FINAL YEAR PROJECT REPORT

PROJECT TITLE: IoT Based Personal Healthcare Monitoring and Recommendation System using AI Chatbot

"A dissertation submitted in partial fulfillment of the requirement of an Honours Degree in **Computer Science** at INTI International University under the management and supervision of Faculty of Information Technology."

I declare that this project is my own work, it has not been copied in part or in whole from any source except where duly acknowledged. As such, all uses of previously published works (from books, journals, internet, etc.) have been properly acknowledged within the report to an item in the references or bibliographies. I hereby submit my dissertation, dated ***23/07/23*** *for review and assessment.*

by

Student Name : Saurav Hazra

Student ID : I21019956

Passport No : T9263896

Project ID : FDSIT-INTI-IU-BCSI-JUN-2023-0016

ACKNOWLEDGEMENTS

I would like to express my sincere appreciation to INTI International University and these specific individuals for their contribution with the completion of this Final Year Project.

First and foremost, I am profoundly grateful to my supervisor, Ms. Sarasvathi A/P Nagalingham, for her unwavering support, guidance, and expertise throughout the entire project. Her insightful feedback, patience, and direction have been instrumental in shaping the course of this study.

I would also like to express my appreciation to Dr. Narinderjit Singh Sawaran Singh, the FYP I and FYP II class lecturer, for his valuable guidance and input throughout the project. His knowledge and expertise were appreciated.

Special thanks are due to Dr. Rajermani and one of my friends Taha Mohamed Taha for graciously agreeing to provide a laptop and their phone respectively. Without their support, the direction of this project would have gone astray, and it would actually not be possible to complete this project.

I cannot forget to express my deepest appreciation to all the interviewees that were contacted as well as my family and friends for their understanding throughout this challenging academic journey. Their support throughout, truly encouraged and provided motivation for the entire timeline.

This project would not have been possible without the contributions of all the individuals mentioned above. I am beyond thankful for their support and encouragement, which have been instrumental in the successful completion of this project.

ABOUT THE AUTHOR

The author Saurav Hazra is a 21-year-old currently pursuing their Bachelor of Computer Science (Hons) Software Engineering at INTI International University, Nilai, Malaysia.

The project idea lies in the awareness of the issues faced by regular people regarding healthcare. The author has observed that many patients struggle getting healthcare because of personal, financial, or even other reasons. The author is driven by a strong desire to make healthcare more accessible to more people at all times since the author themselves, has had previous personal problems regarding a lack of communication with doctors.

Through this project, the author aspires to provide everyone with a system where they can be self-dependent and in a much more convenient manner manage to keep their health in check. The goal is to ensure people live happy and prosperous lives.

Table of Contents

List Of Tables

List of Figures

Project Proposal

Title : IoT Based Personal Healthcare Monitoring and Recommendation System using AI Chatbot

Abstract:

Health is one of the most significant factors in having a long and enjoyable life. Every day people are diagnosed with illnesses and diseases that require specific attention and medication for treatment. The development of advanced healthcare systems for patients and others in the medical industry has become a great priority worldwide to increase the life expectancy of countries. Nowadays, more importance is given to identifying illnesses, diseases and their treatments as early as possible instead of developing systems for treatment only. Technologies like Artificial Intelligence and Internet of Things are increasingly being implemented in medical applications and systems to deliver effective healthcare to a greater population of people regularly. IoT provides people with accurately measured medical data while AI helps with understanding and explaining the data for the best forward approach. So, this project proposes a system that provides individuals the ability to detect and identify an illness or a health problem based on their current body vitals through a smart phone application. The application utilizes an AI chatbot and medical IoT sensors concurrently such that patients receive personalized medical solutions and validated sickness reasonings in a quicker and more accessible manner.

Project Description:

Introduction

Sustainable Development Goal 3 defines that good health and well-being of people are essential in a sustainable world (Das et al., 2021). The health and well-being of people has always been a massive priority in their lives. From their childhoods, humans visit hospitals and other medical facilities to consult for any health-related issue. Research indicates that 60% of patients visit doctors for simple and common health problems which can be solved by simple medication or treatment (Nethrasri et al., 2021). However, the lack of patient's knowledge about the health issue and the fear that the problem falls within the other 40%, makes individuals want to complete their treatment as soon and smoothly as possible. Hospitals, clinics and other facilities in the medical industry are

increasingly looking to exploit and use advancing technologies to treat their patients in a quick yet effective manner.

One such mechanism implemented is IoT. The Internet of Things is a technology that enables Internet connections and improvements on normal processed data through use of sensors and other computer architecture (Khan et al., 2022). From a medical standpoint, IoT systems are used to measure, monitor and collect different health parameters of a patient. Specialized sensors can record and store a person's heart rate, oxygen and glucose levels as well as body temperature in a quick and efficient manner, without the need for a professional doctor (Hossain et al., 2020). Because of the proposed IoT healthcare system, patients do not need to search, spend their expenses on doctors in particular and can check for themselves. Doctors and nurses focus on their patients with more quality treatment instead of serving larger quantities. However, the drawback with the system is that it is only good for recording and storing data. IoT architecture alone is not intelligent to provide further support to patients.

Nowadays, there is enormous growth in the field of Artificial Intelligence (AI). AI involves various subsets, including machine learning (ML), decision tables and trees, conventional neural networks etc.., that once implemented in the medical industry can improve large aspects of medical sciences (Manickam et al., 2022). Intelligent systems can be used to provide healthcare assistance by analyzing data such as body vitals and then making calculated decisions to identify any harmful diseases or irregular body conditions. Smart healthcare also can provide the capability of suggesting precautionary / protective measures against a specific issue found.

Hence, this project promotes the design and development of an intelligent healthcare application that works with sensor recorded vitals. The IoT architecture will provide detailed information to the smart algorithms, with the intention of receiving reasonable and accurate medical conclusions in an autonomous and quicker manner.

Literature Review

Several systems have been developed in the past few years to improve the general healthcare of patients in the world. As a result of the global Covid-19 pandemic, numerous doctors and hospitals became inaccessible. According to S. Khamitkar (2020), the necessity of having an IOT-based heart rate monitoring system became essential. With this system in place, elderly people and others with the risk of heart attacks could regularly checkup on themselves at ease. Similarly, Anushree et al. (2021) created a body temperature monitoring system using IoT sensors as a way of detecting hypothermia or hyperthermia for people at home. Reddy et al. (2020) as well as Jegatheesh et al. (2019) developed systems that combined the temperature and heart rate monitoring systems into one and deployed new architecture such as a glucose sensor and a pressure transducer to monitor high blood-pressure, diabetic patients. According to Jegatheesh et al. (2019), their system uses recorded data to evaluate and send reminders to diabetics regarding the next time users

are required to take their medication. Rahimi et al. (2021) also developed an IoT medication reminder system, but the difference is there is no real time monitoring of body vitals. This system is based on complete past records. Amin et al. (2020) also suggested another combined IoT-based monitoring system. This system differed by sounding an alert for the patients if any of their vitals deviated from normal ranges.

As seen from the discussed systems, IoT-based healthcare has seen a lot of growth in recent times. The same can be said for smart healthcare systems. Alfandi (2022) focused on producing an enhanced diabetes monitoring system. Not only does the system record blood glucose levels in a non-invasive manner, but the stored data is also applied to a Machine learning model for prediction of future issues. Nethrasri et al. (2021) proposed an AI-based healthcare assistant which delivered online doctor consultations. Based on the symptoms experienced in real life, patients would input these symptoms into the application interface and be recommended a list of specialist doctors to visit and communicate for appropriate treatment. In 2022, Mahveen & Patil developed an IoT Virtual Doctor Robot. The idea allowed a robot to move around an area, monitor the body vitals of a patient and report it to an appropriate doctor for further analysis (Mahveen & Patil, 2022). In a similar approach, Thale et al. (2020) proposed a smart medical assistant robot. Their system incorporated a voice detection system to a navigation robot, so the users can describe their symptoms. All the information is compiled and then delivered to the doctor. However, this system as well as the other studies discussed cannot accurately identify any current illness, disease, or other issues by itself. There is a lack of IoT monitoring and AI-based healthcare approaches that are able to recommend and identify a patient's current health problem and solutions from recorded data. One that combines both IoT and AI for this purpose, is not discussed at all.

Problem Statement

According to a WHO status report as of 2020, countries around the world have less than one professional physician per thousand people delivering required medical assistance (Hossain et al., 2020). Because of the rapidly increasing populations, there is a growing shortage in the availability of healthcare practitioners. Doctors prioritize significant issues first. Critical conditions and health issues require continuous monitoring but how are regular people supposed to know if the health problems they are facing are critical or not. Patients are uninformed and take precautionary measures based solely on assumptions when they do not have a checkup with a doctor. If people take the wrong medication or ignore the issue as insignificant, letting an illness grow, this only risks more severe health issues. Certain patients utilize smart applications that request users to enter symptoms they currently face - providing a prescription based on their input. But what if the individual cannot correctly describe the symptoms they face. What if there are unrecognized symptoms? In such situations, personal health monitoring systems come in handy. However, even the examination of body vitals is not enough. Even though people can identify concerning values, the system is still not smart

enough to provide information on the particular illness / disease they are suffering from. By addressing all these problems, this project plans to produce an improved healthcare solution for people i.e., achieving accurate diagnoses without the need for doctor consultations.

Problem Objectives

In order to achieve the goals of this project, it is necessary to:

1. To perform a detailed study on the important factors and body vitals that are essential to general healthcare.

2. To research different sensors and design an IoT based system that is capable of recording and reporting various body parameters of a patient automatically.

3. To develop a mobile application with an AI chatbot recommendation system, that displays and analyzes recorded information, detects hidden health issues and provides solutions for individual users.

4. To conduct testing to ensure the effectiveness and accuracy of the predicted illnesses and the recommended solutions.

Project Scope

The proposed software system is an internet-based mobile application. The healthcare application created via Android Studio will provide regular people the ability to conduct a health checkup themselves using the linked IoT architecture. Sensors that detect body temperature, heart rate, glucose and oxygen levels are programmed and connected together using Raspberry Pi to record all the body vitals of an individual. The collected information will be stored in an internet-based database through Google Firebase, allowing automated entries directly from the IoT sensors .

The app reports these individual health records in graphs and more importantly applies the dataset into a smart algorithm to evaluate any signs of an illness / disease. The smart algorithm, written using Python in PyCharm, compares each data item with the standard, healthy vital ranges and then observes the potential links, key terms, relationships in values to decide and indicate any medical issues. In the unfortunate situation where a sickness or condition is estimated, the application utilizes the chatbot component designed through API software to prompt the user for any further symptoms experienced, or simply displays the predicted issue while also identifying the possible precautionary medication or immediate procedures to follow for temporary or permanent improvement.

Project Limitation

Since the project only has a duration of 19-21 weeks, the healthcare system is designed to work only as a mobile application. Users will not be able to access the interface through their personal computers as a web application or website. Another similar limitation with this system is that the application is only compatible with Android smartphones. The mobile app is not built as a cross-platform interface and cannot operate on IOS devices or other platforms. The application only functions when it is fed information from the IoT sensors. If the internet is cut off at any point of time, the entire system is rendered useless with no data being recorded. IoT architecture is extremely fragile. All the IoT components need to be fitted perfectly and require high maintenance. Any damage to the hardware or wiring can produce inaccurate results and be quite expensive to fix or replace. Finally, the system is not applicable for children's use. Since there is application of sensors, and other electronic equipment, it might be harmful and dangerous to expose most of the devices' radiation to minors.

Research Methodology:

This project will be conducted in three separate phases :

1. Phase 1 : Requirements Gathering

The project begins by collecting and analyzing valuable information. The primary research of the proposed healthcare system will be in a qualitative manner. To get an in-depth understanding of health parameters, symptoms and medical solutions, a face-to-face interview with hospital patients and specialized doctors will be conducted. The interviews will also focus on gathering information on monitoring times, hospital / doctor visitations. So, observing and questioning elderly people, nurses, caretakers etc., will prove to be useful. To provide more supporting research, medical datasets and journal articles found through internet libraries will be analyzed. The working of medical IoT sensors will also be studied through online publications.

2. Phase 2 : Agile Development using Extreme Programming

This project will use an agile model for the development of the proposed healthcare system. The system and its specifications are designed and iteratively updated using standard UML diagrams - Use-case, activity, entity and class diagrams. The IoT architecture will also be displayed in a block diagram, detailing the sensors and other components will be connected and programmed using Raspberry Pi. These diagrams illustrate the basic flow and structure of the system but are not descriptive enough to model the recommendation component. Smart algorithms will be illustrated using decision tables and trees and will be programmed with a rule-based system. Using the XP (Extreme Programming) framework of the agile model, running prototypes

of the designed diagrams will be rapidly programmed and tested to improve and alter the system based on the needs of users.

3. Phase 3 : Testing & Evaluation

Since this project uses XP agile methodology, this phase of the project works concurrently with the previous phase. The project requires different types of testing – white box and black box testing. All the functions of the application will be individually tested to ensure they work as intended. Each sensor will be evaluated to ensure they record the accurate value correctly into the database by visualizing the reports and graphs. The predictions of the medical illnesses and recommended solutions can be tested one by one by means of forward or backward chaining . After unit testing, integration and user acceptance testing will follow. The entire application is tested to check if all the modules of the system work together, and the non-functional requirements of the system are satisfied. If these conditions are met, the system will then be deployed for testing by different users to find any remaining problems or faults that need further corrections.

Target Audience:

The project is targeted at all kinds of adults. The personal healthcare system allows the monitoring and health examination of an individual anytime at their homes. Individuals like elderly people, who need constant monitoring can greatly benefit from using this system to identify any health problems before they build up. It can be a very expensive and time-consuming process to visit hospitals for checkups and treatment. Thus, this system can be especially useful to those who do not have access to doctors in their locality. Hospitals and clinics are still critical for medication and confirmation from doctors. So, medical facilities like these can also implement this system to ease the lives of doctors and nurses.

CHAPTER 1: INTRODUCTION

1.0 Overview

Health is one of the most significant factors in having a long and enjoyable life. The health and well-being of people has always been a massive priority in their lives. From their childhoods, humans visit hospitals and other medical facilities to consult for any health-related issue. Every day people are diagnosed with illnesses and diseases that require specific attention and medication for treatment. However, all these prescriptions are completely dependent on a medical professional. Instead of just focusing on advancing treatment of illnesses and diseases, equal importance should be given to developing systems that automatically identify health issues and recommend valid prescriptive treatments / medication as early as possible. As a result, the need for continuous monitoring of an individual's body vitals using Internet of Things (IoT) technology becomes a necessity. However, the drawback with using these sensors is that it is only good for recording and storing data. Hence, the author proposes a seamless solution that combines IoT with Artificial Intelligence (AI). This project promotes the design and development of a personal healthcare application that utilizes an AI chatbot with the sensor recorded body vitals. The chatbot technique prompts the IoT architecture to record and pass the users' collected real time health information to rule based AI algorithms which make correct and accurate decisions. The application retrieves and presents any identified / predicted health problem and its corresponding solution via the chatbot interface again. Thus, delivering reasonable and accurate medical conclusions in an autonomous and quicker manner. This chapter discusses the proposed project in detail - reviewing the problems related to this topic all in all, specifying the project objectives, scope, limitations and finally addressing the project methodology and the target audience.

1.1 Problem Statement

Recent research indicates that 60% of patients visit doctors for simple and common health problems which can be solved by simple medication or treatment (Nethrasri et al., 2021). The lack of a patient's knowledge about the health issue and the fear that the problem falls within the other 40%, incites individuals to repetitively return to hospitals and trusted doctors for treatment. A large number of people end up being unable to have a checkup with a doctor, be it because of financial reasons, unavailability of the doctors or the hassle of meeting them. They end up taking precautionary measures based solely on assumptions. If people take the wrong medication or ignore the issue as insignificant, letting an illness grow, this only risks further severe health issues.

In such situations, personal health monitoring systems come in handy. From a medical standpoint, IoT systems are used to measure, monitor and collect and record a person's heart rate, oxygen, glucose levels as well as body temperature in a quick and efficient manner. However, even the examination of body vitals is not enough. Even though people can identify concerning values, the system is still not smart enough to provide information on the particular

illness / disease a person is suffering from. To avoid these problems, certain patients utilize growing medical applications. There are helpful applications that request users to enter symptoms they currently face - providing a prescription or recommending the nearest specialist based on their input. But, if the individual cannot correctly describe the symptoms they face or if there are unrecognized or coinciding symptoms, these systems produce faulty results which in the medical industry potentially could be catastrophic. By addressing all these problems, this project plans to produce an improved healthcare solution for people i.e., achieving accurate diagnoses without the need for doctor consultations.

1.2 Project Objectives

In order to achieve the goals of this project, it is necessary to :

1. To perform a detailed study on the important factors and body vitals that are essential to general healthcare.

2. To research different sensors and design an IoT based system that is capable of recording and reporting various body parameters of a patient automatically.

3. To develop a mobile application with an AI chatbot recommendation system, that displays and analyzes recorded information, detects hidden health issues, and provides solutions for individual users.

4. To conduct testing to ensure the effectiveness and accuracy of the predicted illnesses and the recommended solutions.

1.3 Project Scope

The proposed software system is an internet-based mobile application. The healthcare application created via Android Studio will provide regular people with the ability to conduct a health checkup themselves using a medical AI chatbot and the linked IoT architecture. Sensors that detect body temperature, blood pressure, heart rate, glucose and oxygen levels are programmed and connected together using Raspberry Pi to record all the body vitals of an individual. The collected information will be stored in an internet-based database through Google Firebase, allowing automated entries directly from the IoT sensors .

The app reports these individual health records in graphs and tables and more importantly applies the dataset into a smart algorithm to evaluate any signs of an illness / disease. The smart rule-based algorithm, written using Python in PyCharm, compares each data item with the standard, healthy vital ranges and then observes the potential links, key terms, relationships in values to decide and indicate any medical issues. In the unfortunate situation where a sickness or condition is estimated, the application utilizes the chatbot component to prompt the user for any further symptoms experienced, or simply displays the

predicted issue while also identifying the possible precautionary medication or immediate procedures to follow for temporary or permanent improvement.

1.4 Project Limitation

The healthcare system is designed to work only as a mobile application, i.e., people will not be able to access the interface through their personal computers as a web application or website. Another unfortunate limitation with this system is that the application is only compatible with Android smartphones. The mobile app is not built as a cross-platform interface and cannot operate on IOS devices or other platforms.

The application only functions when it is fed information from the IoT sensors. If the internet is cut off at any point of time, the entire system is rendered useless with no recordings being saved. IoT architecture is extremely fragile. All the IoT components need to be fitted precisely and require high maintenance. Any damage to the hardware can produce inaccurate results and be quite expensive to fix or replace.

Finally, the proposed healthcare system is not applicable for children's use. Since there is application of sensors, and other electronic equipment, it might be harmful and dangerous to expose most of the devices' radiation to minors. Other people who might not be able to use the application are people who are not language proficient in English as the whole app is developed and designed in English.

1.5 Research Methodology

This project will be conducted in three separate phases :

Phase 1 : Requirements Gathering

The primary research of the proposed healthcare system will be in a qualitative manner. To get an in-depth understanding of health parameters, symptoms and medical solutions, a face-to-face interview with hospital patients and specialized doctors will be conducted. Observing and questioning elderly people, nurses, caretakers etc., will prove to be useful to gather information on monitoring times, hospital / doctor visits. As part of secondary research, medical datasets and journal articles found through internet libraries will be analyzed and the working of medical IoT sensors will also be studied through online publications.

Phase 2 : Agile Development using Extreme Programming

The proposed healthcare application and its flow are designed and iteratively updated using standard UML diagrams - Use-case, activity, and class diagrams. The IoT architecture will be illustrated in a block diagram, detailing the sensors and other components that will be connected and programmed using Raspberry Pi. Additionally, AI algorithms will be explained using decision tables and trees. These medical rules are the core that will be programmed to

develop the recommendation component. Using the XP (Extreme Programming) framework of the agile model, running prototypes of the designed diagrams will be rapidly programmed and tested to improve and alter the system based on the needs of users.

Phase 3 : Testing & Evaluation

Since this project uses XP agile methodology, this phase of the project works concurrently with the previous phase. The project requires different types of testing – white box and black box testing. All the functions of the application will be individually tested to ensure they work as intended. Each sensor will be evaluated to ensure they record the accurate value by visualizing the reports and graphs. The predictions of medical illnesses and recommended solutions via the chatbot can be tested one by one by means of forward or backward chaining. The entire application system will then be tested to ensure all the modules work collaboratively before being deployed for final testing by different target users to find any remaining problems or faults that need further corrections.

1.6 Target Audience

The project is targeted at all kinds of adults. The personal healthcare system allows the monitoring and health examination of an individual anytime at their homes. Individuals like elderly people, who need constant monitoring can greatly benefit from using this system to identify any health problems before they build up. It can be a very expensive and time-consuming process to visit hospitals for checkups and treatment. Thus, this system can be especially useful to those who do not have access to doctors in their locality. Hospitals and clinics are still critical for medication and confirmation from doctors. So, medical facilities like these can also implement this system to ease the lives of doctors and nurses.

1.7 Summary

To summarize, this chapter proposes a new healthcare system that uses IoT technology to monitor an individual's body vitals and combine it with AI chatbots to provide autonomous and quicker medical conclusions. The suggested project aims to develop a personal healthcare application that integrates a medical chatbot to identify health issues and recommend prescriptive treatments based on sensor recorded body vitals.

CHAPTER 2: LITERATURE REVIEW

2.0 Overview

This chapter covers the complete background study behind developing the proposed healthcare system. Since the aim of the proposed system is to improve the general health of individuals, the topic of healthcare and understanding the current standings and issues is widely looked into. Statistical figures, tables and graphs are studied to illustrate the importance of managing and maintaining good health standards. To follow, the importance of having improved technology like the Internet of Things (IoT) and Artificial Intelligence (AI) implement key features are highlighted under this chapter. Key features, being the ability to regularly monitor body vitals and potentially predict illnesses with their recommended solutions are signified by comparing existing works. Existing IoT monitoring and other smart healthcare systems that currently offer help will be reviewed and compared with the proposed system to analyze the strengths and shortcomings. After evaluating the similarities and differences in all the systems, the concept of the proposed system will be explained in depth. All the different hardware components needed to construct and program the IoT monitoring system as well as each software tool and programming requirement required to develop the complete mobile application will be defined. A complete description of the AI Chatbot technique and the concept and algorithms in identifying possible health issues with their recommended solution will be detailed.

2.1 Good Health & Well Being

2.1.1 Definition

Health can be defined as a state of overall physical, mental, psychological and social well-being (The Scientific World, 2023). An individual having good health indicates that their entire body is safe from any kind of physical and internal disorders. A physically healthy person's functional and metabolic efficiency protects the membranes of each body part from external diseases while mental health is indicated by the ability to adapt and accomplish tasks in a satisfactory and successful manner (The Scientific World, 2023).

Good health is absolutely integral to human happiness and prosperity. For many a people, it is a general consensus that health should be a fundamental human right. A healthy person is able to carry out their own duties without being dependent on others around them. If a person starts suffering from an illness or health issue, they are unable to meet their individual needs by themselves and more than likely require help from others. Along with weakening an individual's body, it affects a person's psyche, where the presence of an illness brings upon a sense of helplessness and mental anguish. Diseases end up harming the resources of families and societies with individuals expending a large amount of their saved-up finances on treatment. Eventually, treatment of issues can become a huge hassle, only being resolved by meeting trusted doctors and nurses, i.e., going to hospitals on a frequent basis.

2.1.2 Healthcare Practice

One of United Nations' primary 17 Sustainable Development Goals, SDG 3 defines that good health and well-being of people at every stage of their lives are essential in a sustainable world (Das et al., 2021). The current state of health in numerous countries is extremely concerned with billions of people unable to access necessary medicines and millions of adults and children suffering from health problems that lead to eventual fatalities (Küfeoğlu, 2022). To correct remedies of this sort, governing bodies of small as well as large countries are seeking to implement SDG 3 and improve the health conditions of their citizens and residents. The main objective of SDG 3 is to enhance healthcare so that 40% of premature deaths are prevented in all the nations across the globe (United Nations, 2021). To achieve this goal, countries are providing more focus on improving sanitation and hygiene, financing healthcare systems and most significantly providing greater access to doctors.

In recent years, the COVID-19 pandemic demonstrated how a particular disease might spread from a small group of people and become a health concern of international significance in a matter of days (Küfeoğlu, 2022). The importance of hospitals, clinics and medical personnel was greatly depicted during this tough period. Hospitals and their services prove to be central to the well-being of all populations. Today, hospitals, clinics and medical offices are where people go and meet certified doctors, nurses, and other practitioners to get diagnosed and treated for a health problem they might be experiencing. Physicians thoroughly check up on patients and provide them with justified and valid diagnoses as part of their consultations. Individuals are treated by specialists who are able to cater to all kinds of patient problems and are not limited to simply treating one kind of disease. People admitted to these medical facilities maybe suffering from the simplest general infections or be diagnosed with life-threatening conditions such as diabetes or cancer that require extreme care.

As measured from past global health records between 2000 and 2020, cardiovascular diseases, cancer, diabetes, infectious and finally respiratory diseases carry the highest risks of premature death. In 2019 alone, approximately 33.2 million people worldwide were unfortunately killed due to cancer, cardiovascular disease, diabetes, and chronic respiratory diseases which was a 28% increase when comparing the statistics of year 2000 (WHO, 2022). The COVID-19 pandemic only aggravated these numbers, adding more deaths, this time due to infectious viruses.

Due to these disastrous numbers, patients rightly build up large amounts of concern. So, when people face any kind of painful symptoms, without risking any future catastrophe, they schedule immediate visitations to doctors to receive immediate treatment. According to healthcare statistics from the Organization for Economic Co-operation and Development, it was analyzed that out of all the OECD countries as of 2021, South Korea had the highest rate of yearly visits to an individual doctor. With an average of 14.7 yearly visits with a doctor per capita (OECD, 2021), research states that the citizens of South

Korea end up consulting their doctors considerably more than other countries such as Japan, Australia, Brazil etc. As compared to South Korea, the citizens of Brazil only have an average of 1.6 yearly visits with a doctor per capita. Thus, when we compare both the countries' mortality rates, South Korea considerably has a better mortality rate. Not only Brazil but all the other nations have a worse number as seen in Figure 2.1. Doctors, nurses, and all other medical practitioners, with the aid of their medical tools and techniques, are in charge of identifying specific problems and determining whether a single patient requires constant monitoring, surgery, or simple, straightforward medication. They control the level of care that citizens of a country receive and ensure that individuals live long healthy lives. These medical practitioners are imperative to a nation and define how strong the healthcare is in the particular country.

2.1.3 Problems with Current Healthcare

As mentioned before, today's current healthcare system is mainly dependent on the physicians that make up hospitals, clinics etc. But because of the rapidly increasing populations, there is a growing shortage in the availability of healthcare practitioners. According to a WHO status report as of 2020, countries around the world have less than one professional physician per thousand people delivering required medical assistance (Hossain et al., 2020). Hospitals around the globe are facing overcrowding issues with patients regularly swarming in, requesting doctors to prioritize care for themselves first. With an issue of this sort, doctors at a hospital or clinic may not be able to provide their best for each patient, distracted or taken away by another issue. Doctors prioritize significant issues first. Critical conditions require the utmost attention, but regular people are uninformed and only want their issue resolved first. These types of situations create a lot of chaos in the environment, distressing both the patients as well as doctors.

