Conference on Biomedical Engineering Systems and Technologies (BIOSTEC 2018) - Volume 1: BIODEVICES, pages 109-118,* pp. 109-118, 2018.

[14] J. Rei, C. Brito and A. Sousa, "Assessment of an IoT Platform for Data Collection and Analysis for Medical Sensors," *2018 IEEE 4th International Conference on Collaboration and Internet Computing,* pp. 405-411, 2018.

[15] M. Shin, "Secure Remote Health Monitoring with Unreliable Mobile Devices," *Journal of Biomedicine and Biotechnology,* pp. 0-5, 2012.

[16] T. M. S. Sayeed, M. T. Rayhan and S. Chowdhury, "Bluetooth Low Energy (BLE) based portable medical sensor kit platform with cloud connectivity," 2018.

[17] A. J. H. R. V. Sebastian O.H. Madgwick, "Estimation of IMU and MARG orientation using a gradient descent algorithm," *2011 IEEE International Conference on Rehabilitation Robotics,* pp. 1-7, 2011.

[18] G. Documentation, "Motion sensors," p. 20, 2019.

[19] B. Drew, "Kalman Filtering," *Statistical Techniques in Robotics ,* p. 11, 2019.

[20] "Euler angles," Wikipedia, 17 08 2019. [Online]. Available: https://en.wikipedia.org/wiki/Euler_angles. [Accessed 19 08 2019].

[21] "Gimbal lock," Wikipedia, 29 06 2019. [Online]. Available: https://en.wikipedia.org/wiki/Gimbal_lock. [Accessed 19 08 2019].

[22] D. C. G. D. G. Daniela Berardia, "Reasoning on UML class diagrams," *Artificial Intelligence 168 70–118,* pp. 1-49, 2005.

[23] Google, "Cloud Firestore documentation," 2019.

[24] Google, "Usage and limits," 2019.

[25] Google, "Get Started with Firebase Authentication on Android," 2019.

[26] T. Lacey, "Tutorial: The Kalman Filter," pp. 133-141.

[27] Google, "Sensors Documentation," 2019.

[28] x.-i. Technologies, "Open source IMU and AHRS algorithms," x-io Technologies, 2012.

Bibliography

MEMS for automotive and aerospace applications, Kraft Michael, White Neil M. (2013)