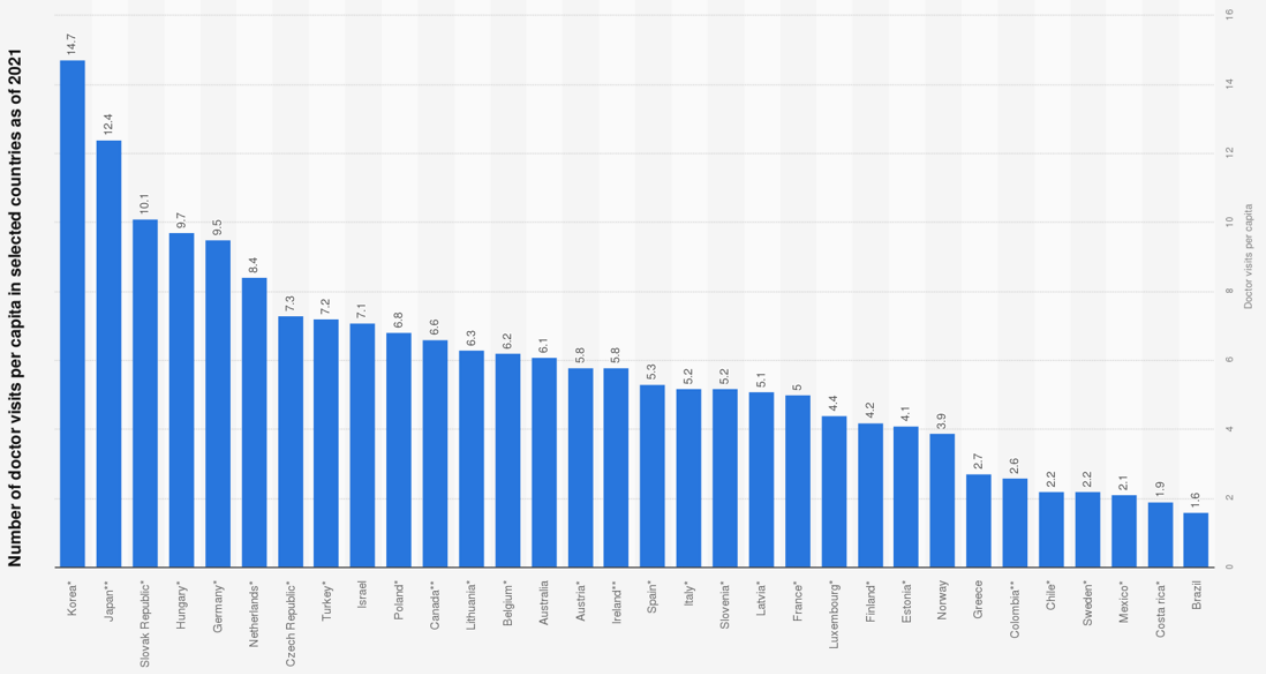


Figure 2.1 OECD Statistic for Number of Doctor Visitations per capita as of 2021 (OECD, 2021)

Even though it is true that the availability of doctors is limited, having more hospitals and clinics is still a great positive. There are still a large number of areas and countries where people have no access to medical resources. According to a report by the United Nations (2021), 75% of a country's health infrastructure is concentrated and based in urban areas. A large number of people living in rural societies cannot afford to visit hospitals due to the income inequality in the country. The expenses and time needed for travel and then having a checkup with potential expenditure for treatment and medication is way too expensive and too much of a hassle. So, in a country like India where 72% of the country's population actually live in rural areas, majority of people suffer without ever receiving proper healthcare (Basu, 2022) and rely on their own judgement for a prosperous, long life.

2.2 Managing and Maintaining Personal Health

2.2.1 Importance of Monitoring Health

Regular people have a tremendous amount of reliance on medical professionals in daily life whether they acknowledge it or not. Without their presence, there are numerous health-related issues and questions that arise, especially among those who require frequent communication with the doctors. Elder citizens and young children are more likely to pick up diseases and illnesses because of their weaker bodies and thus require continuous monitoring of their body vitals. It is imperative that these individuals and medical patients suffering from cancer, cardiac issues, diabetes etc., have regular health checkups to keep them in their best possible states.

SDG 3, promoting good health and well-being, states that healthcare infrastructures should give a high priority to identify, reduce and manage health risks in an efficient manner instead of just improving and enhancing medical treatment (UN Office for Outer Space Affairs, 2021). Doctors in hospitals and clinics make sure that their patients regularly come in for scheduled checkups, managing and aiding them on a personal basis for their betterment. However, as seen before, professional medical care is not available at all times and places. It is not the greatest ideal to be completely dependent on medical professionals for healthcare. Every person in the world should make an effort to maintain a good health standard by monitoring their own body vitals. Recent advances in technology have made it possible to monitor an individual's health parameters in a more easy and efficient manner (Humayun et al., 2022). Most smartphones now provide built in healthcare applications to allow their users to track and measure statistics such as calories burnt, steps walked, temperature, heart rate etc. Not only smartphones, but there has been an increase in wearable devices like smartwatches and Fitbits providing monitoring features to users.

Having alternative methods of monitoring health as shown in Figure 2.2 helps numerous people. Unless some sort of physical treatment is immediately required, people visiting doctors for simple consultations to get medicine prescriptions is just a massive inconvenience, especially for elder citizens who

need constant monitoring but might have difficulties with continuous long-distance travel (Humayun et al., 2022). Remote healthcare can be achieved in this manner with information from these devices directed to doctors, and then receiving appropriate medical feedback according to analysis of the data. These smart techniques of monitoring not only help patients but medical practitioners too. Glantz et al. (2019) stated that nurses regularly spend more time on medication and documentation preparation - an inefficient method of recording data with high potential of inaccuracy. The automated mechanism of health monitoring is slowly being deployed on a larger scale. Devices that fall under this category employ a growing technology that undoubtedly brings upon improved benefits. This technology is defined as the Internet of Things.

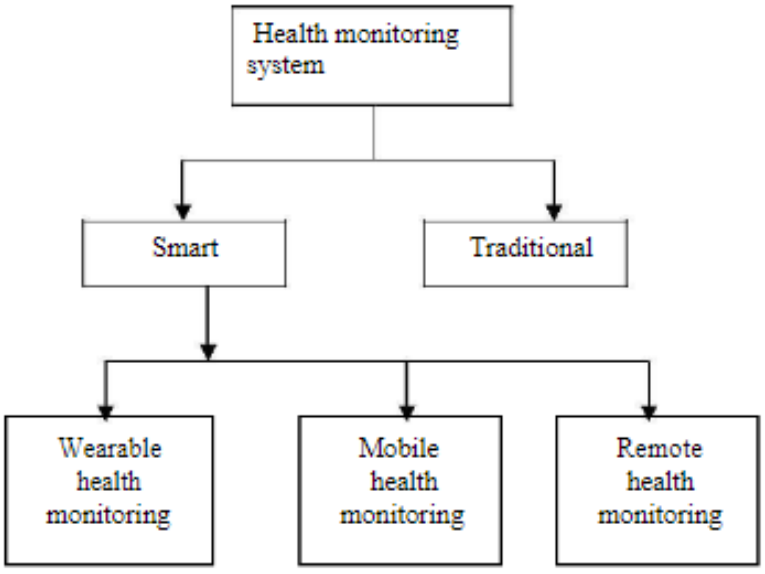


Figure 2.2 Types of Healthcare Monitoring Systems (Abdulmalek et al., 2022)

2.2.2 Internet of Things (IoT)

The Internet of Things is a term invented by Kevin Ashton in 1999 that refers to devices and technology enabled by Internet connections, providing improvements on normal processed data through use of sensors and other computer architecture (Khan et al., 2022). With the IoT paradigm, sensors collect data on various environmental or process-related variables such as temperature, pressure, volume etc., and connect the ‘things’ they measure - machinery, environments, or even human bodies themselves to a network of computers. IoT systems may also utilize devices called actuators in the case a certain measured condition is reached. Actuators take immediate, real-time actions if sensors detect events and situations that require a response. However, in all IoT processes the captured analog data is converted to digital information and eventually sent to the connected datacenter or cloud for storage via an Internet gateway (Abdulmalek et al., 2022).

Because of this method of record collection, controlling personnel can perform quicker and more frequent analysis of the data, taking appropriate actions in an improved manner with more precise data. The main goal of IoT technology is to track parameters accurately and quickly, thus the practical use of IoT in healthcare holds tremendous amount of potential. IoT monitoring provides a more accessible and cost-effective method of providing healthcare and people all across the globe have recognized these benefits. The global market for patient monitoring devices has been steadily improving as shown in Figure 2.3.

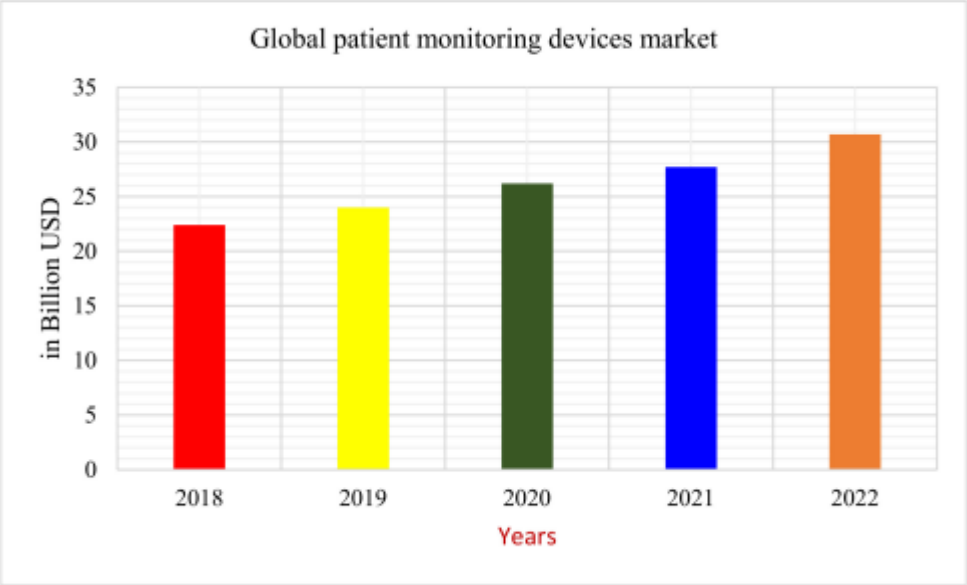


Figure 2.3 Global Market of Healthcare Monitoring Devices (Humayun et al., 2022)

From a more detailed standpoint, IoT healthcare systems are used to streamline necessary hospital or clinic processes - measuring and monitoring different health parameters of a patient. Specialized sensors record critical body vitals without the need for a professional doctor and deliver the information to them at their will (Hossain et al., 2020). These sensors are built into wearable or attachable devices through sensor nodes which control the communication and transmission of data with other devices. A standard sensor node comprises of a radio transceiver, processing unit and memory that processes the specific value of the body vital being measured. Technologies such as Bluetooth, infrared, RFID, Wi-Fi, or ZigBee, then enable the wireless communication and transmit the processed medical data to other devices over the Internet ensuring interoperability between all required devices and the related human (Abdulmalek et al., 2022). Figure 2.4 presents a visual depiction of how this process works.

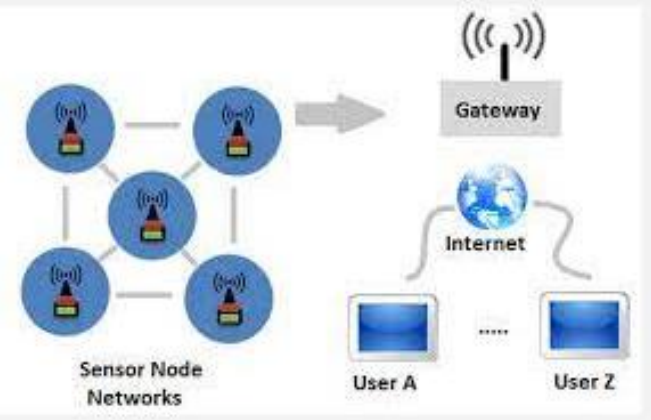


Figure 2.4 Working of Sensors in an IoT Architecture (Rakshit et al., 2022)

IoT monitoring systems are used in healthcare measure and monitor multiple different body parameters. Table 2.1 describes a few of the most important applications where IoT monitoring comes in handy.

Application		**Evaluation Metric(s)**	**Targeted Patients**
Heart Rate / Blood Pressure Monitoring	The heart is in charge of pumping blood to the entire body. To ensure that the heart is pumping properly, and proper volume of blood is flowing, systems monitor heart rate of people.	Heart Rate, ECG, Pulse Rate (Abdulmalek et al., 2022)	Elderly People Critical Patients People suffering from Cardiovascular Problems (Cardiac Arrest / Stroke)
Blood Glucose Monitoring	Monitoring is usually reserved for people who are diabetic patients to maintain their blood sugar levels and avoid extreme levels that causes harm.	Blood Glucose	Diabetic Patients
Body Temperature Monitoring	Body temperature is the first indication when something is wrong with the body (Khan et al., 2022. In order to measure the temperature of an individual, this type of monitoring is useful.	Body Temperature	People having Fever, Common Cold, Hypothermia, Hyperthermia etc. (Abdulmalek et al., 2022)

			Covid Patients
Blood Oxygen level / Saturation Monitoring	To keep an eye on an individual's oxygen levels and ensure that the human body's respiratory system is operating in a correct manner. (Rakshit et al., 2022)	SpO2, Pulse Rate	People having Breathing difficulties, Tiredness. People suffering from Cardiovascular Problems

Table 2.1 Applications of IoT Healthcare Monitoring

2.2.3 Comparison & Analysis of Existing IoT Healthcare Monitoring Systems

Several systems have been developed in the past few years to improve the general healthcare of patients in the world. Because of the global COVID-19 pandemic, S. Khamitkar (2020) believed the necessity of having an IoT-based heart rate monitoring system became essential, thus developed a system that measures and sends the heart rate to a mobile application for monitoring. This particular system utilized Arduino Uno with a pulse sensor and Bluetooth HC-05 module to record heart rate measurements on Android smartphones. The individual used a platform called Blynk to manage all the IoT architecture and processing to output the heart rate for its users.

Similarly, Anushree et al. (2021) created a body temperature monitoring system using IoT sensors as a way of detecting hypothermia or hyperthermia for people at home. The system designed connects an MLX90614 sensor to Raspberry Pi 3 Model B+, to measure the body temperature of an individual as displayed in Figure 2.5. To enhance the accuracy of measurement, the system only enables measurement of temperature when the Pi camera detects a face. The individuals' recordings are monitored and delivered through the user's personal email after connecting the Raspberry Pi with Simple Mail Transfer Protocol (SMTP).

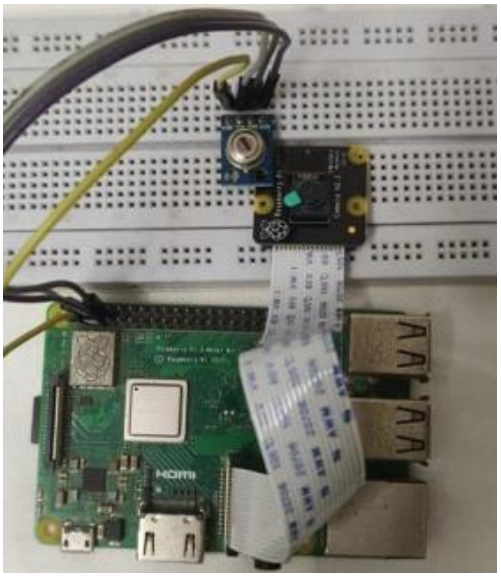


Figure 2.5 Temperature Monitoring System designed by Anushree et al.

Reddy et al. (2020) as well as Jegatheesh et al. (2019) developed systems that combined the temperature and heart rate monitoring systems into one and deployed new architecture to monitor specific patients. Reddy et al. (2020) constructed their system using sensors like the pulse sensor, LM35 temperature sensor, NIR glucose sensor and connected them to Arduino Uno to record glucose levels for diabetic patients. An addition to this certain system, is that there was an insulin infusion pump, shown in Figure 2.6, connected to the monitoring system to push insulin into the body in case the concentration of blood sugar was low.

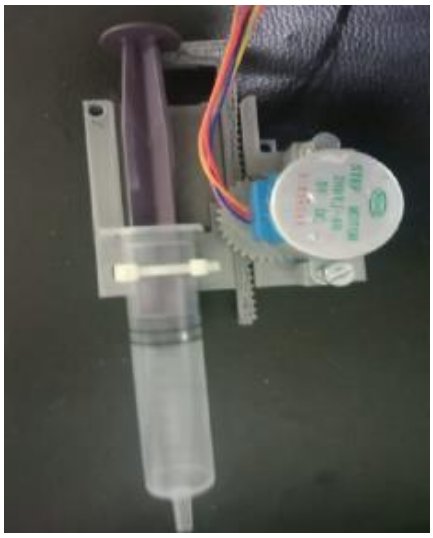


Figure 2.6 Insulin Infusion Pump built into Glucose Monitoring System.

According to Jegatheesh et al. (2019), their system uses recorded data to evaluate and send reminders to diabetics regarding the next time users are required to take their medication. They use the same sensors as the system developed by Reddy et al. (2020) and also add a pressure transducer to

measure pressure but main the difference with this monitoring system is that it uses Raspberry Pi 3 instead of Arduino UNO. The systems also differ in the way the recordings are output. The system made by Jegatheesh et al. (2019) monitors and delivers the recordings and reminders via SMS using the Global System for Mobile Communication (GSM module). Whereas the other system just displays the monitored values through an LCD display.

Amin et al. (2020) also suggested another combined IoT-based monitoring system. Their system uses an IR sensor to measure heart rate, LM35 sensor to measure temperature and GY521 sensor to detect the movement of a patient. All these sensors are connected to Arduino UNO and through the integrated Bluetooth HC-05 module connect to an Android application for use. This system also offered alerts for patients if any of their vitals deviated from normal ranges.

2.2.4 Summary of Existing IoT Healthcare Monitoring Systems

Authors w/ Reference	**Aim**	**Monitoring Technology**	**Output Protocol**	**Evaluation Metric(s)**
S. Khamitkar (2020)	Provide a heart rate monitoring system.	Pulse sensor, Bluetooth HC-05 module, Arduino UNO	Bluetooth sends data to Blynk mobile application	Heart Rate
Anushree et al. (2021)	Provide a body temperature monitoring system.	MLX90614 sensor, Raspberry Pi 3 B+	Monitored data is sent to emails via the SMTP server.	Body Temperature
Reddy et al. (2020)	Blood Glucose monitoring system for diabetic patients with insulin infusion.	LM35 sensor, Pulse sensor, NIR Glucose sensor, Arduino UNO	LCD display Insulin Pump	Blood Glucose level, Body Temperature, Heart Rate

	Monitor temperature, heart rate			
Jegatheesh et al. (2019)	Blood Glucose monitoring system for diabetic patients. Monitor temperature, heart rate, blood pressure	LM35 sensor, Pulse sensor, NIR Glucose sensor, transducer, Raspberry Pi 3 B+	Through the GSM module via an SMS message. LCD display	Blood Glucose level, Body Temperature, Pressure and Heart Rate,
Amin et al. (2020)	Monitor temperature, heart rate, blood pressure Monitor movement of an individual	LM35 sensor, GY521 sensor, IR sensor, Bluetooth HC-05 module, Arduino UNO	Bluetooth Module forwards information to an Android application	Body Temperature, Blood Pressure, Heart Rate,

Table 2.2 Summary of Existing IoT Monitoring Systems.

2.3 Improving Healthcare Quality

2.3.1 Identifying Health Issues & Recommending Solutions

The main problem with most people and patients is the question of knowing what someone is actually suffering from when they are unwell. If an individual is feeling ill, they may experience numerous symptoms and seek treatment and medicine based on assumption. But without communicating to a true professional, a person can never ever receive a definitive diagnosis of what is wrong. IoT most definitely detects, records and monitors health parameters in a better way as compared to the manual method. However, the drawback with the system is that it is only good for recording and storing data. IoT architecture alone is not intelligent to provide further support to patients.

In today's world it is imperative that the health industry focuses on finding methods to predict and identify problems rather than just focusing on improving treatment techniques. Early detection of health issues can also have broader public health benefits, such as preventing the spread of infectious diseases, thus reducing healthcare costs by preventing unnecessary hospitalizations, surgeries, and other costly interventions. Not to mention, identifying health issues early allows for a much better life for patients. Prompt intervention and early treatment can prevent the progression of the condition and potentially save lives which is especially crucial in cases of serious illnesses such as cancer (Manickam et al., 2022). It is important for healthcare practitioners,

policymakers, and researchers to continue exploring and implementing approaches to enhance early detection and intervention in healthcare. One such method which is increasingly growing is the application of Artificial Intelligence.

2.3.2 Artificial Intelligence (AI)

Nowadays, there is enormous growth in the field of Artificial Intelligence (AI). AI involves various subsets, including machine learning (ML), decision tables and trees, conventional neural networks etc.., that once implemented in the medical industry can improve large aspects of medical sciences (Manickam et al., 2022). Already in the medical industry AI has shown great potential with it being able to analyze medical images. AI algorithms can quickly and accurately analyze large amounts of medical images to identify abnormalities that may be missed by human radiologists, leading to earlier detection and treatment (Basu, 2022). Remote healthcare as discussed before with wearable watches, and other monitoring devices are usually operated with AI to gather locations and aid a user with recommendations on the steps to take to live healthy or maintain their lifestyle.

By applying specific subsets like machine learning and rule-based techniques, intelligent systems can be used to provide healthcare assistance by analyzing genetic information or chemical structures to improve the identification and research of medicinal drugs and their discovery (Basu, 2022). This can significantly reduce the time and cost associated with traditional drug discovery methods and lead to the development of more targeted and effective therapies (Manickam et al., 2022). In a secure and private manner, AI allows medical personnel to predict outbreaks or a patient deterioration / improvement over time, by analyzing data such as body vitals and then making calculated decisions to identify any harmful diseases or irregular body conditions. Smart healthcare also provides the capability of suggesting precautionary / protective measures against a specific issue based on the same analysis which are the major objectives of this proposed project.

2.3.3 Comparison & Analysis of Existing AI Healthcare Systems

Smart healthcare systems has seen a lot of growth in recent times. Alfandi (2022) focused on producing an enhanced diabetes monitoring system. Not only does the system record blood glucose levels using IoT technology in a non-invasive manner, but the recorded data is also applied to a Machine learning model for prediction of future issues. The monitoring system uses NIR glucose sensor to detect the level of glucose in the patient's body and outputs the recorded data via an LCD display and an SMS message using the GSM module placed on Arduino UNO. More importantly though, with each recording made by this system, the patient information – glucose level along with other factors such as current time, exercise activity etc., is appended to a dataset and fed into a Linear Regression and Decision Tree algorithm for training and testing of diabetes predictions.

Rahimi et al. (2021) developed a smart medication reminder system that is also able to detect and monitor whether a user has taken the medicine. This system does not record real time body vitals using the IoT architecture and is based completely on past medical records such as a medicine intake schedule, last time taken etc. IoT sensors in this system are connected to a medicine box and simply used to send a reminder via a buzzer and LED light as shown in Figure 2.7. The sensors can also detect whether the medicine is eventually taken. It uses a Passive Infrared (PIR) Sensor connected to Arduino UNO to detect motion of opening the medicine box and taking the medicine (Rahimi et al., 2021). When motion is detected, notifications are passed to the Blynk application.

Figure 2.7 Medicine Box for Medicine Intake Reminders (Rahimi et al., 2021).

Nethrasri et al. (2021) proposed an AI-based healthcare assistant which delivered online doctor consultations. Based on the symptoms experienced in real life, patients would input these symptoms into a healthcare AI chatbot interface and be recommended a list of specialist doctors to visit and communicate with for appropriate treatment. Figure 2.8 shows the chatbot interface developed by Nethrasri et al. (2021). The application is mainly coded through JavaScript and Python to provide a live chat and the ability to perform online video appointments and calls with recommended doctors. The healthcare application stores and receives the needed data from Firebase storage.

Figure 2.8 Chatbot Interface for Inputting Symptoms (Nethrasri et al., 2021)

In 2020, Thale et al. (2020) proposed a smart medical assistant robot. Their system was based on the Arduino Mega R3 with incorporated a few sensors

like a pulse sensor, MAX30105 oximeter and MLX90614 temperature sensor to detect body parameters. The architecture is enclosed within a well-designed robot, as shown in Figure 2.9, which has a motor driver for the locomotion and movement of the robot. The sensors are used to perform the preliminary health checkup, but the main feature of the system is the voice detection system to a navigation robot, so the users can describe their symptoms. This particular robot also had a connected monitor acting as the interface for text-based input and output but also allowed for voice-based communication using speech recognition technology with an AI chatbot. All the information is recorded, is compiled and then delivered to the doctor for future handling. In a similar approach, Mahveen & Patil (2022) developed a similar IoT Virtual Doctor Robot. The idea allowed the robot to move around an area and report the manually entered symptoms of a patient to an appropriate doctor for further analysis (Mahveen & Patil, 2022).

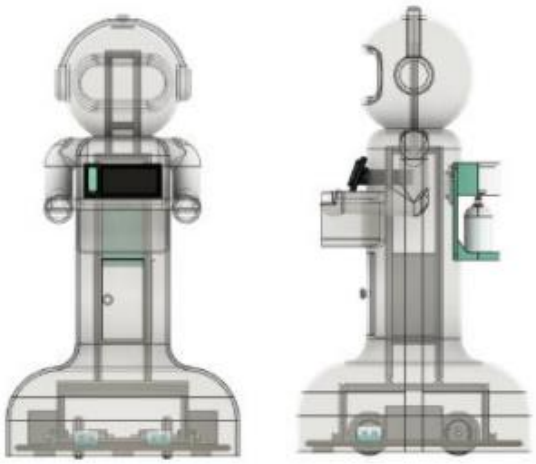


Figure 2.9 Casing of the Medical Assistant Robot (Thale et al., 2020)

2.3.4 Summary of Existing AI Healthcare Systems

Authors w/ Reference	**Aim**	**Technology Applied**	**Output Protocol**	**Evaluation Metric(s)**

Alfandi (2022)	Produces enhanced diabetes monitoring system that predict future issues using ML.	NIR Glucose sensor, transducer, Raspberry Pi 3 B+, ML dataset	Data sent via SMS messages to user mobiles.	Blood Glucose level Prediction Accuracy
Rahimi et al. (2021)	Provide a medication reminder system and identify whether medicine is taken	IR Sensor, Raspberry Pi 3 B+	LED light, buzzer	Body Glucose Concentration
Nethrasri et al. (2021)	AI based online healthcare assistance to consult specific doctors	AI Chatbot, Video Call API	Information is input delivered using a healthcare application	Accuracy of Recommended Doctor
Thale et al. (2020)	Conduct automated preliminary health checkup of body temperature, Forward data to professional doctors for analysis	AI Chatbot, MAX30105 oximeter, pulse sensor, MLX90614 temperature sensor	Through the GSM module via an SMS message. LCD display	Body Temperature, Heart Rate,
Mahveen & Patil (2022)	Conduct automated preliminary health checkup of body temperature, Forward data to professional doctors for analysis	AI Chatbot, pulse sensor, MLX90614 temperature sensor, transducer	Bluetooth Module forwards information to a doctor via Android application	Accuracy of recorded heart rate, pressure, temperature

Table 2.3 Summary of Existing AI Healthcare Systems.

2.4 Proposed System Development

Figure 2.10 is a visual representation of the overall proposed healthcare system. The diagram represents all the modules and individual components that make up the system and how they work to fulfill the objectives of the system.

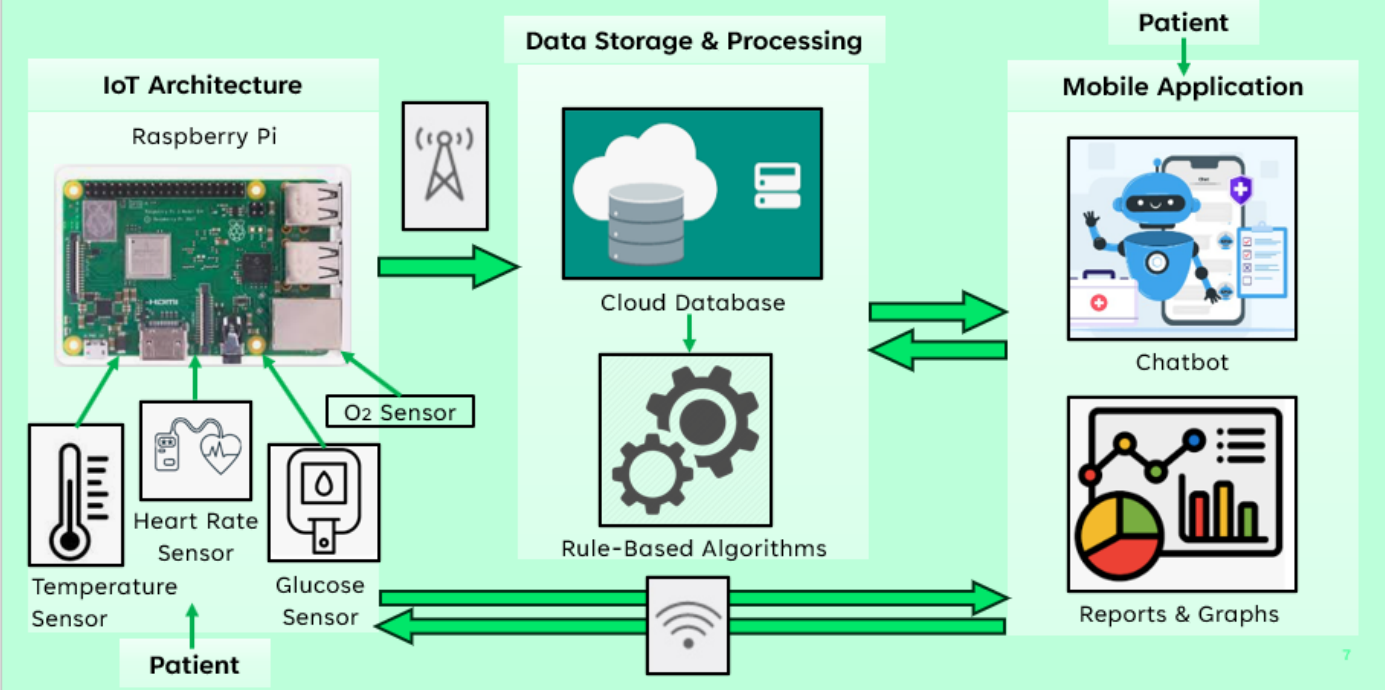


Figure 2.10 Visual Representation of the Overall Proposed System

2.4.1 IoT Architecture

The hardware listed in this section are the main components that make up the IoT architecture for the proposed healthcare system.

Raspberry Pi

Raspberry Pi is a small and affordable microcomputer that connects to other hardware peripherals such as sensors, monitors, an LCD display, and LED lights to work. When a hardware component like a sensor or monitor provides input to the system, data received by the microcomputer is processed according to the data collected and then output through other devices such as buzzers and LCD displays. With the use of programming languages like Python or Java, users can program and control how an IoT system is run at an instance depending on what information needs to be collected. Raspberry Pi is increasingly trending with tech industries and programmers (Ganesh, 2019). This is because the microcomputer devours less power and at a lower cost provides an efficient, portable and strong solution to everyday business as well as home applications (Jegatheesh et al., 2019). Moreover, the computer is equipped with all the features needed for communication with other devices – ZigBee wireless connectivity, Bluetooth etc. Although the functions of the IoT

system is written using Python or Java, Raspberry Pi actually operates on a Linux based operating system – the Raspbian OS. As displayed in Figure 2.10, the proposed IoT system will run using this hardware. However, it is important to note that the specific model is not certain as there might be compatibility issues with other components. The safest option so far seems to be the Raspberry Pi 3 Model B+ as shown in Figure 2.11.

Figure 2.11 Sample Raspberry Pi 3 Model B+ (Jegatheesh et al., 2019)

Temperature Sensor (MLX-90614 Sensor)

The temperature sensor is an input device that is connected to a computer like Raspberry Pi to quantify the temperature of an item. Temperature sensors can come in different varieties, but the proposed healthcare system utilizes the MLX-90614 IR sensor to record body temperature. The MLX-90614 is a small and low-cost sensor that works as a non-contact thermometer. This sensor measures values of temperature from -70°C, all the way to 380°C (Gsangaya et al., 2022) and works best when recording body temperature values at a short distance. The values are calibrated and measured with an accuracy of 0.1°C using this sensor when the infrared ray from the sensor is directed at the object within 2 to 5 cm (Gsangaya et al., 2022). The reason the MLX-90614 sensor is used for the healthcare system is that most other sensors record and provide the surrounding environment temperature. But for healthcare purposes, it is necessary to simply obtain the body temperature, without being affected by atmospheric humidity. Because of this, the sensor model displayed in Figure 2.12 is linked to the Raspberry Pi as part of the proposed IoT architecture.

Figure 2.12 Sample MLX-90614 IR Temperature Sensor (Gsangaya et al., 2022)

Glucose Sensor (NIR Glucose Sensor)

A glucose sensor is employed to detect the concentration of sugar level in the human body and present the data into the IoT architecture for processing. Glucose concentration is typically measured by pricking the finger of the patient which can cause a lot of pain and discomfort (Alfandi, 2022). Because of this, the proposed healthcare system applies a non-invasive glucose sensor as shown in Figure 2.13. The NIR Glucose Sensor uses an infrared LED light to detect the glucose levels without actually harming the skin. The infrared light with a wavelength of 900 - 950 nm hits one side of the finger and penetrates the skin tissue with an attenuated signal to be received by the photodiode on the other side. This process converts visual light into electrical voltage to collect and output the glucose readings (Alfandi, 2022).

Figure 2.13 Sample NIR Glucose Sensor (Alfandi, 2022)

Heart Rate Sensor (Pulse Sensor)

The pulse sensor displayed in Figure 2.14 is used to measure the heart rate of the user in Beats per Minute (BPM). The sensor structure is made of a photosensor and an LED that can produce light while trying to detect the variation of blood flow - caused by how strong the heart beats. Once a fingertip is placed on the device, the LED emits light towards the blood vessels whereas the photosensor detects the reflected light from the same vessels. The amount of light reflected is measured by the volume of blood in the arteries (Khamitkar, 2020). A high volume of blood absorbs more light creating a weaker reflection and this working principle of light reflection eventually defines the heart rate of a single person.

Figure 2.14 Sample Pulse Sensor (Jegatheesh et al., 2019)

Blood Oxygen Sensor (MAX30102 Sensor)

The blood oxygen saturation (SpO2 level) is a parameter that has rapidly changing values and need continuous monitoring. A pulse oximeter / blood oxygen sensor measures this body vital by directing a light emission through the fingertip – similar to that of a pulse sensor (Khan et al., 2022). By measuring the light intake of the blood vessels, the heart rate is measured when using a pulse senor. But, when using a sensor like MAX30102 as shown in Figure 2.15, the body vital that is measured is the SpO2 percentage.

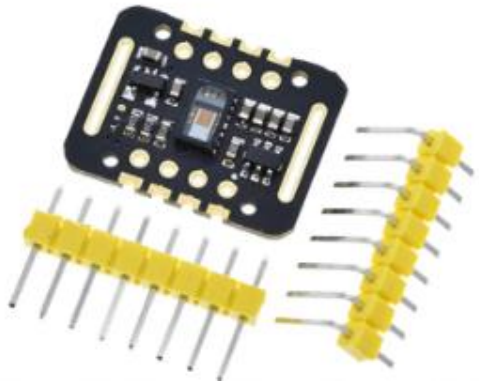


Figure 2.15 Sample MAX30102 Sensor

2.4.2 Mobile Application Development

As shown in Figure 2.10, a person interacts with the proposed system either using the IoT architecture or though the mobile application. The healthcare system is majorly dependent on the application. Thus, programming and design of the healthcare interface is critical. To make a simple and easy-to-use app that most people can access, the application is developed for Android smartphones. The IDE used to develop this app is Android Studio.

Android Studio

Android Studio is an IDE that specializes in developing Android applications. The software provides all kinds of Android SDK tools for designing, development, testing and debugging the programming of a particular app. The IDE uses programming languages like Java or Kotlin to code in variables, methods, classes and design the mobile display / layout screens (Uzayr, 2022). All the Java or Kotlin files for the program can be easily integrated with the layout files as part of one project application. After completing the entire project, the application can be exported and run as an Android Package (.APK file) in any Android compatible device.

In the proposed healthcare system, IoT sensors record the body vitals in real time, but the application is what people use to monitor the parameters. The data from the sensors is sent to a cloud storage and then eventually received by the application from the same cloud. The internet based storage used with the proposed system is Firebase.

Firebase

Google Firebase is a flexible and scalable internet-based cloud service that can be used with mobile application development. The software provides a real-time database which records and stores users' data synchronized across multiple devices (Omisola, 2021). This software is especially useful for recording data from IoT devices as they work with quickly updating information.

The application in the author's proposed system displays personalized medical information stored on the cloud database through visual graphs and reports for healthcare monitoring. Using the analysis and visualization of the monitored data, the application also provides users an interface for communicating any illness predictions and recommendations via the medical AI Chatbot.

2.4.3 Chatbot Recommendation

AI Chatbot

A chatbot is an intelligent computer program that can simulate a normal interaction with users using natural language. A chatbot is a typical example of smart Human-Computer Interaction (HCI) which applies the rules of Natural language Processing (NLP) so that computers can respond in the most natural and smart manner to human queries (Shinde et al., 2021). Users input their queries through a chat window to fire up the chatbot communication. The text entered is sent for pre-processing before being fed into an NLP model to generate a human-like response based on the processed, contextual information. An input sentence into the chatbot interface is converted into single words to locate and differentiate keys, root words which will help in delivering a more accurate and appropriate response. These processes are called tokenization and stemming (Shinde et al., 2021) as displayed in Figure 2.16. An NLP model is used to process the eventual key words, but the process of Natural Language Generation (NLG) prepares the response according to specific rule-based, retrieval or generative model.

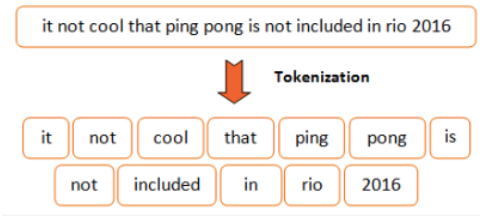


Figure 2.16 Pre-Processing Tokenization of Chatbot Text (Shinde et al., 2021)

Rule Based Learning

A rule-based system is a special type of expert system that consists of a set of 'if-then' rules that can be applied to make supportive or predictive decisions in real life applications. Rule based systems are constructed based on expert knowledge specific to one domain. Information for the system can either be collected from human experts, where the rules and relationships between attributes are traditionally engineered. Rules and conclusions are linked

together repeatedly to create a nested structure which can be traversed backwards or forwards to achieve a classification (Manickam, P. et al., 2022). The other approach is to build the rules via machine learning data. Using machine learning techniques, patterns and relationships can be found and labeled. Based on these patterns and pre labeled data, information can be classified with certified accuracy (Manickam, P. et al., 2022).

In the proposed healthcare system, the medical chatbot provides assistance to the user by providing them the ability to conduct specific or combined health checks. The chatbot also functions to prompt users for more information such as symptoms to identify illnesses and diseases from the monitored data and eventually find and recommend their solutions. Using the Python programming language and the PyCharm IDE, the chatbot accomplishes all the required functions by following a traditional a rule-based system and performing the forward chaining methodology to locate key terms and associated relationships to find the corresponding solutions.

CHAPTER 3: SYSTEM SPECIFICATION & DESIGN

3.0 Overview

The third chapter of this project is about defining the requirements and design of the proposed healthcare system. This chapter will discuss the methods used to collect the requirements of the system and then list down the functional and non-functional requirements to be implemented in the design of the system. The process of developing the healthcare system will then begin by designing the system diagrams as well as the interface diagrams so the implementation can be smoothly executed.

3.1 Requirements Gathering

The process of system specification and design begins by collecting the required information regarding the healthcare system. For this proposed system, critical medical information is collected from appropriate people and sources such that their needs and requirements are satisfied when developing the functionalities of the healthcare system. For this research, there are three different methods of data collection conducted – interviews, observation, and dataset analysis.

3.1.1 Interview

In order to achieve an in-depth analysis of medical information (Alfandi, 2022), the author decided to conduct interviews. Since the author is not proficient with medical knowledge, it was highly likely that the responses to the initial questions would require a clearer understanding. Interviews allowed the interviewer to ask questions immediately to better understand their response as well as provide the interviewee with the ability to explain in an unlimited way – something that was not possible with a questionnaire. Not only did the author get in touch with different medical practitioners such as doctors and nurses, patients, and regular people with or without medical troubles were also interviewed to gain a broad perspective. Since health differs according to various factors, people of different ages and genders were interviewed to collect the largest amount of valuable information possible. The author personally sent emails and messages to the all the interviewees requesting a few minutes of their busy schedules for the interview. The interviewees were gathered through personal connections with many of them working or living in external countries. As a result, interviews for the research were conducted both physically and online. The target of these interviews – are mainly nurses and doctors but also regular people to get all perspectives.

1. **Interview Questions**

Questions for Doctors & Medical Physicians

Question 1: What are the duties you carry out on a regular day as a medical practitioner?

Purpose: To understand how busy a single doctor can be and what activities they perform in a single working day.

Question 2: How frequently are patients recommended to come in for checkups? From a doctor's perspective, can some examinations be pointless, only resulting in a time hassle?

Purpose: To get a doctor's perspective on how frequently patients should come to visit them for medical monitoring and how important these visits are for the patient's health.

Question 3: What are the regular ranges for heart rate, oxygen saturation, body temperature, glucose level in a healthy body teenager / adult?

Purpose: To gather the body vital ranges in a healthy body.

Question 4: What kind of illnesses are most commonly found in patient checkups? Can these problems usually be resolved by simple medication?

Purpose: To discover the most probable and common health problems / diseases and identify whether these problems can be fixed by medicine and what their names are.

Question 5: How early is it possible to identify cancer / cardiac problems / diabetes / common illnesses in a body? What are the common symptoms / indications for these diseases?

Purpose: To gather information on how to identify major harmful diseases and how to recognize the basic indications that might prove a health problem in a person's body.

Question 6: What are the fluctuations in body parameters like temperature, blood oxygen / glucose level, heart rate when a person is suffering from these problems?

Purpose: To understand what happens with the body vitals in the scenario when a patient or person is suffering from a health issue.

Question 7: What other patient factors may be analyzed when identifying a health issue?

Purpose: To gather what other factors need to be analyzed before reaching a medical conclusion.

Question 8: What medication and treatment can be used to reduce the effects of these diseases (if effective)?

Purpose: To identify the names of the treatments and medication that are the best solution to a specific disease or health problem.

Questions for Nurses / Caretakers

Question 1: What are the duties you carry out on a regular day as a medical practitioner?

Purpose: To understand how busy a single nurse can get and what activities they perform in a single working day.

Question 2: How frequently are patients recommended to come in for checkups? From your perspective, can some examinations be pointless, only resulting in a time hassle?

Purpose: To get a nurse's perspective on how frequently patients should come visit the hospital / clinic for medical monitoring and how important these visits are for the patient's health.

Question 3: What are the regular ranges for heart rate, oxygen saturation, body temperature, glucose level in a healthy body teenager / adult?

Purpose: To gather the body vital ranges in a healthy body.

Question 4: What kind of illnesses are most commonly found in patient checkups? Can these problems usually be resolved by simple medication?

Purpose: To discover the most probable and common health problems / diseases and identify whether these problems can be fixed by medicine and what their names are.

Question 5: What are the fluctuations in body parameters like temperature, blood oxygen / glucose level, heart rate when a person is suffering from these problems?

Purpose: To understand what happens with the body vitals in the scenario when a patient or person is suffering from a health issue.

Question 6: Can these problems usually be resolved by simple medication? What other patient factors may be analyzed when conducting a preliminary checkup to identify a health issue?

Purpose: To gather all the factors that need to be analyzed in order to identify the names of the treatments and medication that are the best solution to a specific disease or health problem.

Question 7: What are the steps taken to constantly monitor and pay attention to your patients? Which type of patients require the most monitoring?

Purpose: To identify which potential health problems require the most monitoring and how often nurses are forced to monitor these types of patients. The question also gathers information on how nurses work to monitor patients.

Question 8: How difficult is it to continuously monitor the health conditions of patients? How much is the quality of monitoring affected by other nursing work?

Purpose: To gather how busy nurses can become and understand whether potential mistakes or problems can arise due to the business and commotion in medical facilities.

Questions for Patients / Regular People

Question 1: How frequently do you visit hospitals or clinics for health checkups?

Purpose: To evaluate how often patients need to monitor their health parameters.

Question 2: What is the most common illness, problem for which you visit hospitals and clinics? Is going / contacting a professional the first thought when you experience symptoms?

Purpose: To gauge how important doctors and medical professionals are in patient lives and for what reasons they visit the hospital.

Question 3: From your perspective, can some checkups end up being pointless, only resulting in a time hassle?

Purpose: To get a regular person's perspective on repetitive visits to the hospital / clinic for medical monitoring.

Question 4: Have you ever taken medication based on assumptions and avoided going to the hospital? Did this have a further negative affect?

Purpose: To analyze the potential harm that can be caused by taking medication that might be incorrect.

Question 5: What are the reasons you may not go to visit a clinic or hospital?

Purpose: To understand the potential difficulties of a person and the reasons as to why they might avoid professional help.

Question 6: Is it important to monitor your health constantly? How do you monitor your own health at home then?

Purpose: This particular question is necessary to gather what users think of maintaining health and how they might monitor their health when they cannot access a doctor or hospital.

2. List of Interviewees

List of Doctors & Medical Physicians

Name	**Hospital / Clinic Name**	**Job Title / Specification**	**Description**
Ms. Zainab Sharif (baniaz56@hotmail.com)	PNS Shifa Hospital Karachi, Pakistan	Medical Intern	Ms. Zainab Sharif is currently a medical intern gaining work experience and completing her studies to become an eventual oncologist.
Dr. Hamza Iftikhar (Hamza.iftikhar911@gmail.com)	PNS Shifa Hospital Karachi, Pakistan	House Officer	Dr. Hamza Iftikhar is a medical professional that has recently completed his basic studies to become a licensed general physician.
Dr. Lobna Mohamed Yahia	Tadawi General Hospital, Saudi Arabia	Pediatrician / Primary Care Physician	Dr. Lobna Mohamed Yahia is a medical practitioner with more than 15 years of experience as a pediatrician. She has a complete PhD and is a respected senior at her hospital.

Table 3.1 List of List of Doctors & Medical Physicians Interviewed

List of Nurses / Caretakers

Name	**Hospital / Clinic Name**	**Job Title / Specification**	**Description**

Ms. Angel Gimeno (baniaz56@hotmail.com)	Fatima University Hospital, Philippines	Student Nurse / Caretaker	Ms. Angel Gimeno is currently a student nurse gaining work experience and completing her studies.
Mr. Anselmo Revistual Jr. (anselmorevistual@gmail.com)	-	ER Nurse	Mr. Anselmo Revistual Jr. is a medical practitioner with more than 20 years of experience of being a nurse. He is not a retired nurse that worked majorly as an ER nurse.

Table 3.2 List of List of Nurses / Caretakers Interviewed

List of Patients / Regular People

Name	**Age**	**Current Illness (If Any)**	**Description**
Mr. Dilon Fernando (baniaz56@hotmail.com)	24	Hyperthyroidism and Gout	Mr. Dilon Fernando is primarily a student that is completing his bachelor's degree in computing. However, he also works part-time as a data administrator.
Mrs. Janaki Fernando (Did not consent to give email)	55	Thyroid, High Blood Pressure	Mrs. Janaki Fernando is a stay-at-home wife and mother of three children. She used to work as a business analyst but has decided to retire.
Mrs. Rubina Khatoon	47	Type 2 Diabetes	Mrs. Rubina Khatoon is a high school principal in her home state. She

(Did not consent to give email)			works a full-time job, completing all her daily working hours.
Mr. Dane Francis Dasalla (dasallafrancis@gmail.com)	21	-	Mr. Dane Francis Dasalla is a third-year student that is completing his bachelor's degree in medical technology.
Ms. Surbhi Hazra (hazrasurbhi@gmail.com)	17	Anemia	Ms. Surbhi Hazra is a high school student completing her final years of A-Level education.

Table 3.3 List of Patients / Regular People Interviewed

3. Analysis of Answers Provided by Interviewees

Answers Provided by Doctors & Medical Physicians

Question 1: What are the duties you carry out on a regular day as a medical practitioner?	
Answer	
Ms. Zainab Sharif	Take morning, evening patient and event notes. General examination of vitals, solving active complaints and counselling
Dr. Hamza Iftikhar	Historic evaluation, in-patient care and management, OPD duties and emergency care.
Dr. Lobna Mohamed Yahia	Performing examinations and assessments for patients , providing consultations and advice, prescribing

	medication, and referring patients to specialists when necessary.
Result Summary	The responsibilities of a doctor / physician is understood.

Question 2: How frequently are patients recommended to come in for checkups? From a doctor's perspective, can some examinations be pointless, only resulting in a time hassle?	
Answer	
Ms. Zainab Sharif	Every 2 weeks. Yes some examinations can be a waste which is why is why a general physical examination is always compulsory
Dr. Hamza Iftikhar	Patients are often called to come in for checkups taking into account their medical and family history. They are examined not only to figure out what's wrong but also to regularly check whether the rest is all normal. Every examination is useful to optimize the care of a person.
Dr. Lobna Mohamed Yahia	While opinions vary, routine physical exams are generally recommended once a year if you're over the age of 50, and once every 3 years if you're younger than 50 and in good health
Result Summary	All the interviewees highlight that having health checkups is necessary and they should be done even though some might not be fruitful.

Question 3: What are the regular ranges for heart rate, oxygen saturation, body temperature, glucose level in a healthy body teenager / adult?	
Answer	
Ms. Zainab Sharif	HR: 60-100bpm, SpO2: 95 to 98%, body temp: 98F, glucose level (fasting) <100mg/dl, (random) <200.
Dr. Hamza Iftikhar	Heart rate : 60-100, Oxygen Saturation : 97-100%, Body temperature : 97-98.6°F, Glucose level : 80-120 mg/dl.
Dr. Lobna Mohamed Yahia	Normal values for each of these vital signs vary by age and, in some cases, by sex. They may also vary based on weight, exercise capability, and overall health. In healthy adults at rest, normal values are as follows: Heart rate (pulse): 60-100 bpm, Respiratory rate: 16-20 breaths per minute, Blood pressure: 120/80 mm Hg, Temperature: 98°F (36.6°C) to 98.6°F (37°C), Normal glucose level From 90 to 130 mg/dL (5.0 to 7.2 mmol/L) for adults and teenagers
Result Summary	The ranges provided by the interviewees are compared against each other to justify the critical body vital ranges.

Question 4: What kind of illnesses are most commonly found in patient checkups? Can these problems usually be resolved by simple medication?	
Answer	
Ms. Zainab Sharif	Abdominal pain, headache, fever, sore throat and nasal congestion. Yes they can be resolved by OTC medications

Dr. Hamza Iftikhar	Upper respiratory tract infections, Acute gastroenteritis, Flu, Common cold, Urinary tract infections. Yes, medication exists for most of these diseases, for example: Cough syrup, Gastric tablets
Dr. Lobna Mohamed Yahia	The physical health check is to help pick up on signs that you may be at risk of diabetes, stroke or heart problems.
Result Summary	The interviews provide the most common diseases that an individual suffers from and visits the hospital for.

Question 5: How early is it possible to identify cancer / cardiac problems / diabetes / common illnesses in a body? What are the common symptoms / indications for these diseases?	
Answer	
Ms. Zainab Sharif	As early as mid-30s with regular health checkup and incidental findings. The most common risk factors are age, lifestyle, diet and positive family history. For example, if a member has cancer, the rest of the immediate family should undergo testing too.
Dr. Hamza Iftikhar	Cardiac problems are usually seen in middle age to older individuals or in those with family history and positive risk factors such as a sedentary lifestyle, obesity, diabetes and Hypertension. Common symptoms are chest pain, shortness of breath and peripheral edema. Diabetes can be diagnosed in individuals of any age. Common symptoms are increased frequency of urination, increased thirst and increased hunger. A strong family history is a good indicator of future disease.
Dr. Lobna Mohamed Yahia	Prediabetes and type 2 diabetes can be detected by measuring fasting plasma glucose or HbA1c level, or with an oral glucose tolerance. A fasting plasma glucose

	level of 100 to 125 mg/dL (5.55-6.94 mmol/L), an HbA1c level of 5.7% to 6.4%, or a 2-hour postload glucose level of 140 to 199 mg/dL (7.77-11.04 mmol/L) are consistent with prediabetes
Result Summary	The interviewees provide information on major medical problems such as cancer and diabetes which they believe might help identify the problem as early as possible.

Question 6: What are the fluctuations in body parameters like temperature, blood oxygen / glucose level, heart rate when a person is suffering from these problems?	
Answer	
Ms. Zainab Sharif	It depends on the disease/infection. Majority of the infections raise the temperature. Blood oxygen is affected when the heart and lungs are compromised. These all also alter with different kind of medications and their side effects.
Dr. Hamza Iftikhar	With cardiac diseases, blood oxygen levels and heart rate can fluctuate. With Diabetes, the glucose levels fluctuate.
Dr. Lobna Mohamed Yahia	Hyperglycemia may affect the compliance of the vascular system, resulting in high blood pressure fluctuations. Diabetes may also directly influence resting heart rate. Hyperinsulinemia and elevated blood glucose levels have both been associated with a higher heart rate.
Result Summary	All the interviewees agree that the presence of diseases cause fluctuation in the body vitals in one way or multiple ones.

Question 7: What other patient factors may be analyzed when identifying a health issue?	
Answer	
Ms. Zainab Sharif	Substance abuse, occupation, bowel movements, micturition and mental status
Dr. Hamza Iftikhar	Family history, Lifestyle, Occupation, History of drug use.
Dr. Lobna Mohamed Yahia	Your personal health risk factors include your age, sex, family health history, lifestyle, and more. Some risks factors can't be changed, such as your genes or ethnicity. Others are within your control, like your diet, physical activity,
Result Summary	All the interviews indicate that body vitals alone are not enough to identify health problems. There are important factors like age, sex, family history etc.

Question 8: What medication and treatment can be used to reduce the effects of these diseases (if effective)?	
Answer	
Ms. Zainab Sharif	Protein pump inhibitors, paracetamol/analgesics, anti-inflammatory nasal drops and warm saltwater gargles with steam.
Dr. Hamza Iftikhar	Lifestyle changes, balanced diet intake, regular exercise and avoiding pollutants and occupational hazards can effectively reduce disease effects as well as the disease occurrence itself. Specific medications for diabetes include insulin therapy and oral antidiabetics such as metformin and sitagliptin.

	Drugs used to treat cardiac diseases include aspirin, cholesterol regulators such as statins and antihypertensives.
Dr. Lobna Mohamed Yahia	Management of type 2 diabetes includes: Healthy eating, Regular exercise, Weight loss, Possible diabetes medication (oral hypoglycemic drugs - Metformin... Sulfonylurea and others) ,Insulin therapy for resistant cases or type 1 diabetes, Blood sugar monitoring. These steps make it more likely that blood sugar will stay in a healthy range. And they may help prevent complications
Result Summary	The interviews provide a list of specific medicine that might be used to prevent further health problems or limit the issue until a person can visit a professional.

Table 3.4 Answers Provided by Doctors & Medical Physicians Interviewed

Answers Provided by Nurses / Caretakers

Question 1: What are the duties you carry out on a regular day as a medical practitioner?	
Answer	
Ms. Angel Gimeno	As a learning student nurse, I carry out simple tasks like assessing the client, administering drugs, taking his/her vital signs and also taking his/her medical history.
Mr. Anselmo Revistual Jr.	Administering medications, treatments, and interventions as ordered by the attending physician, which may include everything from wound care and IV fluids to more advanced interventions like intubation or chest tube placement. Monitoring patients' vital signs and other key health indicators, including heart rate, blood pressure,

	oxygen saturation, and blood glucose levels, and responding quickly if any abnormalities are detected. Providing emotional support and education to patients and their families, helping to ease anxiety and promote understanding of the treatment process.
Result Summary	The responsibilities of a nurse / caretaker is understood.

Question 2: How frequently are patients recommended to come in for checkups? From your perspective, can some examinations be pointless, only resulting in a time hassle?	
Answer	
Ms. Angel Gimeno	The frequency depends on how serious the condition of the client is. From a nurse's perspective, there are no "pointless checkups". Because it is best to be preventive and sure, than neglect small health issues that lead to serious conditions.
Mr. Anselmo Revistual Jr.	The frequency of check-ups can vary depending on factors such as age, medical history, and overall health status. In general, routine check-ups are recommended annually for adults and may be more frequent for individuals with chronic health conditions. While some check-ups may seem pointless or result in a time hassle, it is important to keep up with routine medical care to prevent or detect health issues early on.
Result Summary	All the interviewees highlight that having health checkups is necessary and they should be done even though some might not be fruitful.

Question 3: What are the regular ranges for heart rate, oxygen saturation, body temperature, glucose level in a healthy body teenager / adult?	
Answer	
Ms. Angel Gimeno	Normal range of teenager / adult: Pulse rate: 60-100 bpm, Temperature: 36.5-37.5, Oxygen saturation: 95%-100%, Glucose level: 4.0-7.0 mmol/L (not sure)
Mr. Anselmo Revistual Jr.	I can tell you that the normal heart rate, oxygen saturation, body temperature, and glucose levels can vary slightly based on the individual and their medical history. In general, a healthy adult's heart rate should be between 60 and 100 beats per minute, oxygen saturation levels between 95% and 100%, body temperature around 98.6°F (37°C), and glucose levels between 70-99 mg/dL when fasting and less than 140 mg/dL two hours after eating. These ranges may fluctuate slightly for teenagers, depending on age and gender. It's crucial to note that these ranges might change depending on factors including exercise, stress, and medication use.
Result Summary	The ranges provided by the interviewees are compared against each other, also with the doctor interviews, to justify the critical body vital ranges.

Question 4: What kind of illnesses are most commonly found in patient checkups? Can these problems usually be resolved by simple medication?	
Answer	
Ms. Angel Gimeno	Cancer - The common symptom and indication of these are Weight changes, including unintended loss or gain. Skin changes - such as yellowing, darkening or redness of the skin, sores that won't heal, or

	changes to existing moles. Changes in bowel or bladder habits. Persistent cough or Trouble breathing.
Mr. Anselmo Revistual Jr.	I have seen a wide range of illnesses in patient check-ups, but some of the most commonly seen include hypertension, type 2 diabetes, hyperlipidemia (high cholesterol), thyroid disorders, and skin cancer.
Result Summary	The interview question provide the most common diseases that an individual suffers from and visits the hospital for.

Question 5: What are the fluctuations in body parameters like temperature, blood oxygen / glucose level, heart rate when a person is suffering from these problems?	
Answer	
Ms. Angel Gimeno	Skin cancer can be detected during a routine check-up by examining the skin for abnormal moles or lesions. Fluctuations in skin appearance may occur over time and can be affected by exposure to sunlight or other environmental factors.
Mr. Anselmo Revistual Jr.	Fluctuations in blood pressure may occur throughout the day, but consistently high readings may be a sign of hypertension. Type 2 diabetes may not have any noticeable symptoms, but healthcare professionals may detect it during routine check-ups by measuring blood glucose levels. Fluctuations in blood glucose levels may occur throughout the day, with high levels potentially causing symptoms such as fatigue or nausea.
Result Summary	Both the interviewees agree that the presence of diseases cause fluctuation in the body vitals in one way or multiple ones.

Question 6: Can these problems usually be resolved by simple medication? What other patient factors may be analyzed when conducting a preliminary checkup to identify a health issue?	
Answer	
Ms. Angel Gimeno	Yes, some problems only require the doctors providing prescriptions to the patient, but other people need more severe, operational treatment.
Mr. Anselmo Revistual Jr.	Some of these problems can be resolved with simple medication, but others may require more extensive treatment. When conducting a preliminary checkup, other patient factors that can be analyzed to identify health issues include family medical history, lifestyle habits (such as smoking or excessive drinking), and previous medical conditions or surgeries.
Result Summary	Both the interviewee responses address the fact that medication is available for the issue but more sever issues will need more than medicine.

Question 7: What are the steps taken to constantly monitor and pay attention to your patients? Which type of patients require the most monitoring?	
Answer	
Ms. Angel Gimeno	Probably regular checking of the vital sign of the patient will do. We should check them regularly to get an update on their health condition. We can also ask the patient's guardian or next of kin about how they feel as a secondary source. The patients that needs to be monitored continuously are those who are diagnose in a high risk conditions and other acute conditions. Were In as nurses we should be alert and always monitor their vital signs for chart records

Mr. Anselmo Revistual Jr.	The first step in monitoring patients is to assess their vital signs, including heart rate, blood pressure, oxygen saturation, blood glucose levels, and temperature.
Result Summary	The interview responses address how it is possible to monitor health conditions without them and why it is important to first check the body vitals .

Question 8: How difficult is it to continuously monitor the health conditions of patients? How much is the quality of monitoring affected by other nursing work?	
Answer	
Ms. Angel Gimeno	For me, it is in normal range of difficulty. The process itself is not difficult but with so much work that nurses do, the monitoring quality is highly affected by other work. There are different patients to take care of and the need for documenting and delivering this information to doctors needs us to perform quick.
Mr. Anselmo Revistual Jr.	As an ER nurse, continuously monitoring the health conditions of patients is a critical aspect of my job. It can be quite challenging to keep track of multiple patients and their various health parameters, especially during busy shifts. The quality of monitoring can be affected by other nursing work, as we have many responsibilities, such as administering medication, documenting patient information, and communicating with other healthcare professionals. However, we are trained to multitask and prioritize our duties to ensure that patient monitoring is not compromised
Result Summary	Both the interviewees agreed that that the quality of monitoring can be hindered by the quantity of work. However, they are trained to handle all their responsibilities with equal importance.

Table 3.5 Answers Provided by Nurses / Caretakers Interviewed

Answers Provided by Patients / Regular People

Question 1: How frequently do you visit hospitals or clinics for health checkups?	
Answer	
Mr. Dilon Fernando	To take care of my health issues, my doctors ask me to come and visit for a checkup every 5-6 months so that my well-being does not worsen.
Mrs. Janaki Fernando	In order to maintain good health, I am forced to visit the hospital at least once in three months.
Mrs. Rubina Khatoon	To maintain my overall health and wellness, I make it a habit to get a weekly health checkup at the hospital.
Mr. Dane Francis Dasalla	In all honestly, I do not visit the hospital that often even though I know I should go more often.
Ms. Surbhi Hazra	My parents take me to visit a doctor almost every month. Since my childhood, my immune system has been quite weak. So, I have to visit the hospital to be free of recurring health problems.
Result Summary	All the interviewees state that it is important to visit the hospital / clinic frequently whether they go or not.

Question 2: What is the most common illness, problem for which you visit hospitals and clinics? Is going / contacting a professional the first thought when you experience symptoms?
Answer

Mr. Dilon Fernando	Mostly go to get checkups done for my uric acid and thyroid levels but can also visit when I might have high fevers.
Mrs. Janaki Fernando	I visit the hospital to treat my thyroid issues, since I need professional help to treat my problem. I would go more often if I feel other symptoms such as tiredness and body- fatigue
Mrs. Rubina Khatoon	Type 2 Diabetes is the illness that I am most frequently diagnosed with, prompting me to go to the hospital for a checkup every week. While my initial thought is to take the medications prescribed by my doctors, my family insists that I visit the clinic to have my condition assessed regularly if I experience any symptoms.
Mr. Dane Francis Dasalla	I do not face a lot of issue so usually if I go to a hospital / clinic, it is because I am suffering through an extremely high fever or some terrible coughing.
Ms. Surbhi Hazra	Because I suffer from anemia, whenever my body starts feeling tired, my family rushes me to the hospital. We can buy the same medicine, but we would rather rely on the doctor's opinion
Result Summary	The interview question provide the most common diseases for which a patient / person suffers visits a hospital.

Question 3: From your perspective, can some checkups end up being pointless, only resulting in a time hassle?	
Answer	
Mr. Dilon Fernando	The visits are necessary as I have to get a certain prescription based on the test results. Even, if the

	prescription does not change from the last time the tests give me assurance.
Mrs. Janaki Fernando	No, I think it is very important for me as I can't go without my medication. So, keeping time to go for a checkup is crucial.
Mrs. Rubina Khatoon	Occasionally, the weekly health updates can be frustrating when there are no significant updates, and it can feel like a waste of time and money if the doctor provides the same feedback as the previous week.
Mr. Dane Francis Dasalla	Yes, as sometimes the doctors do not get the right medicine or not get the right method of consultation therefore leading me to buy inappropriate medicine such as aspirin.
Ms. Surbhi Hazra	Yes, it can be useless at time. I am a student and usually have a lot of work to complete. When I go for a checkup, it takes up a lot of a time and would just be irritating if nothing potentially changed.
Result Summary	All the interviewees highlight that even though health checkups are necessary, they can be troublesome to perform and perform

Question 4: Have you ever taken medication based on assumptions and avoided going to the hospital? Did this have a further negative affect?	
Answer	
Mr. Dilon Fernando	Never. I may only take Panadol or aspirin tablets to reduce body pain.
Mrs. Janaki Fernando	Never. It would be too dangerous to ever risk my health by taking random medication.

Mrs. Rubina Khatoon	No, thanks to my family's vigilant monitoring of my medication dosages, I haven't taken any medication based on assumptions or simply a reason to not attend my hospital appointments.
Mr. Dane Francis Dasalla	Yes, I have taken medication based on assumptions. There might have been a slight negative effect as it did not heal my issues and I felt the same or even worse after taking the medicine.
Ms. Surbhi Hazra	No, I have only taken medication prescribed by my doctors. But there have been occasions where, I did not visit the hospital and my parent purchased the same medication as before from a clinic.
Result Summary	All the interviewees agree that taking medication on assumption can only cause more harm than benefits.

Question 5: What are the reasons you may not go to visit a clinic or hospital?	
Answer	
Mr. Dilon Fernando	Sometimes I may not get enough time to visit the hospital because of my work and studies. I have a hectic schedule and really have to squeeze the hospital visit in.
Mrs. Janaki Fernando	Only if the problem is minor like a common cold, coughing would I not got to the hospital / clinic
Mrs. Rubina Khatoon	As the principal of a school, it can be difficult to take time off from work responsibilities. Additionally, I have had negative experiences with the healthcare system in the past and find the high cost of healthcare to be an obstacle.

Mr. Dane Francis Dasalla	It is quite expensive to visit a hospital even if it is for a simple checkup. I live in a suburban areas and getting to the hospital is already expensive without the actual medical bill. It also takes quite a while getting there and coming back.
Ms. Surbhi Hazra	As mentioned earlier, being a student, I struggle with managing time.
Result Summary	The interviewees mainly pointed that financial and time issues prevent them from visiting hospitals and clinics.

Question 6: How do you constantly monitor your own health at your homes then?	
Answer	
Mr. Dilon Fernando	It is not really possible to monitor my illnesses. I can only follow the doctor's instructions and rely on symptom indications to remain healthy.
Mrs. Janaki Fernando	I do have an arm blood pressure checker to check my levels to reduce hospital visits for needless reasons.
Mrs. Rubina Khatoon	To keep track of my blood sugar levels, I utilize a blood pressure monitor and a glucose meter. In addition, I make an effort to maintain a healthy and balanced diet for optimal health, while ensuring that I get a minimum of 8 hours of sleep every night.
Mr. Dane Francis Dasalla	I do not usually monitor my own health if I am healthy and feeling in shape. If I start feeling any symptoms then my monitoring only involves checking the temperature and possibly checking my heart rate through the smart phone or smart watch..

Ms. Surbhi Hazra	I don't monitor my health. I visit the doctor once a month and only can see if I have a problem if I feel tiredness symptoms.
Result Summary	The responses indicate that it is difficult to monitor their health and they are very reliant on their doctors

Table 3.6 Answers Provided by Patients / Regular People Interviewed

3.1.2 Dataset Analysis

The author received a lot of information from the interviews conducted. However, all the interviewees kept their patients' or their own medical data private. Therefore, it was not entirely possible to get specific body values for the information base that will be used to predict disease and their recommended solutions. With respect to the interviewees privacy, no further questions were asked but gaining that information was also critical for the healthcare system. In order to achieve this, the author gathered data from public medical datasets and other online sources. Medical information was searched for and found through Kaggle datasets.

For example, Diabetes is among the most prevalent chronic diseases (Tebol, 2021). While there are different types of diabetes. Through predictive models and their characteristics, the data was evaluated to figure out further symptoms, values etc., through machine learning graphs and tables. This was done for other diseases such as cardiac problems where information such as cholesterol levels were signified. Even a recent problem such as the Corona virus was investigated for the system to find the factors (Xingyu, 2023).

3.1.3 Fact-Finding Summary

The interview and dataset analysis provided a lot of valuable medical information. From the data sources, the author was able to collect critical medical data such as the health body vital ranges for temperature, blood glucose, blood oxygen saturation as well as heart rate. In addition to these ranges, information was collected on different kinds of diseases and illnesses. The most common diseases were identified as well as the specific names of the medicine which can be implemented in the eventually developed healthcare system. The requirements gathering process highlighted the difficulties the people face in monitoring their health conditions, the symptoms they face and the frequency at which they visited or wished to visit hospitals. All of this further iterated how important the role of doctors are in their lives and highlighted significant points as to why and how the proposed healthcare system would help the target audience.

3.2 Classifying Requirements

To develop the proposed healthcare system, the requirements gathered from the different fact-finding techniques are classified and specified to ensure that they are implemented in the development of the overall system. The requirements can be mainly classified into functional and non-functional requirements.

3.2.1 User Requirements

The user of the healthcare system should be able to perform all basic operations such as accessing the healthcare application - logging in and registering but also be able to change their password if they do not remember. These registered datils should also be editable to the users to keep up-to-date information. In terms of healthcare, users should be able to conduct an overall health checkup and also be provided with the ability to choose a specific type of body parameter monitoring if needed. Users should be able to communicate with the chatbot and finally be able to view and track all the monitored data and receive disease names with their analyzed solution.

3.2.2 Functional Requirements

1. Login & Register

The proposed healthcare system allows users to register an account for the application and log in. The application can be logged into by anyone that is above the age of 13 since this system is not compatible for younger children. In the case, the user forgets their registered account's password, the system allows the user to perform a reset password operation. The system delivers a one-time password (OTP) to the user's email so that the user can change their password.

2. Record & Monitor Body Vitals

The healthcare system will use all the different sensors in the IoT architecture to measure the values of a single user's body temperature, heart rate, blood glucose level as well as blood oxygen saturation level. The user will physically interact with the sensors connected so the IoT devices can pick up values.

3. Display Monitored Body Parameters

Once the medical sensors are done recording the body parameters of the user, the system forwards the information to the healthcare application so the user can observe his / her body vitals' values. The user can observe this data in different ways in the interface with the recordings first being shown immediately after a health checkup and the other display being shown in graphs and reports as recorded statistics.

4. Identify Potential Diseases / Illnesses

There will be a rule-based AI algorithm that will be employed in the healthcare system to analyze monitored data values to identify health issues from possible symptomatic body vital ranges. When a checkup is completed, the system will use the all the data with logic operations in a forward chaining process to reach a conclusion with the body vital values acting as the control key for initial comparison. The information will be displayed along with the complete reasoning that made the system identify the particular issue.

5. Recommend Medical Solutions to Identified Disease / Illness

As part of the rule-based AI algorithm that identifies a disease, the healthcare system will also identify the recommended solution corresponding to the specific disease / illness found. The information will be displayed alongside the disease for the user to consider and take the recommended, precautionary steps.

6. Edit User Information

The healthcare system provides users the ability to edit their personal information. The functionality will ensure that the system is up to date with the user's current health status and ensures that all disease identifications and health recommendations are reliable and accurate. Furthermore, the user can also edit their account details if they require.

3.2.3 Non-Functional Requirements

1. Availability

Since the healthcare system uses IoT devices and stores data into a cloud database, any Android smartphone that is connected to the internet can access this healthcare application (system). When a user wants to conduct a health checkup, they must use and have access to the physical IoT architecture for recording the values into a cloud database. The sensors must obviously be connected to a power supply to measure the body vitals and send the values via Wi-fi. When the user logs in, if the application has an internet source, the data of the user is retrieved and displayed in the app screens. The system is only applicable for use via an Android application so all users that have Android-based smartphones and tablets, can install this application, and use the healthcare system. It might be more convenient to install onto the tablet to view the graphs and reports in an upscaled view, but the device used simply depends on the user's wishes.

2. Accuracy

Since the healthcare system utilizes sensor technology, the healthcare system identifies body parameter values to the sharpest precision possible. There is no manual input involved and all the sensed, accurate measurements are automatically input and forwarded to cloud storage. The accuracy of measurement depends mainly on how well the sensors fit together as part of the IoT structure. The accuracy of these values is also a key requirement when

it comes to identifying the potential diseases as the measurements are the first rules of a rule-based algorithm. The accuracy of predicted diseases and their solutions depends not only on the vital recordings but also on other health factors such as eating habits, current medications etc. The system considers all these factors before coming to a conclusion and goes through the same rule-based process to recommend accurate treatment or medicine.

3. Reliability

Since the healthcare system uses IoT sensors and every recording is automated, the healthcare system is pretty reliable in receiving the health parameters and conducting all the analysis on the data to identify an illness. Since the body vitals are continuously monitored the longer the user interacts with the sensor architecture, the system uses a reliable method of calculating the average recorded value for the duration the body vital was kept steady. The healthcare application will be able to support and protect a user's heath information from other users as all the data is always specific to one account so the graphs and reports a user can view through the application are always up-to date and accurate. The application is consistent overall with the user interface and the healthcare Chatbot. The Chatbot replies and prompts the user in a streamlined way to simply help the user conduct a checkup and deliver its results in a standardized way.

4. Performance

When the user conducts a health checkup, the healthcare system provides the measurements in a rapid and frequent manner. There is minimal time delay between the time the IoT devices sensing the human body and the value being recorded. When using the healthcare application, the user is navigated through each interface screen via button presses in an efficient method. The Chatbot processes user input by only searching for key words through the process of tokenization so that a response can be delivered in a correct and efficient manner. If the Chatbot does not find any keywords to take further action, it messages the user to redirect them back to a situation where the Chatbot could provide help.

3.3 System Design

The system design process follows the requirements gathering and classification processes. The system is designed using common UML diagrams to provide key personnel such as the system developers and analysts with a clearer view of the healthcare system (Alfandi, 2022). The described UML diagrams in the section below include all the requirements gathered such that the development of the system and its components are laid out in a straight-forward and clear manner.

3.3.1 Rich Picture Diagram

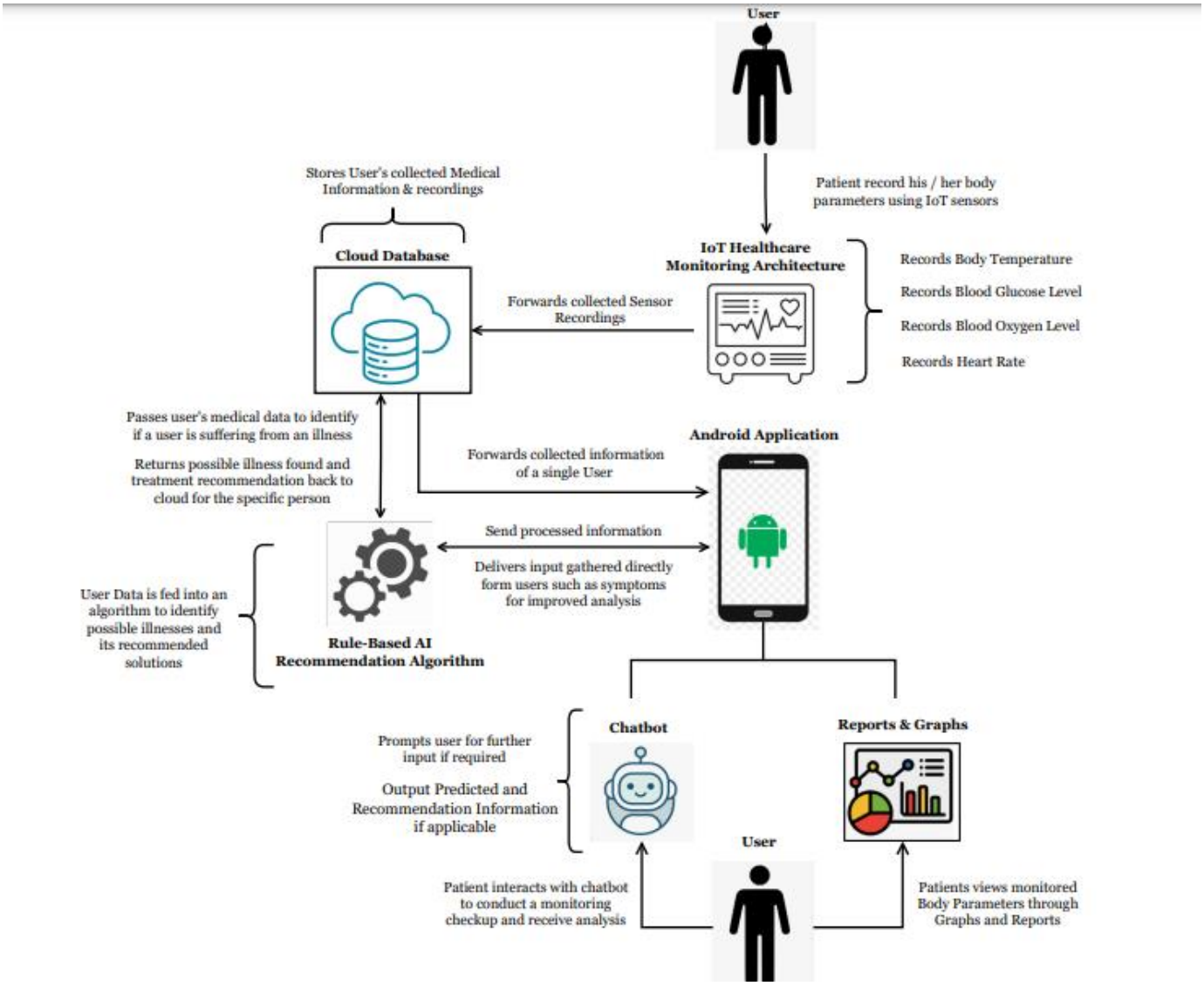


Figure 3.1 Rich Picture Diagram of Proposed Healthcare System

Figure 3.1 depicts the overall working of the proposed healthcare system. The figure displays how the major components of the system interact with each other and process information. The system involves using the same user at two points of the system. The user can first log into the healthcare application and access the chatbot to conduct a health checkup. The user utilizes the IoT Monitoring sensors to record their current body parameters. This recorded data is sent to the database cloud for storage and future retrieval. But also, data from the cloud is put into rule-based AI algorithms to evaluate the data and identify if the information indicates any potential diseases with their solutions. The analysis is forwarded back to the cloud database which is eventually retrieved back into the healthcare application. The user can view all the past and current monitored data with the chatbot also providing information on any identified health problems and their projected solutions.

3.3.2 IoT Block Diagram

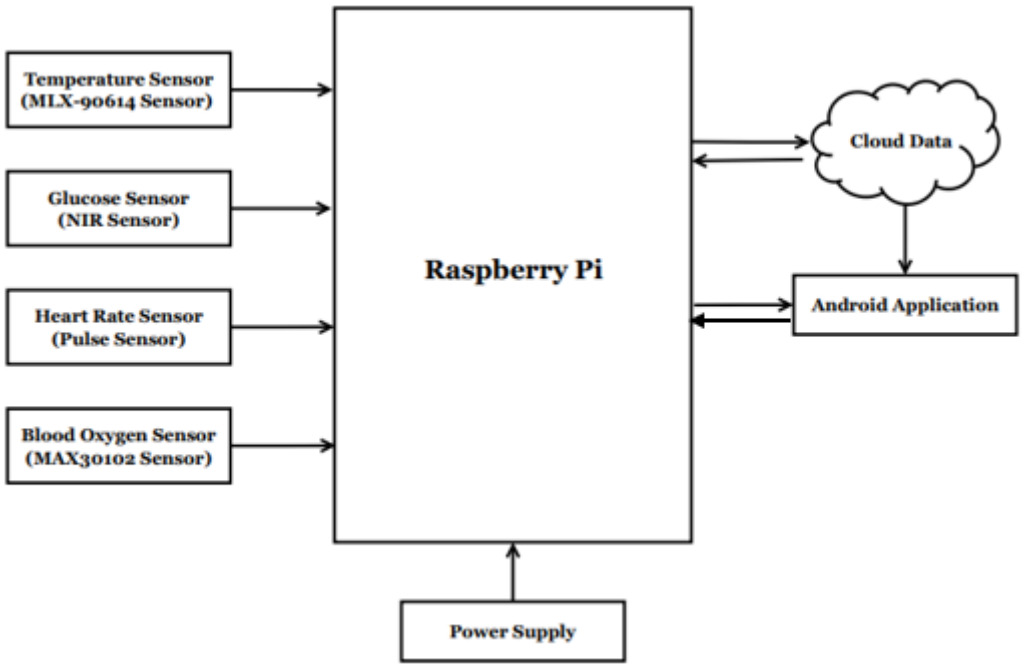


Figure 3.2 Block Diagram of IoT Architecture in Proposed Healthcare System

The IoT Block Diagram in Figure 3.2 illustrates how the sensors are placed and connected together. All the health monitoring sensors – temperature, glucose, heart rate and blood oxygen sensors are connected to the Raspberry Pi processor so that it can receive input data to be managed. The microprocessor is obviously connected to a power supply for the whole operation to work. Through the use of Internet Wi-Fi module, the processed data can be forwarded as well as received to and from the cloud. The application is connected to the Raspberry Pi module to retrieve and send instant data but mainly collects information from the cloud database.

3.3.3 Use Case Diagram

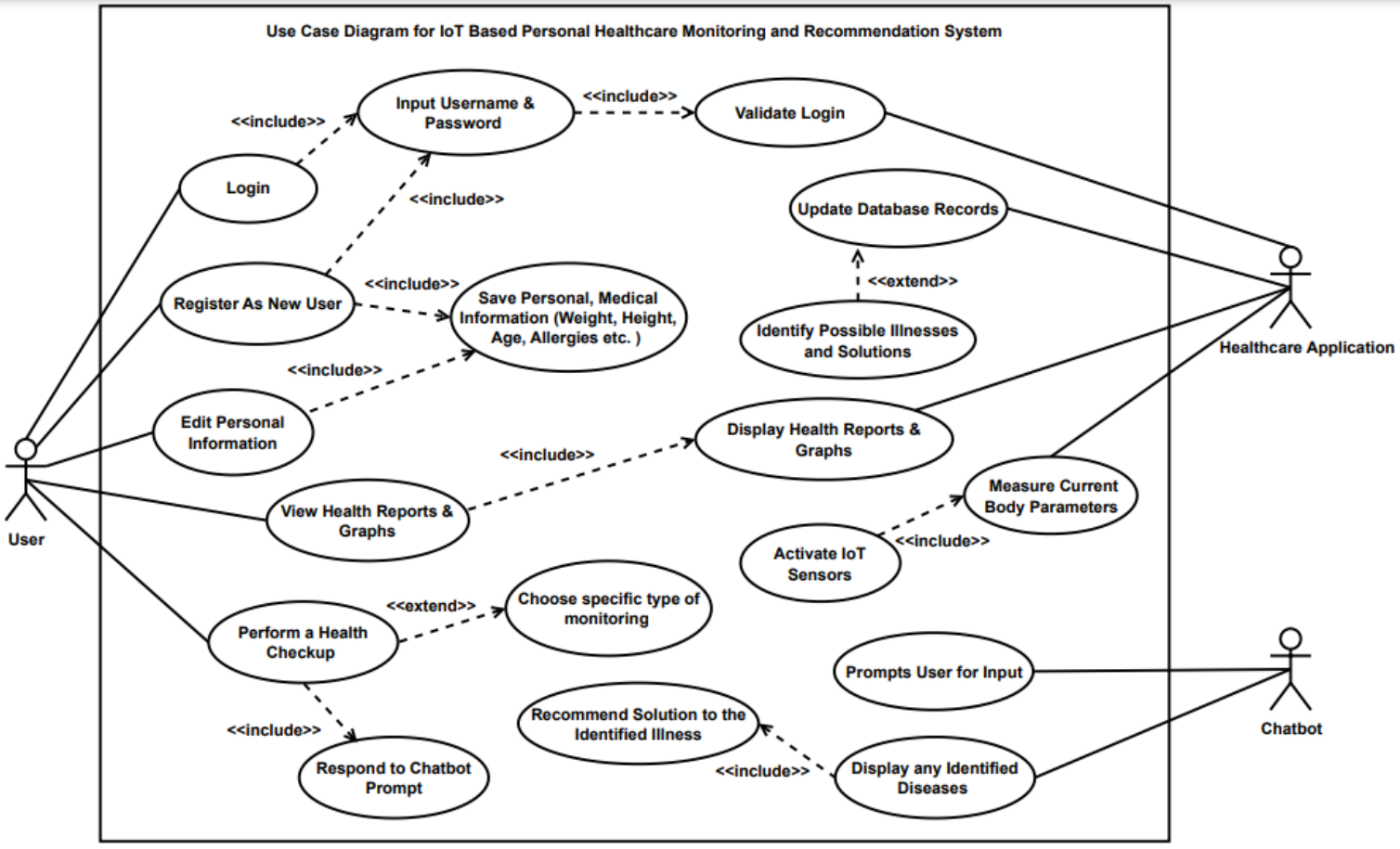


Figure 3.3 Use Case Diagram of Entire Healthcare System

Figure 3.3 displays the Use-Case Diagram of the proposed healthcare system. The Use-Case diagram illustrates all the actors involved and the functionalities they are expected to perform. In this scenario, the health care system involves 3 major actors – the User, the Healthcare Application and finally the Chatbot. Table 3 below describes the complete list of actors and the use cases they execute which is followed by Table 3 which details each use case and the typical course of events associated with it.

1. List of Actors

Actor		**Use Case Name**	**Use Case Description**
User	initiates	Login	Users can attempt to login to the healthcare system by entering their previously registered username and password.
	initiates	Register As New User	If a single user does not have an account, they cannot access the healthcare system. So, users are provided the ability to register a new account and enter their personal and medical information – weight, height, current allergies etc. for future reference.
	initiates	Edit Personal Information	Users when accessing the healthcare system can edit their personal information if certain aspects have changed. For example, if the weight of a user has changed, they can update the information for more predictions and recommendations. Users ae also provide the ability to edit their password if they wish.
	initiates	View Health Reports & Graphs	While using the healthcare application, users can view their current and past monitored body parameters.

			The application collects the records from the database and displays it in the form of graphs and tables for the user to investigate.
	initiates	Perform a Health Checkup	A single user is able to conduct a health checkup when using the healthcare system. The user can interact with the medical chatbot and request to monitor specific or all body vitals using the IoT Sensors.
Healthcare Application w/ IoT System	initiates	Validate Login	The healthcare application communicates with the database to authenticate a user's username and password when attempting to login.
	initiates	Display Health Reports & Graphs	The healthcare application once again connects to the database and retrieves all the logged in user's information and depicts the monitored data in the form of graphs and tables in the application.
	initiates	Measure Current Body Parameters	The healthcare application is responsible for activating the IoT sensors when a user wishes to conduct a health checkup. When the user uses the sensors, the application records this information.

	initiates	Update Database Records	The healthcare application analyzes and forwards information such as predicted illnesses and the corresponding solution of a user to the database so it can be updated along with the new monitored data.
Chatbot	initiates	Prompts User for Input	The Chatbot interacts with user so they can monitor their body parameters.
	initiates	Display any Identified Diseases	Once the user conducts their checkup, the Chatbot delivers a summary of the monitored data along with any analyzed disease.
	initiates	Recommend Solution to Identified Illness	If the system does identify a possible disease, the Chatbot also displays the recommended medication or solution for the user.

Table 3.7 List of Actors in the Healthcare System

2. Use Case Descriptions

USE CASE NAME:	Login	
USE CASE ID:	US01	
PRIMARY ACTOR	User	
OTHER PARTICIPATING ACTORS:	Healthcare Application w/ IoT System	
DESCRIPTION:	Users can attempt to login to the healthcare system by entering their previously registered username and password.	
PRE-CONDITION:	User must have a registered account to the healthcare application.	
TRIGGER:	User presses the 'Log In' button for username and password validation on the Login Page.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User enters their username and password in the Login Page of the application	
	Step 2: User presses the 'Log In' button to get access to the healthcare interface.	**Step 3**: System checks to see if user details are correct and correspondent to each other.
		Step 4: System directs the page to healthcare application Homepage.
	Step 5: User has logged in and accesses Homepage.	
ALTERNATE COURSES:	**Alt-Step 4:** System finds that username and password entered are not correct and could not be found from the database.	
	Alt-Step 5: User is provided a message of unsuccessful login on the Login Page. User stays on the same page and is asked to try again or register.	
CONCLUSION:	The use case concludes when the User reaches the Homepage of the healthcare application	

POST-CONDITION:	Users can access all the different functionalities of the healthcare system once logged in.
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	Unsuccessfully logging in more than 3 times may block the account for a specific duration.

Table 3.8 Use Case Description of 'Login Use' Case

USE CASE NAME:	Register As New User	
USE CASE ID:	US02	
PRIMARY ACTOR	User	
OTHER PARTICIPATING ACTORS:	Healthcare Application w/ IoT System	
DESCRIPTION:	If a single user does not have an account, they cannot access the healthcare system. So, users are provided the ability to register a new account and enter their personal and medical information – weight, height, current allergies etc. for future reference.	
PRE-CONDITION:	User does not have a registered account for the healthcare application.	
TRIGGER:	User presses 'Register' button on the Register Page.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User enters their personal, medical details and creates a username and password for the application.	
	Step 2: User presses the 'Register' button to save the details.	**Step 3**: System saves the newly entered details into the healthcare system database.
		Step 4: System provides successful registration message and directs user back to Login Page.

	Step 5: User can now login using the newly created details	
ALTERNATE COURSES:	**Alt-Step 3:** System provides error message to fill in all fields correctly for Registration.	
	Alt-Step 4: User checks fields and fills in properly according to validation error messages and presses 'Register' button.	
	Alt-Step 5: System provides successful registration message and directs user back to Login Page.	
CONCLUSION:	The use case concludes when User is returned back to Login Page.	
POST-CONDITION:	User can login to the healthcare app with newly created username and password, while also retrieving the saved medical details	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	One email can only have corresponding account.	

Table 3.9 Use Case Description of 'Register As New User' Use Case

USE CASE NAME:	Edit Personal Information
USE CASE ID:	US03
PRIMARY ACTOR	User
OTHER PARTICIPATING ACTORS:	Healthcare Application w/ IoT System
DESCRIPTION:	Users when accessing the healthcare system can edit their personal information if certain aspects have changed. For example, if the weight of a user has changed, they can update the information for more predictions and recommendations. Users are also provided the ability to edit their personal information if they wish.
PRE-CONDITION:	User is on the Profile Page on the healthcare application.

TRIGGER:	User finds and changes the piece(s) of information to be updated and presses the 'Edit' button.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User chooses the piece of information that they wish to change on the User Profile interface pages	**Step 2**: System activates the specific field to be edited.
	Step 3: User modifies the value or text in the chosen fields and saves the changes	**Step 4:** System receives updated information for storage and updates specific database records
	Step 5: User views the saved, updated information as part of the profile.	
ALTERNATE COURSES:	**Alt-Step 3:** User does not actually change any field and returns back to the Profile Page.	
	Alt-Step 4: System does not change any field values and simply directs back to un-editable Profile page.	
CONCLUSION:	The use case concludes when the new information is saved and displayed throughout the application.	
POST-CONDITION:	The healthcare system will consider updated information when analyzing potential diseases, especially if there were changes to factors such as weight, height, or other diagnosed diseases.	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	-	

Table 3.10 Use Case Description of 'Edit Personal Information' Use Case

USE CASE NAME:	View Health Reports & Graphs	
USE CASE ID:	US04	
PRIMARY ACTOR	User	
OTHER PARTICIPATING ACTORS:	Healthcare Application w/ IoT System	
DESCRIPTION:	While using the healthcare application, users can view their current and past monitored body parameters. The application collects the records from the database and displays it in the form of graphs and tables for the user to investigate.	
PRE-CONDITION:	User is logged in to the healthcare application.	
TRIGGER:	User presses the 'Statistics' navigation button or one of the last monitored parameters to view the graphs and reports.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User presses the 'Statistics' navigation button to view the monitored results of a body vital.	**Step 2**: System directs user to the Monitored Statistics Page and displays the graphs and reports of the user's heart rate through the week.
	Step 3: User can view the graphs and tables of their heart-rate recordings over the week	
	Step 4: User views different body vital, for example, temperature by pressing the 'Body Temp.' button.	**Step 5**: System refreshes page to display all the graphs and reports of the user's temperature, O2 levels or glucose levels through the week. In this case, the page retrieves body temperature data.
	Step 6: User can view the graphs and tables of their temperature recordings over the week.	

ALTERNATE COURSES:	**Alt-Step 1:** User presses one of the last monitored parameters on the homepage to view the monitored results of a body vital.
	Alt-Step 2: System directs user to the Monitored Statistics Page and displays the graphs and reports of the vital user chose from the home page.
	Alt-Step 3: User can view the graphs and tables of the chosen body parameter over the week. User decides to view the recording over a month.
	Alt-Step 4: System refreshes page to retrieve all the information and display all the graphs and reports for a chosen body vital in a month.
	Alt-Step 5: User can view the graphs and tables of their chosen body vital over a month.
CONCLUSION:	The use case can be concluded when the Monitored Statistics page is viewed by the user.
POST-CONDITION:	User can track, check and compare values of monitored recordings
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	A newly registered user will not be able to access any records since he / she has not previously monitored body vitals.

Table 3.11 Use Case Description of 'View Health Reports & Graphs' Use Case

USE CASE NAME:	Perform a Health Checkup
USE CASE ID:	US05
PRIMARY ACTOR	User
OTHER PARTICIPATING ACTORS:	Chatbot, Healthcare Application w/ IoT System
DESCRIPTION:	A single user is able to conduct a health checkup when using the healthcare system. The user can interact with the medical chatbot and request to monitor specific or all body parameter using the IoT Sensors.

PRE-CONDITION:	User is logged in to the healthcare application.	
TRIGGER:	User requests (messages) chatbot to conduct a health checkup and IoT sensors are activated for user's interaction.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: User requests for health checkup using Chatbot.	**Step 2**: System activates all IoT Sensors
	Step 3: User physically interacts with the sensors to record the body vitals	**Step 4:** Sensors detect the monitored values and forward the recorded values to the cloud database for storage and analysis.
	Step 5: User views the current monitored recordings through Chatbot.	**Step 6:** Systems analyzes recordings for diseases.
	Step 7: User receives identified health problem and their recommended solution.	
ALTERNATE COURSES:	**Alt-Step 4:** Sensors cannot detect the user and forward the recorded values to the cloud database for storage and analysis.	
	Alt-Step 5: User tries again to interact with the sensors to record the body vitals.	
	Alt-Step 6: Sensors detect the monitored values and forward the recorded values to the cloud database for storage and analysis.	
	Alt-Step 7: User views the current monitored recordings through the Chatbot.	
	Alt-Step 8: Systems identifies no diseases, so User receives an all-clear message.	
CONCLUSION:	The use case concludes when the current body parameters are recorded, and the recordings are analyzed.	
POST-CONDITION:	If a potential illness is identified, the solution is recommended so User can take precautionary measures. The recordings are updated into the graphs and reports.	

ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	The IoT architecture must be constructed and joined together properly. Any wiring or loose connections result in faulty values.

Table 3.12 Use Case Description of 'Perform a Health Checkup' Use Case

USE CASE NAME:	Validate Login	
USE CASE ID:	HA01	
PRIMARY ACTOR	Healthcare Application w/ IoT System	
OTHER PARTICIPATING ACTORS:	User	
DESCRIPTION:	The healthcare application communicates with the database to authenticate a user's username and password when attempting to login.	
PRE-CONDITION:	User has entered and submitted a username and password for verification.	
TRIGGER:	User presses the 'Log In' button for username and password validation on the Login Page.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: The application accepts the user's username and password	
	Step 2: The application searches for input username	**Step 3**: The system returns all the information corresponding to the found username.
	Step 4: The application compares the stored password of associated username with input password.	**Step 5:** The system directs the user to healthcare application homepage if passwords match.
ALTERNATE COURSES:	**Alt-Step 3:** The system cannot find the input username and displays error message on the Login Page.	

	Alt-Step 4: After application accept valid username input**,** the application compares the stored password of associated username with input password.
	Alt-Step 5: The system provides another unsuccessful login message to user if passwords do not match.
CONCLUSION:	The use case concludes when the comparison of username, password or both are completed.
POST-CONDITION:	Users can either access all the different functionalities of the healthcare system or be forced to try again.
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	User must have a previously registered account to the healthcare application.

Table 3.13 Use Case Description of 'Validate Login' Use Case

USE CASE NAME:	Display Health Reports & Graphs	
USE CASE ID:	HA02	
PRIMARY ACTOR	Healthcare Application w/ IoT System	
OTHER PARTICIPATING ACTORS:	-	
DESCRIPTION:	The healthcare application once again connects to the database and retrieves all the logged in user's information and depicts the monitored data in the form of graphs and tables in the application.	
PRE-CONDITION:	User has successfully been logged in and validated	
TRIGGER:	-	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: The healthcare application reads the username and retrieves all the information linked to the account	**Step 2**: The system forwards all the user information, including the monitored body vital values

		corresponding to the account, back to the application.
	Step 3: The healthcare application receives all this data and constructs all the graphs and reports of the user's body parameters so that it could be displayed on the Monitored Statistics page.	
ALTERNATE COURSES:	**Alt-Step 2:** The system forwards all the user information but does not send any monitored body parameters corresponding to the account since there are no past records (newly registered user).	
	Alt-Step 3: The healthcare application receives all this data and leaves the Monitored Statistics page clear requesting the user to conduct health checkups for graphs and report visualization.	
CONCLUSION:	The use case is concluded once the User can see some content on the Monitored Statistics page.	
POST-CONDITION:	Users can investigate the recordings of the different measured body parameters - heart rate, body temperature, glucose level and O2 level over a week or month.	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	The Monitored Statistics interface can be shown as four different pages, with each page corresponding to a monitored health parameter.	

Table 3.14 Use Case Description of 'Display Health Reports & Graphs' Use Case

USE CASE NAME:	Measure Current Body Parameters	
USE CASE ID:	HA03	
PRIMARY ACTOR	Healthcare Application w/ IoT System	
OTHER PARTICIPATING ACTORS:	User, Chatbot	
DESCRIPTION:	The healthcare application is responsible for activating the IoT sensors when a user wishes to conduct a health checkup. When the user uses the sensors, the application records this information.	
PRE-CONDITION:	User interacted with Chatbot and specified what kind of checkup / body parameter is to be measured	
TRIGGER:	The system detects the user's physical body via the sensors in the IoT architecture.	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: The healthcare application uses the Chatbot request so that a User can interact with sensors one at a time.	**Step 2**: The system signals the applicable senor to start recording values when it detects the User's body.
	Step 3: The healthcare application records the value for a certain time and saves the average value of recordings.	**Step 4**: The system signals the next applicable senor as part of a health checkup to start recording values
	Step 5: The healthcare application repeats steps 3 and 4 until health checkup is complete.	
	Step 6: The healthcare application sends the real-time recorded values with the analysis of data to Chatbot for informing User.	

ALTERNATE COURSES:	**Alt-Step 3:** The healthcare application cannot detect the users' body so makes the Chatbot send a help message	
	Alt-Step 4: The healthcare application records the value for a certain time and saves the average value of recordings.	
	Alt-Step 5: The system does not signal other sensors as User requested specific monitoring checkup.	
	Alt-Step 6: The healthcare application sends the real-time recorded values to Chatbot for informing User.	
CONCLUSION:	The use case is concluded when the requested checkup is completed.	
POST-CONDITION:	The monitored values can be analyzed to predict possible underlying diseases and immediately recommend treatment or medication	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	Sensors of the IoT architecture must be fit perfectly to receive accurate readings.	

Table 3.15 Use Case Description of 'Measure Current body Parameters' Use Case

USE CASE NAME:	Update Database Records	
USE CASE ID:	HA04	
PRIMARY ACTOR	Healthcare Application w/ IoT System	
OTHER PARTICIPATING ACTORS:	User	
DESCRIPTION:	The healthcare application analyzes and forwards information such as predicted illnesses and the corresponding solution of a user to the database so it can be updated along with the new monitored data.	
PRE-CONDITION:	User completes a whole health checkup.	
TRIGGER:	The system receives data values from the IoT sensors and displays the real-time recorded values to user on application	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
	Step 1: When a health checkup is completed by the user, the healthcare application sends the recorded measurements for storage in the cloud database.	**Step 2**: The system adds the measured data as new records in the database
	Step 3: The health application analyzes the data using rule-based algorithms to find possible indications of illnesses.	**Step 4:** The system identifies an illness and delivers the name and reasoning to the application.
	Step 5: Using the predicted disease, the health application then searches the information base for the corresponding medication and treatments to recommend as solutions to the problem.	
	Step 6: The healthcare application after performing Step 4 and 5 send the identified illness and recommended solution to the database	**Step 7**: The system updates the records with the newly analyzed information.

ALTERNATE COURSES:	**Alt-Step 4:** The system identifies no particular problem and delivers an all-clear message to the application.
	Alt-Step 5: The system updates the records with the newly analyzed information.
CONCLUSION:	The use case can be concluded once the healthcare database records have been updated with the values and analysis.
POST-CONDITION:	The application can update its Monitored Statistics page with the new recordings so the user can view and compare the current recordings with the past ones. Additionally, the predicted illness and recommended solution can be delivered to Chatbot for informing the user.
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	The identified diseases and recommended solutions analyzed and delivered to User are limited to the ones input into the database information base.

Table 3.16 Use Case Description of 'Update Database Records' Use Case

USE CASE NAME:	Prompt User for Input	
USE CASE ID:	CH01	
PRIMARY ACTOR	Chatbot	
OTHER PARTICIPATING ACTORS:	User, Healthcare Application w/ IoT System	
DESCRIPTION:	The Chatbot interacts with user so they can monitor their body parameters.	
PRE-CONDITION:	User is logged in to the healthcare application.	
TRIGGER:	User presses the 'MediBot' navigation button on the Healthcare application to conduct a health check up	
TYPICAL COURSE	**Actor Action**	**System Response**

OF EVENTS:	**Step 1**: Chatbot asks User how they could help to begin user health checkup.	**Step 2**: After User enters their response, the system analyzes their response to check if they requested a checkup.
	Step 3: Chatbot confirms with User whether they would like to conduct an overall checkup.	**Step 4**: After User enters their response, the system analyzes the response and activates IoT sensors for the checkup.
	Step 5: After a Checkup is complete, if the analysis of data requires more input, the Chatbot requests User for more current information such as their activity, or eating habits etc.	**Step 6**: After the user enters the response, the system analyzes the response to provide a better analysis and identification of illness / disease.
ALTERNATE COURSES:	**Alt-Step 3:** Chatbot confirms with User whether they would like to conduct a specific type of monitoring / checkup.	
	Alt-Step 4: After User enters their response, the system analyzes the response and activates specific IoT sensor(s) for the checkup.	
CONCLUSION:	The use case is concluded once the User is satisfied with the entire checkup.	
POST-CONDITION:	User can take appropriate action with the monitored data and its analysis.	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	Chatbot analyzes responses based on limited key words input by the User.	

Table 3.17 Use Case Description of 'Prompt User Input' Use Case

<table>
<tr><td>USE CASE NAME:</td><td colspan="2">Display any Identified Disease</td></tr>
<tr><td>USE CASE ID:</td><td colspan="2">CH02</td></tr>
<tr><td>PRIMARY ACTOR</td><td colspan="2">Chatbot</td></tr>
<tr><td>OTHER PARTICIPATING ACTORS:</td><td colspan="2">User, Healthcare Application w/ IoT System</td></tr>
<tr><td>DESCRIPTION:</td><td colspan="2">Once the user conducts their checkup, the Chatbot delivers a summary of the monitored data along with any analyzed disease.</td></tr>
<tr><td>PRE-CONDITION:</td><td colspan="2">User completes a whole health checkup.</td></tr>
<tr><td>TRIGGER:</td><td colspan="2">Healthcare system passes monitored data values that is resultant of a certain disease after analysis.</td></tr>
<tr><td rowspan="6">TYPICAL COURSE OF EVENTS:</td><td>Actor Action</td><td>System Response</td></tr>
<tr><td></td><td>Step 1: The system captures the monitored data information of the currently conducted health checkup and conducts the analysis of data..</td></tr>
<tr><td>Step 2: Chatbot displays the monitored body vitals for users to receive instant information.</td><td>Step 3: The system completes analysis of data with or without further user input and identifies a disease and its solution.</td></tr>
<tr><td></td><td>Step 4: The system delivers the information to the healthcare application - Chatbot.</td></tr>
<tr><td>Step 5: Chatbot displays the identified disease for the User with its medical reasoning .</td><td></td></tr>
<tr><td></td><td></td></tr>
<tr><td>ALTERNATE COURSES:</td><td colspan="2">Alt-Step 3: The system completes analysis of data with no further input and identifies no illnesses.</td></tr>
<tr><td></td><td colspan="2">Alt-Step 4: The system delivers no further information to the healthcare application and the Chatbot.</td></tr>
<tr><td></td><td colspan="2">Alt-Step 5: Chatbot displays User is healthy and free of diseases.</td></tr>
</table>

CONCLUSION:	The use case is concluded when a User is provided the name of the disease or told they are healthy.
POST-CONDITION:	User's records in the database and future analysis is updated with the illness identified.
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	The identified diseases delivered to User are limited to the ones input into the database information.

Table 3.18 Use Case Description of 'Display any Identified Disease' Use Case

USE CASE NAME:	Recommend Solution to Identified Illness	
USE CASE ID:	CH03	
PRIMARY ACTOR	Chatbot	
OTHER PARTICIPATING ACTORS:	User, Healthcare Application w/ IoT System	
DESCRIPTION:	If the system does identify a possible disease, the Chatbot also displays the recommended medication or solution for the user.	
PRE-CONDITION:	A disease was identified after a health checkup was completed.	
TRIGGER:	-	
TYPICAL COURSE OF EVENTS:	**Actor Action**	**System Response**
		Step 1: When the system completes a health checkup and the analysis of an identified disease, system searches for the associated solution.
		Step 2: The system finds the solution from the stored information base and forwards it to the healthcare application – Chatbot.

	Step 3: Chatbot displays the recommended solution after displaying the identified disease .	
ALTERNATE COURSES:	**There is no alternative course to this as this is a special use case which is only activated if a disease / illness is actually found.**	
CONCLUSION:	The use case is concluded when a User is provided the recommended solution to a disease.	
POST-CONDITION:	-	
ASSUMPTIONS, CONSTRAINTS AND SPECIFICATIONS:	The recommended solutions analyzed and delivered to User are limited to the ones present in the information base.	

Table 3.19 Use Case Description of 'Recommend Solution to Identified Illness' Use Case

3.3.4 Activity Diagram

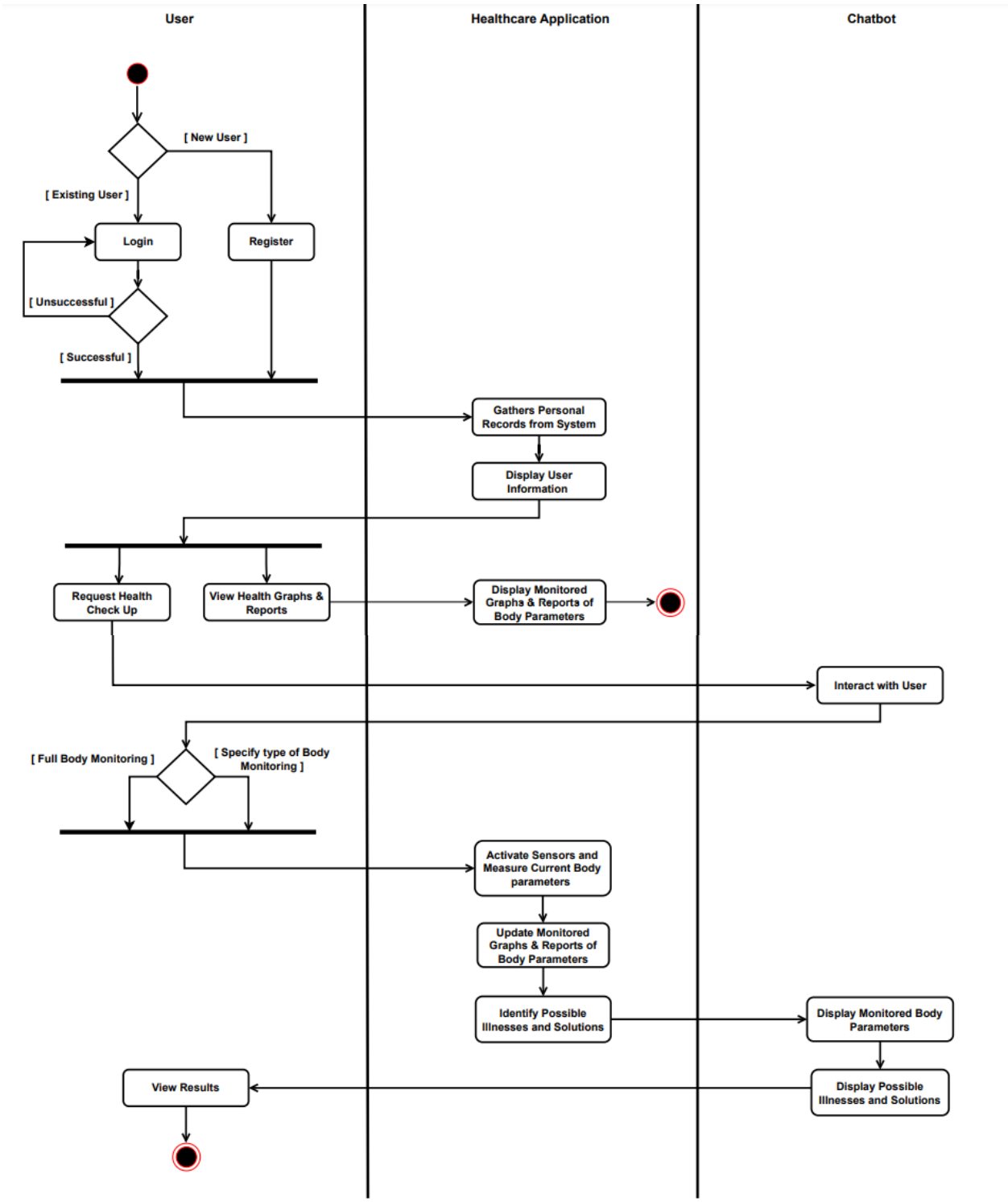


Figure 3.4 Activity Diagram of the Healthcare System

Figure 3.4 presents a basic flow of the healthcare system. All the critical activities performed by each actor are represented in swim lanes. All the different selections and ways the healthcare system actors perform their tasks can be depicted through this activity diagram.

3.3.5 Sequence Diagram

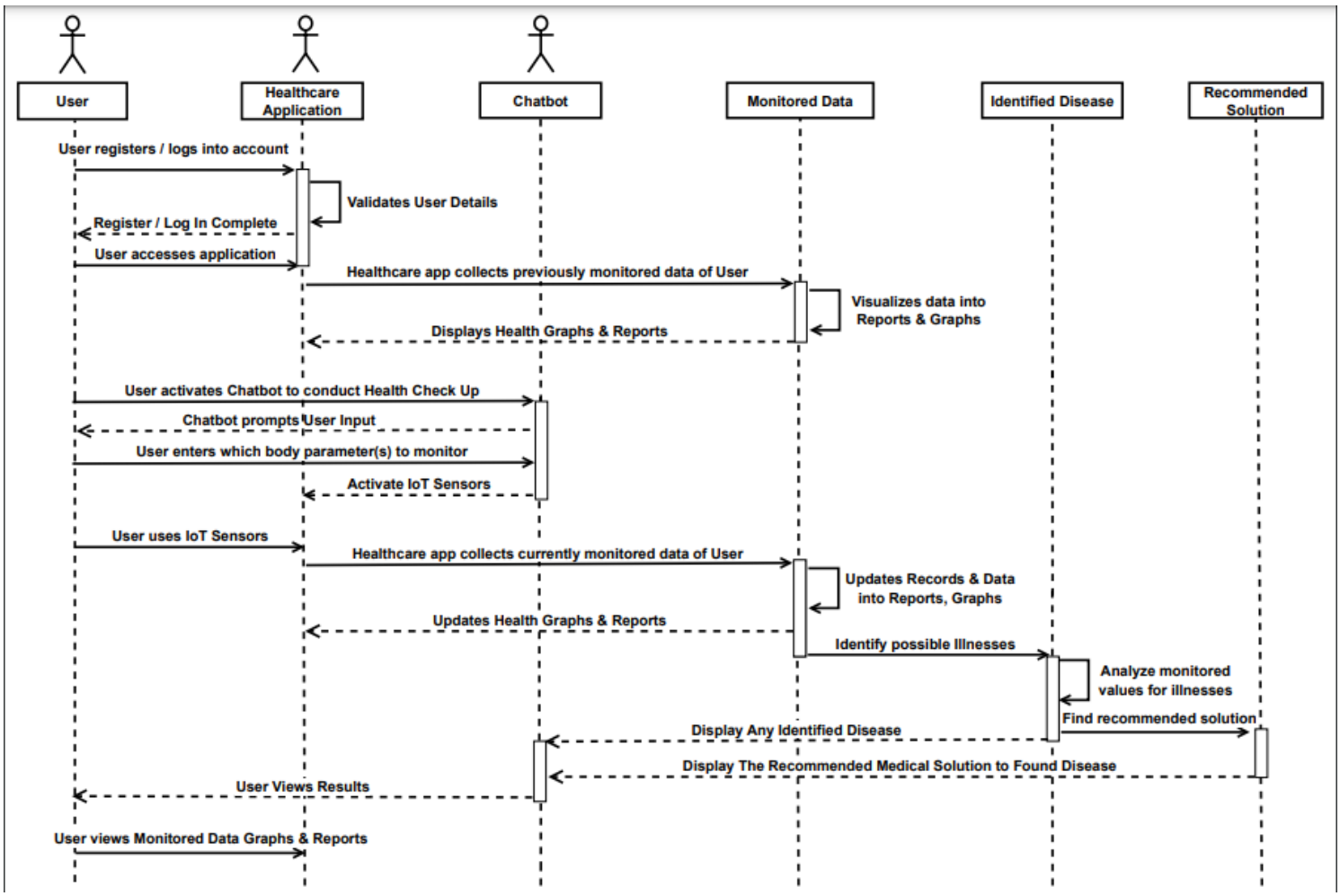


Figure 3.5 Sequence Diagram of Entire Healthcare System

Figure 3.5 shows the complete sequence of events and activities when a user makes use of the healthcare system. The sequence diagram shows the order of each event physically executed by the user as well as the automated tasks triggered by the healthcare application (IoT system) as well as the chatbot, elaborating on the previously designed activity diagram in Figure 3.4.

3.3.6 Class Diagram

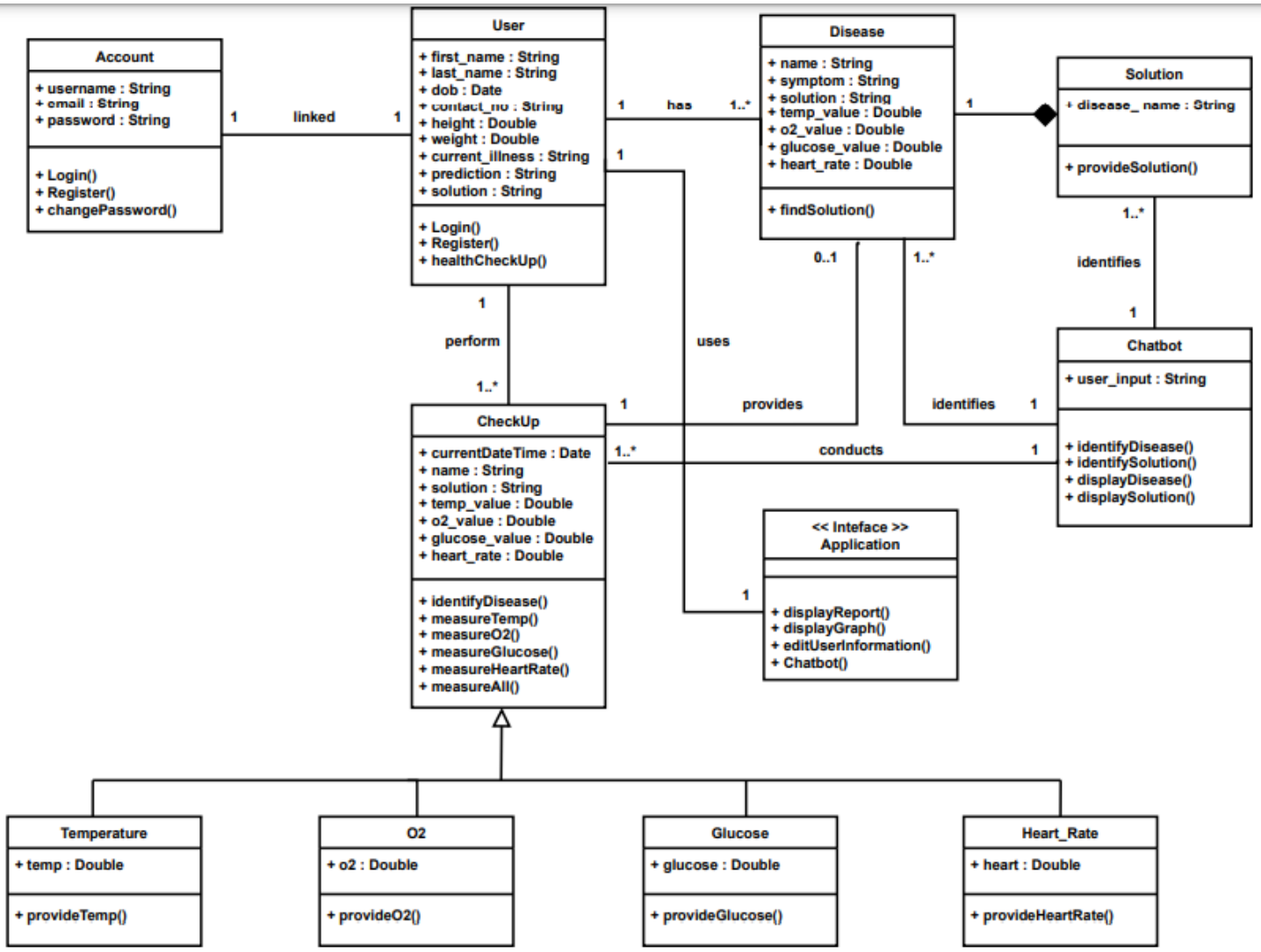


Figure 3.6 Class Diagram of Entire Healthcare System

The class diagram shown in Figure 3.6 details the complete architecture of the overall system. The diagram depicts all the objects – User, Disease, CheckUp etc., and lists down all the functions and attributes as part of the system. The relationships and association of all the objects are shown in the figure above, showing how all the objects will be linked to one another when developing the system. For the proposed healthcare system, the class diagram also utilizes 'Application' as an interface and inherited classes as part of the design for specific monitoring – Temperature, O2, Glucose, Heart_Rate.

3.4 Interface Design

This section of the proposed system's design showcases how the healthcare application's interface will appear to be. As mentioned earlier, the healthcare application is the means by which a user uses the overall system. Thus, the major functionalities and essential requirements of the system are spread across the application. The different interfaces that the user requires with the application are displayed below.

3.4.1 Login Page

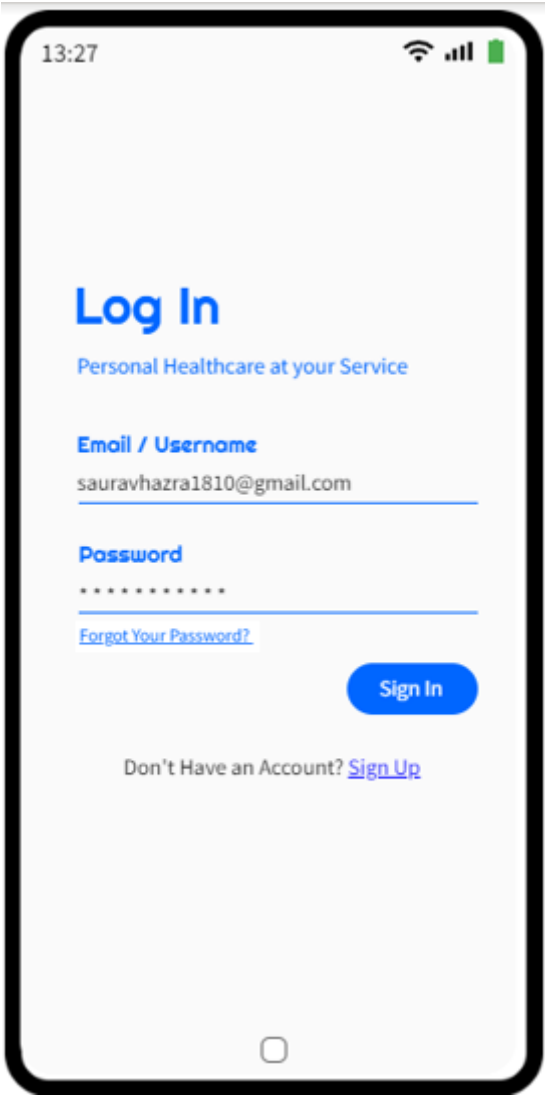


Figure 3.7 Login Page of Healthcare Application

Figure 3.7 shows the first page when the user opens the application for use. The user is asked to enter his / her login details to access the actual functionalities of the healthcare application by entering their registered account's email / username and password. Only when the user successfully enters a matching pair of usernames and the corresponding password, are they able to move to the homepage of the healthcare application. If the user is signed

up and has forgotten their password, they are provided the opportunity to reset their password by clicking the Forgot Your Password link. And finally, if the user is not registered, they are also provided an option to sign up.

3.4.2 Register Page

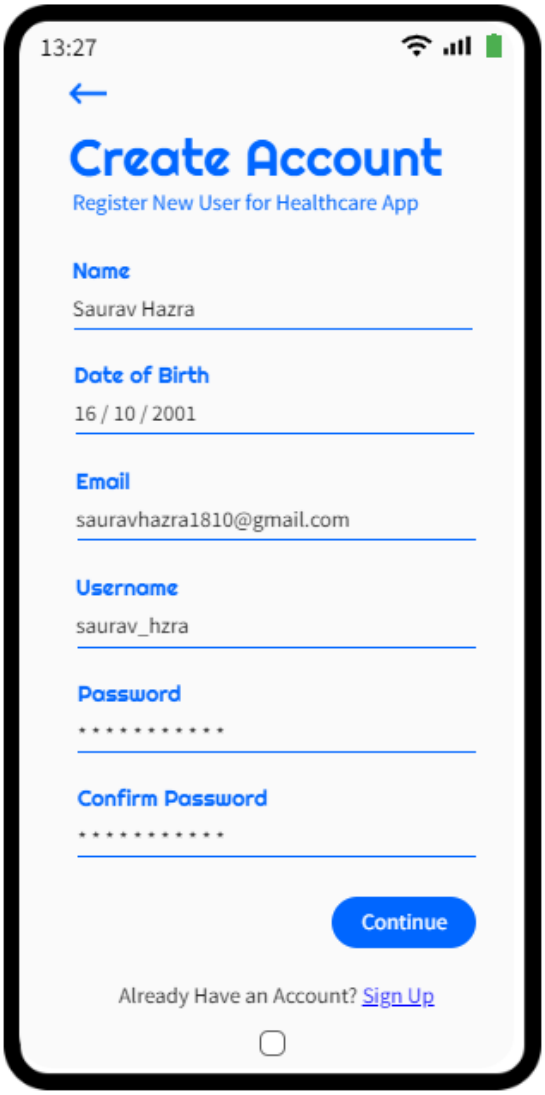


Figure 3.8 Account Registration for Healthcare Application

If the user wishes to register a new account, they click on the Sign-Up link in Figure 3.8 which directs them to the Register page as shown in Figure 3.8. Here, the user can enter their personal details to create an account for accessing the healthcare application. The Register page is actually divided into two interface screens. The first one requesting personal and new account information with fields such as name, email, password etc. Once the user fills in all the fields and presses the 'Continue' button, they are directed to the second registration part as shown in Figure 3.8.

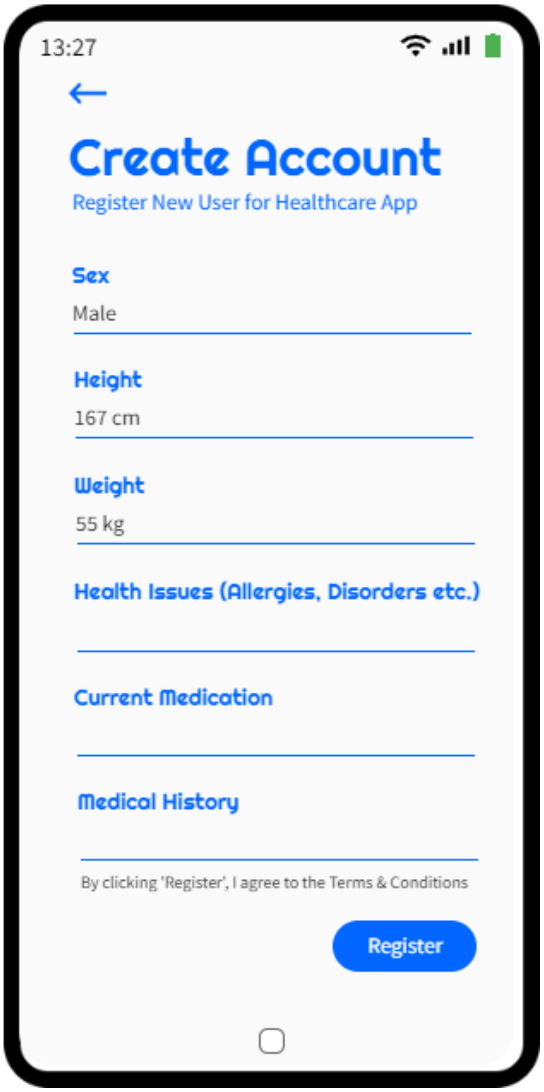


Figure 3.9 Medical Information Registration for Healthcare Application

In the second registration page, the user is prompted to enter his / her necessary medical details that might be helpful later on. Information such as height, weight as well as any current health problems and medication are requested. Once the user completes all the fields, they can complete the registration process to have an account for the healthcare app.

3.4.3 Homepage

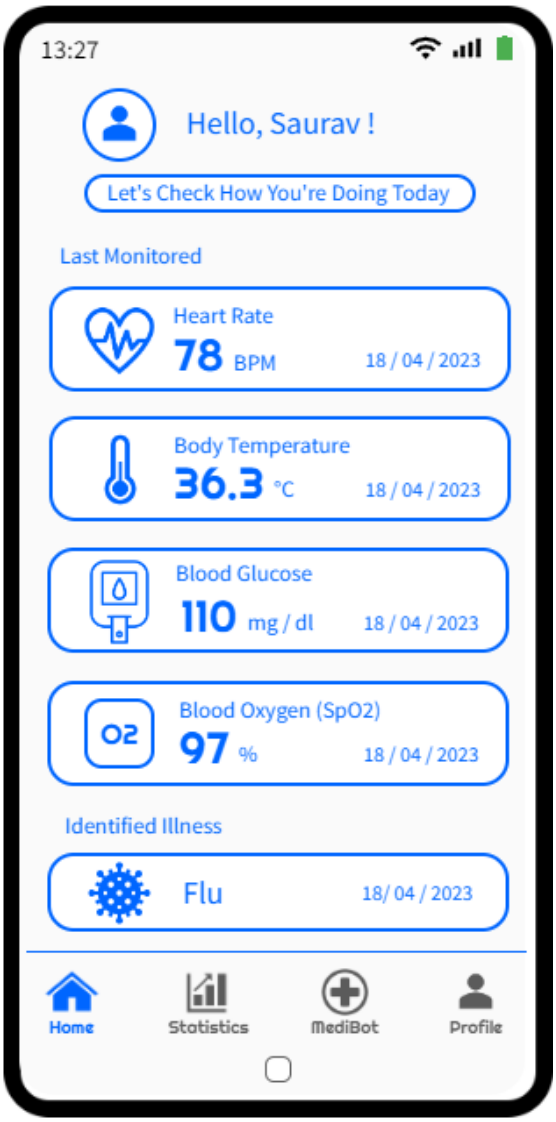


Figure 3.10 Homepage of Healthcare Application

Figure 3.10 shows the homepage of the healthcare application. On the homepage of the healthcare application, the user can mainly view the last recordings of each of the monitored body vitals. The homepage also provides a navigation bar to move to the different pages of the healthcare application.

3.4.4 Monitored Statistics Page

When a user presses on one of the last monitored body vitals on the homepage or presses the 'Statistics' button on the navigation bar, they are directed to the Monitored Statistics page. This particular page is used to display the graphs and reports of each sensor monitored body parameter of the user for tracking and comparison purposes. The page is divided into four separate interface screens – body temperature, heart rate, blood glucose level and blood oxygen level monitoring.

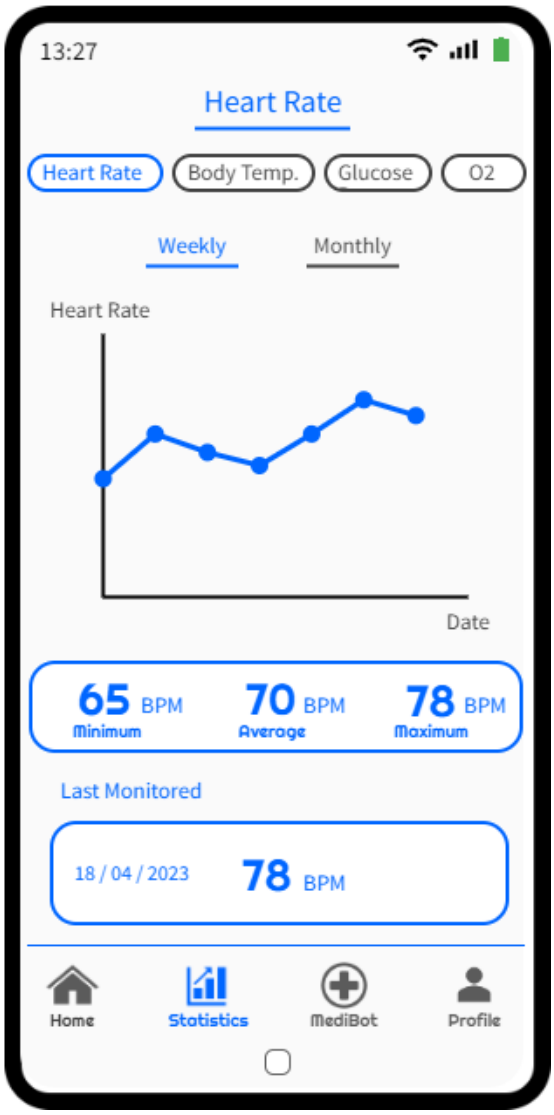


Figure 3.11 Monitored Heart Rate of User in the Monitored Statistics Page

Figure 3.11 shows the heart rate display of the Monitored Statistics page. The user can observe the graph to compare a recent recording with a previous one. By default, the reports and graphs are set to show only a week's recordings, however the user is provided an ability to change it into a monthly recorded value. Figure 3.11 also displays that the interface shows the minimum, average and maximum value in the chosen time period.

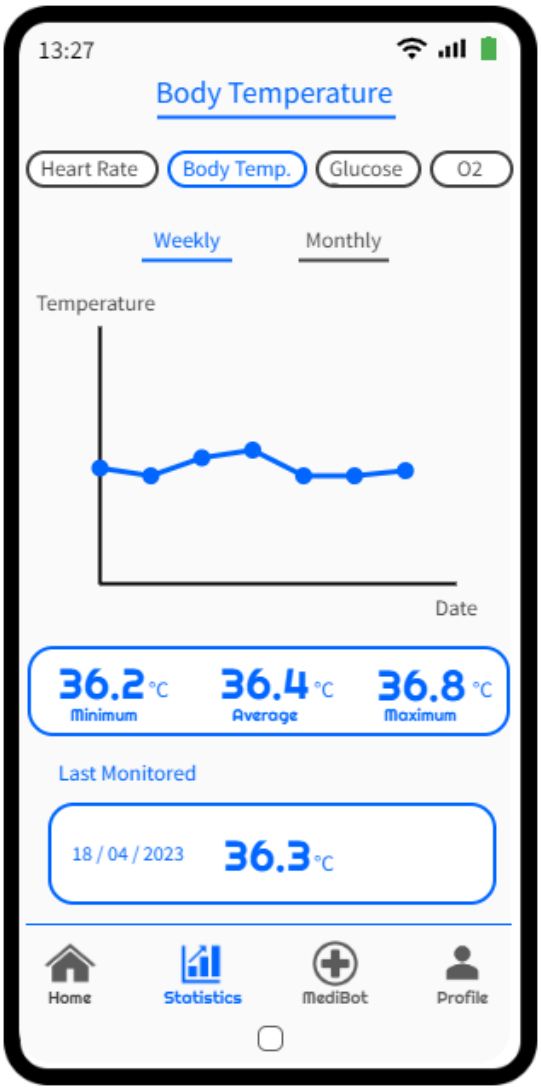


Figure 3.12 Monitored Body Temperature of User in the Monitored Statistics Page

Figure 3.12 displays the monitored body temperature display. The layout and functionalities is the same as the one shown in Figure 3.12 with the only difference being the change of the body vital being measured. In Figure 3.12, the User is viewing the body temperature recordings instead of the heart rate, so the graphs, tables and the other values are changed to the temperature body vital.

Figure 3.13 Monitored Blood Glucose Level of User in the Monitored Statistics Page

Figure 3.13 displays the blood glucose display in the Monitored Statistics Page. Again, the layout and functionalities remain the same with the difference mainly being the change in graphs, tables, and the other values to visualize the blood glucose body vital.

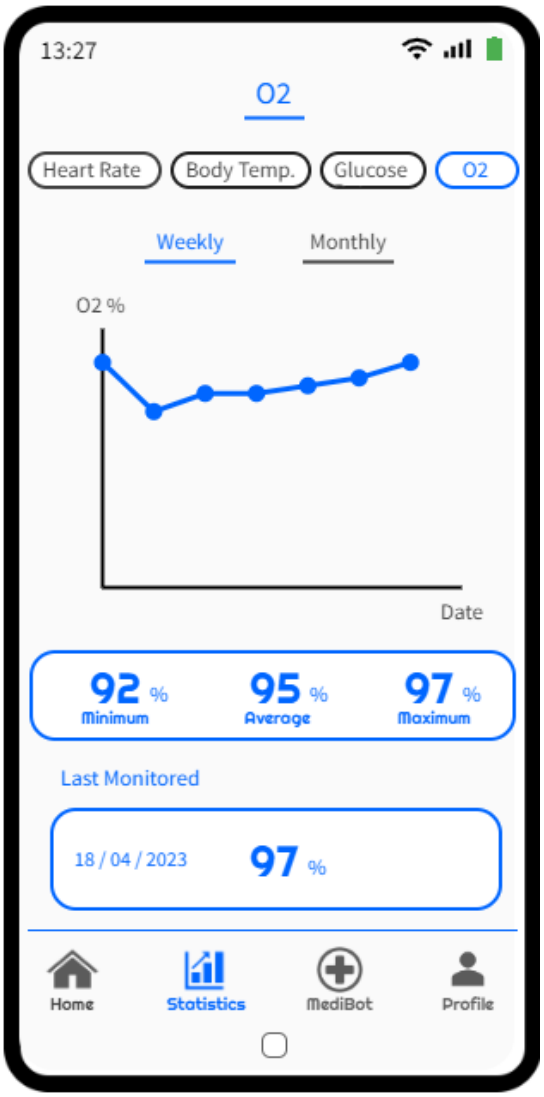


Figure 3.14 Monitored Blood Oxygen Level / Saturation of User in the Monitored Statistics Page

Figure 3.14 displays the blood oxygen levels in the Monitored Statistics Page in the same manner as the other body vitals. The O2 vital page is the last of the Monitored Statistics interfaces. All new recordings are automatically updated into the corresponding graphs and can only be done once a user conducts a health checkup.

3.4.5 Medical Chatbot Page

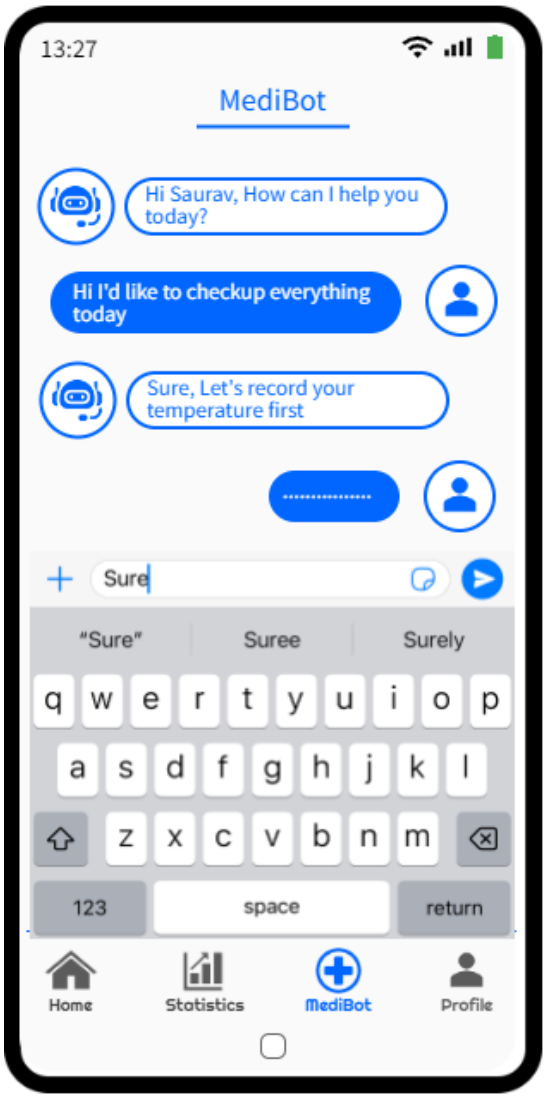


Figure 3.15 Medical Chatbot Page in the Healthcare Application

Figure 3.15 displays the Medical Chatbot interface of the healthcare application that can be accessed when pressing the 'MediBot' button. The interface simply has the user messaging the chatbot for what they wish and the Chatbot responding. The user interacts to start and complete a whole checkup with them, finally receiving the results of the checkup from the Chatbot itself. The Chatbot returns the sensed values alongside any analyzed disease and the recommended treatment.

3.4.6 User Profile Page

The User Profile page displays the users personal and medical information. They can access the page by pressing the 'Profile' button and either view the information or edit something to be changed for . The fields are the same as the ones used while registering as shown in Figure 3 and Figure 3. Also similar

to the Register page, the Profile is separated into the personal and medical sections.

Figure 3.16 Personal User Profile Page in the Healthcare Application

Figure 3.16 shows the Personal display in the User Profile page. The section only has the fields associated to the user's account – email, username, and password along with the user's name and date of birth.

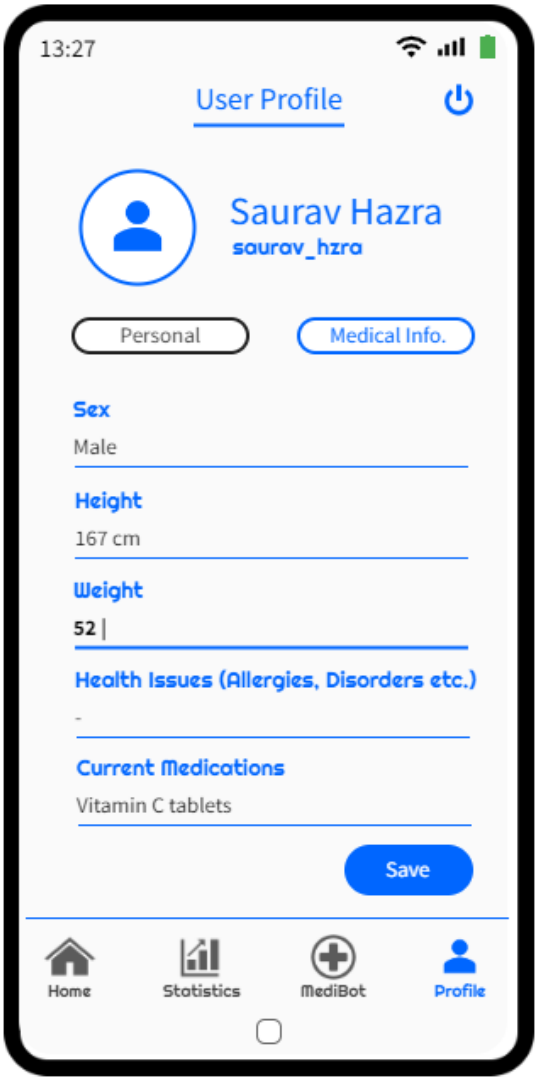


Figure 3.17 Medical User Profile Page in the Healthcare Application

Figure 3.17 on the other hand shows the medical factors that might have changed after registering. To update any information on height, weight or suffering illnesses, this page allows the user to edit their record so that the system can provide more reasonable and harmless identification and recommendations of disease and their solutions.

3.5 Summary

In conclusion, this chapter mapped out all the necessary requirements and indicated how the implementation of the healthcare system will follow the interface and system design diagrams.

CHAPTER 4: SYSTEM IMPLEMENTATION

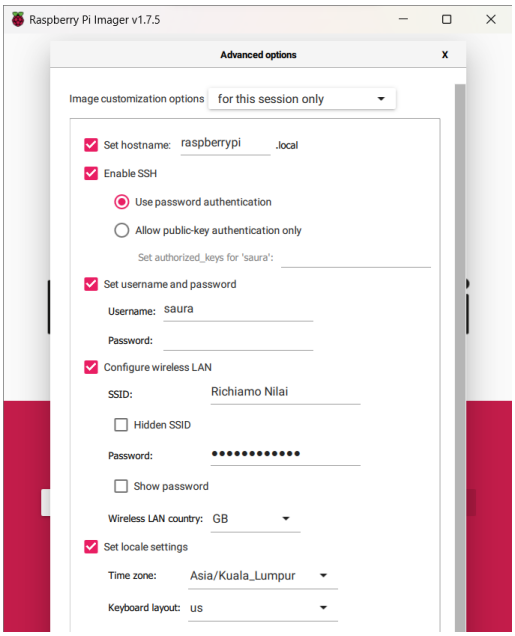
4.0 Overview

The fourth chapter of this project discusses the implementation of the proposed healthcare system. The author describes all the features involved with the IoT hardware setup and each sensor's software functionality. Moreover, the application and the setup of all the software components in combination with each other is thoroughly described, especially the rule-based system and chatbot modules required to perform a health checkup and identify a diagnosis with its recommended solution.

4.1 IoT Development

4.1.1 Raspberry Pi OS Setup

The IoT model of the healthcare system is built on the Raspberry Pi microcomputer. For this project, the Raspberry Pi 4B+ model was obtained, which is the most recent version released. The hardware needs to be setup with the Raspberry Pi OS (previously known as Raspbian OS / Debian) to access the functionalities of the microcomputer. To install the system image, Raspberry Pi provides its own installer known as Raspberry Pi Imager to incorporate the OS to the physical device. The computer board can be configured in various different methods, but for the proposed healthcare system, the author decides to configure it using a wireless Secure Shell (SSH) protocol. With this specific SSH connection, the device can be accessed remotely by a different computer or device in an easy but secure manner. The IoT model needs to be eventually accessed by a mobile application. For this reason, the author is forced to connect to it in a wireless manner.



The Raspberry Pi is configured over a Local Area Network (LAN) as shown in Figure 4.1, such that devices on the same IP address will be able to remotely access the Pi model and in doing so, have the ability to connect to the physical model for a health checkup. In addition to the SSH connection, the author is also required to install and use a software called RealVNC Viewer. Figure 4.2 shows this software that launches a virtual environment for users to alter the Raspberry Pi, implement code and build the desired IoT healthcare model.

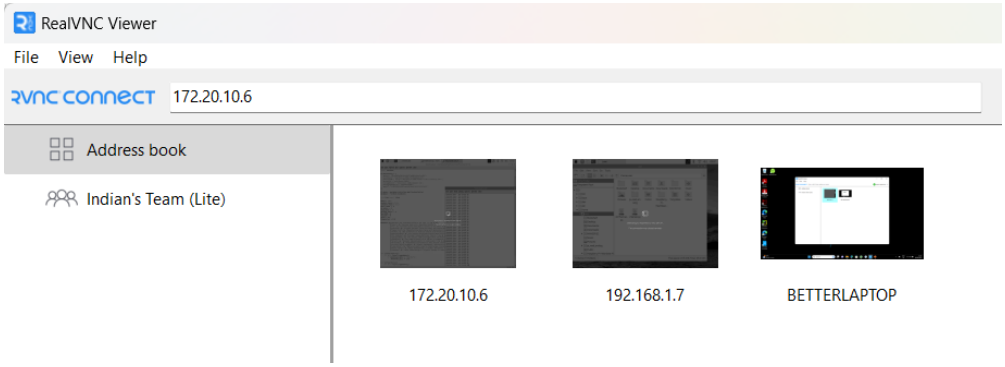
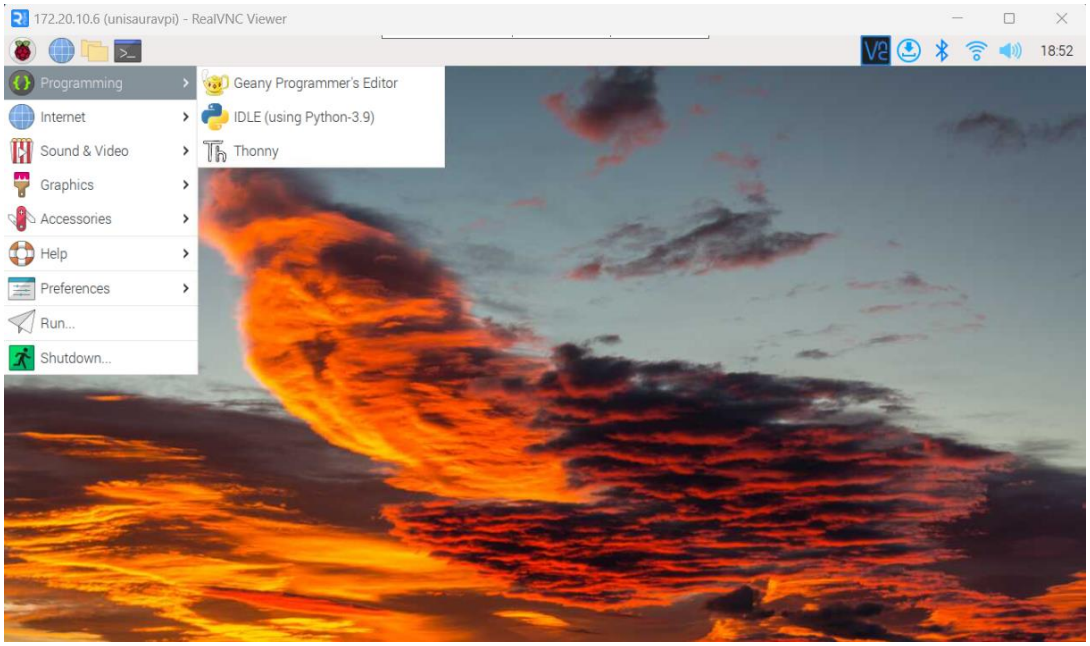


Figure 4.3 shows the default Raspberry Pi operating system. The OS is extremely minimalistic but offers all the required features to develop the required healthcare system. Users are provided the option to program software systems using two integrated development environments (IDE):

- Geanny Programmer's Editor – The flexible IDE allows users to code in different languages using C++, Java, or Python. The text editor has very few dependencies linked to it but can install hundreds of libraries corresponding to each programming language.

- Thonny – Thonny is a straight-forward editor used to develop programs only in Python. The IDE is most useful for basic coding but also provides access to numerous python libraries on the internet to install and implement in the text editor. Even though Thonny is primarily a development environment, users can use this software to save regular text files too.



The author intends to code most of the healthcare system in Python – especially the monitoring and the recommendation / prediction modules. However, instead of using the provided IDEs to code the system, a whole new software package is downloaded. The author downloads the entire Python package separately using the command line terminal by entering the commands shown in Figure 4.4.

```
sudo apt install python3
pip3 install idle
```

The complete Python package is installed since the installation of healthcare sensors will require multiple different functional libraries. The other text editors offered by the operating system have limitations on the specific libraries they can implement. There are no restrictions with the Python package though, since all the libraries needed can be downloaded directly through the terminal and accessed globally. The Python installation also has its own IDE known as IDLE. This is the default IDE installed that allows users to create and save files automatically with the .py extension. The coding performed in IDLE once completed, can be run through the Python Shell. The shell references all the files in the overall Python package and therefore, is the best option to install the sensor classes and apply in this healthcare system.

The final step in the configuration of the operating system is ensuring that Raspberry Pi can connect to external peripherals. The hardware device will have multiple sensors connected to it and thus specific interfacing options must be enabled through the configuration tool. After accessing the Raspberry Pi Software Configuration Tool using the command terminal, the interfacing

options can be changed according to what is required. The SSH and VNC option has already been enabled prior to connecting to the virtual environment. However, the proposed healthcare system requires the enabling of two more interfacing options:

- Serial Peripheral Interface (SPI) – This Raspberry Pi tool is a communication protocol that allows numerous devices to be enabled and connect to each other in a synchronized manner with the Raspberry Pi device.

- Inter-Integrated Circuit Communication (I2C) - This Raspberry Pi tool is a communication protocol that allows multiple devices to be enabled and connect to each other over a two-wire bus.

These interface modules by default are not enabled but need to be changed so that they can be automatically loaded from the respective kernel modules so that the sensors in the healthcare system are readily accessible.

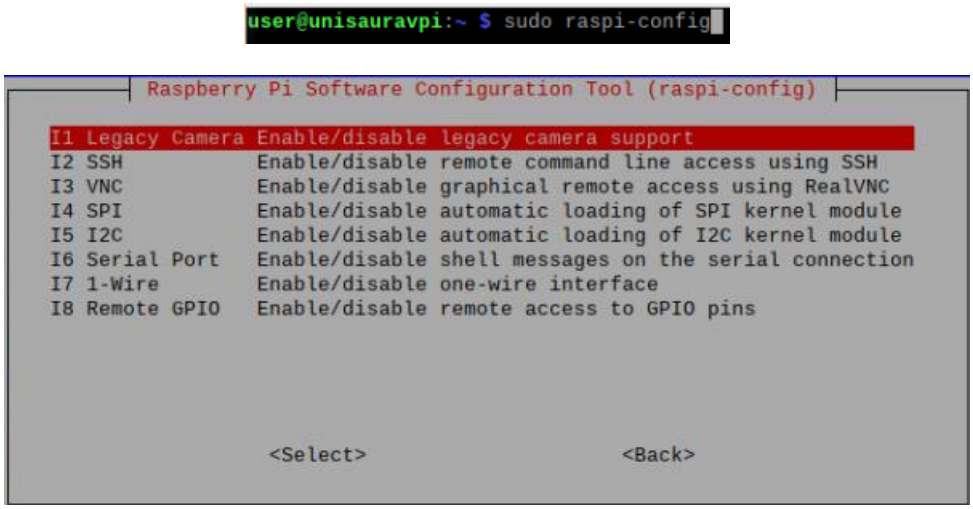


Figure 4.5

4.1.2 Hardware Setup

Once the Raspberry Pi OS is configured, physical connections can be made to the Raspberry Pi model. As described in Chapter 2, the system mainly uses three sensors:

- MAX30102 Sensor – for the measurement of Heart Rate and Blood Oxygen Level

- MLX-90614 Sensor – for the measurement of Body Temperature.

- IR Sensor – for the measurement of Blood Glucose Level.

All these devices must be connected to the same Raspberry Pi board, through the General-Purpose Input Output (GPIO) Pins. Each sensor has its own specific configuration and needs to be connected to particular pins along with

a consistent power supply and a connection to the ground for its operation. Figure 4.6 displays the GPIO pin layout of the Raspberry Pi 4B+ model. Each pin has its own specific function as well as other functions for alternatively configured buses. The model only has two pin locations for a 3.3V voltage supply but more than enough for the ground. The healthcare system uses three sensors as well as other components that require the same voltage supply, i.e., need to be inserted into the same pin and ground connection. To solve this issue, a connection was made from PIN 1 and PIN 6 to a breadboard's positive and negative line to gather more locations for installation of the sensors. The board has a 5V voltage supply that could be used to connect the sensors but PIN 2 and 4 are avoided since the voltage can damage the individual sensor.

	FUNCTION	PIN	PIN	FUNCTION	
3V3	3V3	1	2	5V	5V
GPIO2	SPI3 MOSI/SDA3	3	4	5V	5V
GPIO3	SPI3 SCLK/SCL3	5	6	GND	GND
GPIO4	SPI4 CE0 N/SDA 3	7	8	TXD1/SPI5 MOSI	GPIO14
GND	GND	9	10	RXD1/SPI5 SCLK	GPIO15
GPIO17		11	12	SPI6 CEO N	GPIO18
GPIO27	SPI6 CE1 N	13	14	GND	GND
GPIO22	SDA6	15	16	SCL6	GPIO23
3V3	3V3	17	18	SPI3 CE1 N	GPIO24
GPIO10	SDA5	19	20	GND	GND
GPIO9	RXD4/SCL4	21	22	SPI4 CE1 N	GPIO25
GPIO11	SCL5	23	24	SDA4/TXD4	GPIO8
GND	GND	25	26	SCL4/SPI4 SCLK	GPIO7
GPIO0	SPI3 CE0 N/TXD2/SDA6	27	28	SPI3 MISO/SCL6/RXD2	GPIO1
GPIO5	SPI4 MISO/RXD3/SCL3	29	30	GND	GND
GPIO6	SPI4 MOSI/SDA4	31	32	SDA5/SPI5 CEO N/TXD5	GPIO12
GPIO13	SPI5 MISO/RXD5/SCL5	33	34	GND	GND
GPIO19	SPI6 MISO	35	36	SPI1 CE2 N	GPIO16
GPIO26	SPI5 CE1 N	37	38	SPI6 MOSI	GPIO20
GND	GND	39	40	SPI6 SCLK	GPIO21
	I2C			Ground	
	UART			5V Power	
	SPI			3V3 Power	

Figure 4.6

1. MLX-90614 Sensor Setup

The MLX-90614 Sensor is responsible for measuring the body temperature of an individual. For the sensor to record the temperature values, the device has to be connected to four particular GPIO Pins. MLX-90614 needs to connect to the 3.3V voltage supply and the ground connection. But along with these connections, the sensor also needs to connect to the pins that allow the transfer of Serial Data (SDA) and Serial Clock (SCL). The SDA port is responsible for the transfer of data from the board to the sensor while the SCL port carries the clock signal.

Figure 4.7

The sensor has ports labeled such that the device can be connected with VIN, GND, SCL and SDA. VIN receives the voltage supply and GND connects to the negative line of the breadboard to get the ground connection. The Raspberry Pi by default has only one accessible SDA and SCL port, which are PIN 3 and 5 respectively. The SDA and SCL ports are part of the I2C interface that is automatically loaded after the setup of the environment is done. The SDA and SCL interface are unique since there is only one GPIO port for each of them on a particular bus, which means that the male to female wires need to be connected directly to the specific pins. The sensor inputs and outputs are not read if the wires are connected to the breadboard linking the particular pin or if they are positioned wrong. They must be in direct contact with the breadboard and therefore, the author connects the MLX-90614 according to the GPIO restrictions:

- VIN – PIN 1 (3.3V Breadboard Connection)
- GND – PIN 6 (Ground Breadboard Connection)
- SCL – PIN 5 (GPIO3)
- SDA – PIN 3 (GPIO2)

2. MAX30102 Sensor Setup

The MAX30102 Sensor is responsible for measuring the heart rate and blood oxygen level of an individual. For the sensor to record these body vitals, the device has to be connected to five particular GPIO Pins. Just like the MLX-90614 Sensor, MAX30102 needs to connect to the 3.3V voltage supply, the ground connection, SDA as well as SCL connections. But along with these connections, the sensor also needs to connect to the pin that manages the interruptions in case the sensor does not work or produces faulty results.

Figure 4.8

The sensor has ports labeled such that the device can be connected with VIN, GND, SCL, SDA and INT. VIN receives the voltage supply and GND connects to the negative line of the breadboard to get the ground connection. As mentioned before, the Raspberry Pi by default has only one accessible SDA and SCL port, which are PIN 3 and 5 which are already occupied by the MLX-90614 sensor. The SCL and SDA interface ports need to be connected for the MAX30102 sensor to work but it is not possible to replace the temperature sensor setup since they all need to work together.

In order to solve this issue, a separate bus needs to be configured on the Raspberry Pi. Raspberry Pi offers alternate buses with a unique pin setup just like the regular bus. The different buses offer different locations for the SDA and SCL interface GPIO Pins compared to the default bus. The author uses the command terminal and configures the root file with an alternative bus as displayed in Figure 4.9. The alternative bus is set up with the SCL port set at PIN 16 and SDA port set at PIN 15 so that all the sensors can work on the same board. Finally, the INT port can be configured to any open GPIO pin. The author chooses PIN 7 (GPIO4) for this case since this particular location is not used by other components.

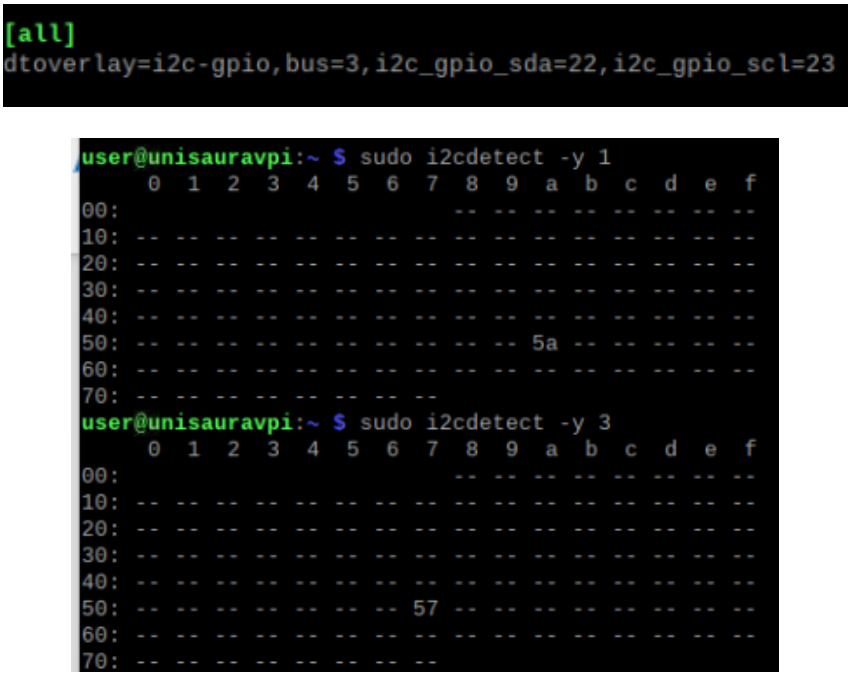


Figure 4.9

In summary the MAX3012 sensor is connected to the Raspberry Pi according to these GPIO restrictions:

- VIN – PIN 1 (3.3V Breadboard Connection)
- GND – PIN 6 (Ground Breadboard Connection)
- SCL – PIN 16 (GPIO23)

- SDA – PIN 15 (GPIO22)
- INT – PIN 7 (GPIO4)

3. IR Sensor Setup

The IR Sensor is responsible for measuring the blood glucose level of an individual. For the sensor to record the blood glucose values, the device has to be connected to four particular GPIO Pins. Similar to the previous sensors, the IR Sensor needs to be connected to the 3.3V voltage supply and the ground to function. The difference though, is the fact that this particular sensor does not require the SCL and SDA ports. Instead, the device needs an Output port that needs to be connected alongside an analogue to digital converter (ADC) to convert analog voltage readings to digital data and retrieve the analogue information once again.

Using an IR sensor to monitor blood glucose levels is not the common / regular method of measuring this body vital. Typically, blood glucose levels are measured when individuals prick their finger, extract blood, and test the collected sample with a glucometer. The traditional method involves pricking a person for measurement which can be a very painful process and could go wrong if performed in a wrong manner. The proposed healthcare system does not stick with the traditional method but implements a growing non-contact method to measure blood glucose levels.

Figure 4.10 shows an IR sensor which is made of an IR emitting LED and an IR photodiode. The IR emitter produces an infrared ray of wavelength 950nm and when a finger is brought close to the IR sensor, the sensor recognizes that a finger is brought close to the device and calculates the glucose level.

Figure 4.10

The infrared ray that is emitted is either absorbed or reflected back. The reflected IR ray is detected by the photodiode and depending on the frequency of the light reflected back, the voltage value going received back to the device is altered. Initially, when the IR ray is first emitted, the voltage is 3.3V as

connected to the VCC port. However, when the finger is placed close to the sensor, the voltage value is reduced since blood absorbs light and the amount / frequency of light reflected back decreases.

The voltage value decreases but to read the changed voltage reading, the IR sensor needs to be connected to a linked analog to digital converter (ADC). The ADC is configured to the Raspberry Pi with the SPI communication interface. The ADC used with the Raspberry Pi model is the MCP3008. The device needs to be connected to specific SPI GPIO pins with particular MCP3008 ports for the conversion of voltage values to accurately occur. The SPI interface involves pins that synchronize the peripheral with the board's Serial Clock (SCLK), input and output transfer through Master In Slave Out (MISO) as well as Master Out Slave In (MOSI) ports. The ADC pins are set up as shown in Figure 4.11:

- MCP3008 VDD – PIN 1 (3.3V Breadboard Connection)
- MCP3008 VREF – PIN 1 (3.3V Breadboard Connection)
- MCP3008 AGND – PIN 6 (Ground Breadboard Connection)
- MCP3008 CLK – PIN 23 (GPIO11 SCLK Connection)
- MCP3008 DOUT – PIN 21(GPIO9 MISO Connection)
- MCP3008 DIN – PIN 19 (GPIO10 MOSI Connection)
- MCP3008 CS / SHDN – PIN 24 (GPIO8 CE0 Connection)
- MCP3008 DGND – PIN 6 (Ground Breadboard Connection)

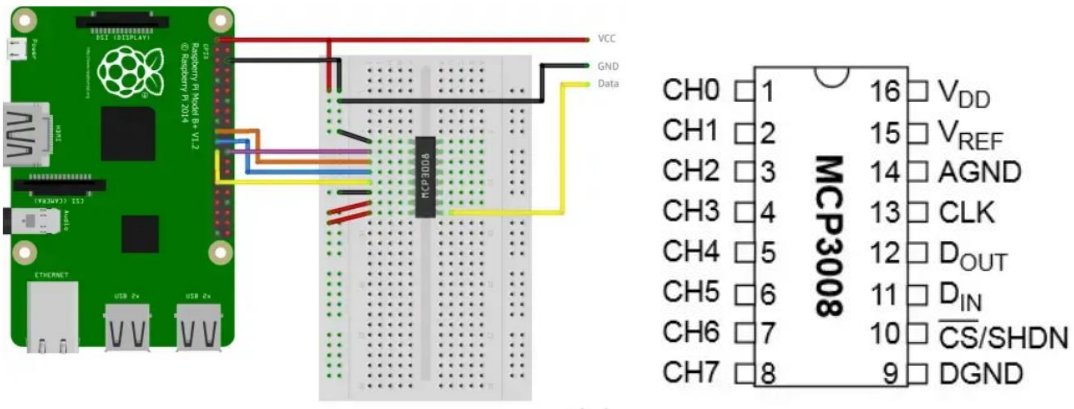


Figure 4.11

The IR Sensor translates the voltage recordings into digital information that can be pushed into a linear regression equation to calculate the glucose level. This equation is extracted after referencing multiple published papers. According to Reddy (2020), the regression model is the most accurate method of plotting

voltage to glucose concentration levels. There are a lot of calculation with Sum of Mean Squares as well as Regression Coefficients but at the end of the day the formulae that was equated was:

y = Max Voltage * Voltage of Signal * 18.0182

To construct the regression model, the regression line must be calculated. The general linear regression model is given by:

$$Y = mX + b \quad (7)$$

Where Y: The predict blood glucose concentration.
X: The voltage of the PPG signal.

The (m) and (b) are the regression coefficients which is given by:

$$m = \frac{X'_v Y'_v - (X_v Y_v)'}{(X'_v)^2 - (X_v^2)'} \quad (8)$$

$$b = mX'_v + Y'_v \quad (9)$$

Where X'_v: the mean of vector voltages PPG readings.
Y'_v: the mean of vector reals GCB readings.

The 18.0182 in the equation is required for the conversion of units. The regular equation calculates the value in mmol / l. However, the application needs the value in mg / dl.

4.1.3 Software Installation of Sensors

1. Body Temperature Measurement

To work with the MLX90614 sensor using the Adafruit library, the installation of the Adafruit_CircuitPython_MLX90614 library to the root environment is needed. The library provides the required support for the sensor to work with python.

When coding the Adafruit library is imported alongside the board and busio packages. These files provide an abstraction for hardware pin names and configuration and allows easy interaction with pins on different microcontroller boards without worrying about the specific pin names. Once the code initializes the I2C and the system bus communication, an instance of the MLX90614 sensor can be created to read the object temperature after a resting duration of 7 seconds to boot up and save / print the readings appropriately.

- Here i2c is defined as the instance of the busio.I2C class, which represents the I2C bus.
- board.SCL and board.SDA are the pins on the development board connected to the SCL (clock) and SDA (data) lines of the I2C bus, respectively.
- mlx is defined as the Adafruit_MLX90614 instance.

```
def temp():

    import board
    import busio as io
    import RPi.GPIO as GPIO
    import adafruit_mlx90614

    GPIO.cleanup()
    i2c = io.I2C(board.SCL, board.SDA, frequency=100000)
    mlx = adafruit_mlx90614.MLX90614(i2c)

    print("Measuring Temperature Now : ")
    sleep(7)

    # ambientTemp = "{:.2f}".format(mlx.ambient_temperature)
    # targetTemp = "{:.2f}".format(mlx.object_temperature)

    ambientTemp = round(mlx.ambient_temperature, 2)
    targetTemp = round(mlx.object_temperature, 2)

    sleep(3)
    GPIO.cleanup()

    # print("Ambient Temperature: ", ambientTemp, " C")
    print("Target Temperature: ", targetTemp, " C")
    database.child("Model").child("001").child("Temperature").set(targetTemp)
```

2. Heart Rate & Oxygen Level Measurement

In this code, we create a MAX30102 class that handles the sensor initialization and data reading. We use the interrupt pin (GPIO4 in this example) to receive notifications from the MAX30102 sensor when new data is ready to be read. The MAX30102 class has an __init__ method that initializes the I2C communication, sets up the interrupt pin, and creates an instance of the MAX30102 sensor using the adafruit_max30102.MAX30102 class with the interrupt parameter set to the interrupt pin.

```
class MAX30102():
    # by default, this assumes that physical pin 7 (GPIO 4) is used as interrupt
    # by default, this assumes that the device is at 0x57 on channel 3
    def __init__(self, channel=3, address=0x57, gpio_pin=7):
        print("Channel: {0}, address: 0x{1:x}".format(channel, address))
        self.address = address
        self.channel = channel
        self.bus = smbus.SMBus(self.channel)
        self.interrupt = gpio_pin

        # set gpio mode
        GPIO.setmode(GPIO.BOARD)
        GPIO.setup(self.interrupt, GPIO.IN)

        # self.reset()

        sleep(1)  # wait 1 sec

        # read & clear interrupt register (read 1 byte)
        reg_data = self.bus.read_i2c_block_data(self.address, REG_INTR_STATUS_1, 1)
        # print("[SETUP] reset complete with interrupt register0: {0}".format(reg_data))
        self.setup()
        # print("[SETUP] setup complete")
```

The read_i2c_block_data method checks if the interrupt pin is active (high), indicating that new data is available. new data is available, it reads the heart rate and SpO2 data using the read_sequential () method and prints the

readings to the console. The code will continuously read heart rate and SpO2 data from the MAX30102 sensor whenever new data is available and if the values of the heart rate and SpO2 fall with an appropriate range, they are printed and stored to the storage.

```
def o2():

    import board
    import busio
    import digitalio
    import RPi.GPIO as GPIO

    GPIO.cleanup()

    import max30102
    import hrcalc

    count = 1
    o2Total = 0
    heartTotal = 0
    avgO2 = 0
    avgHeart = 0

    print("Measuring Heart Rate and O2 Now : ")
    sleep(3)

    m = max30102.MAX30102()
    hr2 = 0
    sp2 = 0

    while count <= 5:
        red, ir = m.read_sequential()
        hr,hrb,sp,spb = hrcalc.calc_hr_and_spo2(ir, red)

        print("hr detected:",hrb)
        print("sp detected:",spb)

        if(hrb == True and hr != -999):
            hr2 = int(hr)
            print("Heart Rate : ", hr2)
        if(spb == True and sp != -999):
            sp2 = int(sp)
            print("SPO2       : ", sp2)

        if (hr2 > 50 and sp2 > 70 and hr2 < 140):
            o2Total += sp2
            heartTotal += hr2
            count += 1

    avgO2 = round(o2Total / 5)
    avgHeart = round(heartTotal / 5)

    print("Final Detected Heart Rate : ", avgHeart)
    print("Final Detected 02 Saturation : ", avgO2)
    database.child("Model").child("001").child("Heart Rate").set(avgHeart)
    database.child("Model").child("001").child("O2").set(avgO2)
```

3. Blood Glucose Measurement

Before proceeding, it is necessary to install the RPi.GPIO library alongside the adafruit-circuitpython-mcp3xxx library in the command terminal to import the specific libraries. In this code, a glucose definition is created that handles the ADC initialization and voltage reading. This method initializes the SPI communication, sets up the cs (Chip Select) pin and GPIO25 input pin, creates an instance of the MCP3008 ADC, along with an instance of the analog input on the specified channel. Once the code initializes the SPI communication channel, an instance of the IR sensor can be created to read the object temperature after a resting duration of 3 seconds to boot up and save / print the readings appropriately.

- Here spi is defined as the instance of the busio.SPI class, which represents the I2C bus.

- board.SCK, board.MISO and board.MOSI are the pins on the development board connected to the SCK (clock), MISO and MOSI (data) lines of the SPI bus, respectively.

- mcp is defined as the Adafruit_MCP3008 instance.

The read voltage method reads the raw ADC value from the IR sensor and converts it to voltage by scaling it to the reference voltage (3.3V in this case). After the voltage reading is read when a signal is received (a finger is brought close to read the blood glucose level) from the GPIO input, the voltage recording is put into the glucose conversion equation discussed earlier:

Glucose = 3.3 * Voltage Recorded * 18.0182

The code checks if the individual is fasting and then appropriately adds 40 to the glucose reading calculated if they are fasting. And at the end of the process, stores and prints the glucose value. The value is recorded 5 times and then the average value of the glucose is recorded, rounding the value to 2 decimal places for a precise glucose reading.

```
def glucose():

    import board
    import busio
    import digitalio
    import adafruit_mcp3xxx.mcp3008 as MCP
    import RPi.GPIO as GPIO
    from adafruit_mcp3xxx.analog_in import AnalogIn

    spi = busio.SPI(clock=board.SCK, MISO=board.MISO, MOSI=board.MOSI)
    cs = digitalio.DigitalInOut(board.D5)
    mcp = MCP.MCP3008(spi, cs)
    channel = AnalogIn(mcp, MCP.P0)

    GPIO.setmode(GPIO.BCM)
    GPIO.setup(25, GPIO.IN)
    vcount = 1
    v = 0
    avg_volt = 0
    not_fasting = False

    print("Measuring Blood Glucose Now : ")
    sleep(3)

    #Setup
    while vcount <= 5:
        sensor = GPIO.input(25)

        if sensor == 1:   # There is no signal
            print("No Object is Detected")
            sleep(1)
        elif sensor == 0: # There is a signal
            print("Finger Detected")
            v += channel.voltage
            print('ADC Voltage: ' + str(channel.voltage) + ' V')
            print("")
            vcount += 1
            sleep(3)

    avg_volt = v / 5.0

    glucose = round(3.3 * avg_volt * 18.0182, 2)
    if not_fasting == True:
        glucose += 40

    print(str(glucose) + " mg/dl")
    database.child("Model").child("001").child("Glucose").set(glucose)
    GPIO.cleanup()
```

4.2 Firebase Realtime Database

All the healthcare related data in this project is stored in a real-time database on Firebase. It is designed to store user-specific health data, monitored health readings over time, as well as personal information for individual users. The database can be expanded and updated dynamically based on user interactions and health data collection which is critical functionality while conducting health checkups or a simple monitoring processes for the application to work with the model. The structure of the Firebase database can be broken down into:

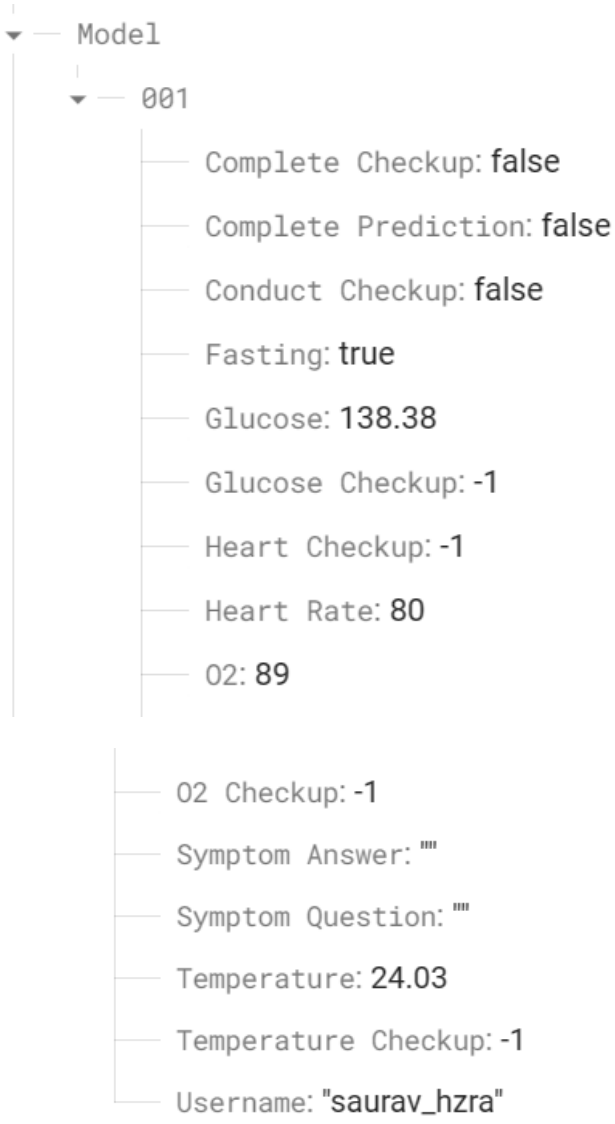
- Model
- Monitored Data
- Users

1. Model

This is a node that contains information related to the medical model used for health assessment. It has a child node with key "001" that holds various attributes related to the health assessment model for a specific user. The "Model" node is designed to keep track of the health assessment process for each individual user. It stores information such as the user's health readings, checkup status, and symptom-related details during the assessment. The attributes specific to a single model are depicted below:

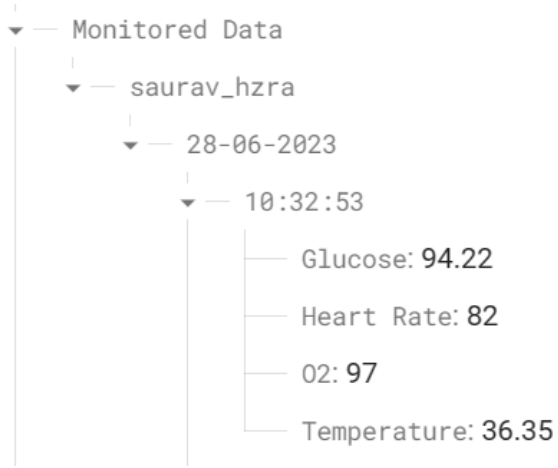
- "Complete Checkup": A Boolean value (true or false) indicating whether the user has completed a full health checkup.
- "Complete Prediction": A Boolean value indicating whether the user's health condition prediction has been completed.
- "Conduct Checkup": A Boolean value indicating whether a checkup is being conducted for the user.
- "Fasting": A Boolean value indicating whether the user is fasting before the health assessment.

- "Glucose": The recorded glucose level of the user during the health assessment.
- "Glucose Checkup": An integer value that stores the result of the glucose checkup.
- "Heart Checkup": An integer value that stores the result of the heart checkup.
- "Heart Rate": The recorded heart rate of the user during the health assessment.
- "O2": The recorded oxygen level of the user during the health assessment.
- "O2 Checkup": An integer value that stores the result of the oxygen level checkup.
- "Symptom Answer": A string that stores the user's answer to a symptom-related question.
- "Symptom Question": A string that represents the question related to the user's symptoms.
- "Temperature": The recorded body temperature of the user during the health assessment.
- "Temperature Checkup": An integer value that stores the result of the body temperature checkup.
- "Username": The username of the user associated with this model entry.



2. Monitored Data

This node contains monitored health data for different users. Each user has a unique key (e.g., "saurav_hzra") that corresponds to their username. Under each user, there are multiple child nodes representing different dates (e.g., "12-07-2023", "14-07-2023", etc.). Each date node has child nodes representing the time of the health data entry (e.g., "10-02-07", "03-17-25", etc.). Under each time node, there are attributes like "Glucose," "Heart Rate," "O2," and "Temperature" to store the corresponding health data readings.



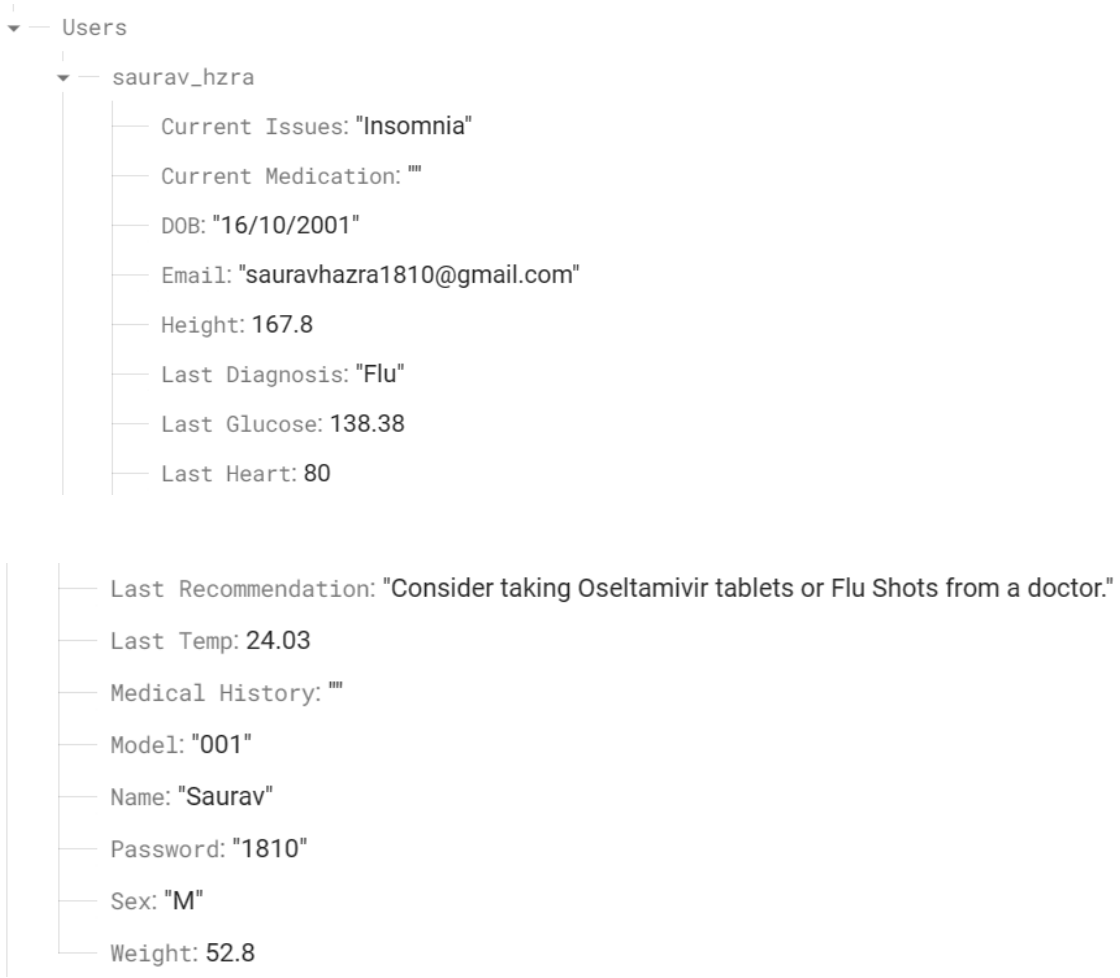
3. Users

This node contains information about different users. Each user has a unique key (e.g., "saurav_hzra", "taha_19") that corresponds to their username. Under each user, there are various attributes related to their personal information, medical history, current health issues, and last diagnosis that are essential to be retrieved by the healthcare application.

The attributes under the "Users" node are described below:

- "Current Issues": Holds the current health issues faced by the user.
- "Current Medication": Information about the user's current medications.
- "DOB": Date of Birth of the user.
- "Email": User's email address.
- "Height": Height of the user in centimeters.
- "Last Diagnosis": Last health diagnosis for the user.
- "Last Glucose": Last recorded glucose level for the user.
- "Last Heart": Last recorded heart rate for the user.
- "Last O2": Last recorded oxygen level for the user.
- "Last Recommendation": Last medical recommendation for the user.
- "Last Temp": Last recorded temperature for the user.
- "Medical History": Information about the user's medical history.
- "Model": Refers to the model key (e.g., "001") for health assessment.

- "Name": User's name.
- "Password": User's password (Note: It is generally not recommended to store passwords in plaintext like this in a real application. Proper authentication mechanisms should be used.)
- "Sex": User's gender (e.g., "M" for Male, "F" for Female).
- "Weight": Weight of the user in kilograms.



4.3 Rule Based System

4.3.1 Monitored Body Vital Values

Since the rule-based system is coded in Python, `pyrebase` library is required to authenticate and interact with a Firebase Realtime Database to perform read and write operations – to store the monitored values, healthcare diagnosis identifications and recommended solutions. The `pyrebase` library is a Python wrapper for the Firebase API, which allows you to interact with Firebase services in your Python application.

The breakdown of the configuration parameters is as follows:

- `apiKey`: Firebase project's API key, which is used for authentication and authorization when making requests to the Firebase services.
- `authDomain`: The domain used for authentication.
- `databaseURL`: The URL to the Firebase Realtime Database. This parameter tells `pyrebase` where to send the data requests.
- `projectId`: The unique identifier for the Firebase project.
- `storageBucket`: The bucket where the Firebase Storage files are stored.
- `messagingSenderId`: The sender ID for Firebase Cloud Messaging, which is used for sending push notifications.
- `appId`: The unique identifier for the Firebase application.

```
import pyrebase

firebaseConfig = {
  "apiKey": "AIzaSyBFwf0H1kuLdGKXJlspQ63b4jgafjcyUDc",
  "authDomain": "healthcare-c3b0a.firebaseapp.com",
  "databaseURL": "https://healthcare-c3b0a-default-rtdb.firebaseio.com/",
  "projectId": "healthcare-c3b0a",
  "storageBucket": "healthcare-c3b0a.appspot.com",
  "messagingSenderId": "1032203371002",
  "appId": "1:1032203371002:web:0532dce790703718622443"
};
```

```
symptomfile = open("/home/user/Healthcare/Symptoms", "r")

sexVal = database.child("Users").child(user).child("Sex").get()
sex = sexVal.val()

heightVal = database.child("Users").child(user).child("Height").get()
height = heightVal.val()

weightVal = database.child("Users").child(user).child("Weight").get()
weight = weightVal.val()

bmi = weight / pow((height/100), 2)
bmi_rounded = round(bmi, 2)

historyVal = database.child("Users").child(user).child("Medical History").get()
family_history = historyVal.val()

issuesVal = database.child("Users").child(user).child("Current Issues").get()
current_problem = issuesVal.val()

heartRateVal = database.child("Model").child("001").child("Heart Rate").get()
heart_rate = heartRateVal.val()

o2LevelVal = database.child("Model").child("001").child("O2").get()
oxygen = o2LevelVal.val()

bodyTempVal = database.child("Model").child("001").child("Temperature").get()
temperature = bodyTempVal.val()

bloodGlucoseVal = database.child("Model").child("001").child("Glucose").get()
glucose = bloodGlucoseVal.val()

fastingVal = database.child("Model").child("001").child("Fasting").get()
fasting = fastingVal.val()

compPrediction = database.child("Model").child("001").child("Complete Prediction").get()
complete_prediction = compPrediction.val()
```

The rule-based system is designed and referenced through a list called medibase. The list is designed as a database that holds medical information for different health conditions. Each element in the list represents a specific health condition and includes various details related to the condition, such as a description, treatment options, and scores.

Each element in the medibase list is a sub list containing the following elements:

- The name of the Health Condition / Diagnosis (e.g., "Healthy," "Fever," "Flu," etc.).
- A score for the health condition (initialized to 0).
- The Treatment Recommendations.
- An integer indicating the line count of the Symptoms file (currently initialized to 0, but later updated when going through the file).
- An integer indicating the number of symptoms associated with a health condition (currently initialized to 0, but later updated).

- An integer indicating the number of symptoms associated with a health condition (currently initialized to 0, but later updated).

```
medibase =[["Healthy", 0, "Continue having a fun life :)", 0, 1, 0],
           ["Fever", 0, "Consider Aspirin or Ibuprofen Tablets for treatment.", 0, 0, 0],
           ["Flu", 0, "Consider taking Oseltamivir tablets or Flu Shots from a doctor.", 0, 0, 0],
           ["Covid-19", 0, "Isolate yourself, Drink hot water and inform medical facility.", 0, 0, 0],
           ["Common Cold", 0, "Oxymetazoline Nasal Spray with Benzonatate tablets.", 0, 0, 0],
           ["Asthma", 0, "Get an Aerosol or Ventolin Inhaler with Prednisolone tablets.", 0, 0, 0],
           ["Hyperglycemia", 0, "You require Prescribed Insulin Shots and Glucotrol tablets.", 0, 0, 0],
           ["Pre Diabetes", 0, "Consider taking Metformin Tablets.", 0, 0, 0],
           ["Hypoglycemia", 0, "You require Timely Glucagon Shots and Metformin tablets.", 0, 0, 0],
           ["Pneumonia", 0, "You require Penicillin G tablets or Augmentin tablets.", 0, 0, 0],
           ["Stress", 0, "Consider taking Aspirin to reduce pain.", 0, 1, 0],
           ["Arrhythmia", 0, "Propafenone and Acebutolol Beta Blocker tablets", 0, 0, 0],
           ["Coronary Artery Disease", 0, "Niacin supplements, Ranolazine and Acebutolol Beta Blocker tablets", 0, 0, 0],
           ["Lung Cancer", 0, "Sadly you might require chemotherapy. Cisplatin and Carboplatin can help but you need a doctor.", 0, 0, 0],
           ["Breast Cancer", 0, "Sadly Chemotherapy is needed. Adriamycin, Cytoxan and Taxotere will help from the doctor.", 0, 0, 0],
           ["Leukemia", 0, "Sadly Chemotherapy or Targeted Therapy is required. Doctors might also use immunotherapy.", 0, 0, 0]]

linecount = 0
for line in symptomfile:
    symp = []
    symp_count = 0
    linecount += 1
    for i in range(len(medibase)):
        medibase[linecount - 1][3] = linecount
        if (re.findall(medibase[i][0], line)):
            cleaned_line = line.replace(medibase[i][0] + ":", "")
            cleaned_line = cleaned_line.replace("\n", "")
            symp.append(re.split(",", cleaned_line))
            for symptom in range(len(symp[0])):
                symp_count += 1
```

When the body vitals have been monitored, they are first evaluated to check whether the certain diagnoses are satisfied.

- The code defines a series of functions for incrementing the score of specific medical conditions in the **medibase** list.
- Each function loops through the **medibase** list and increases the score for the corresponding medical condition if it finds a match.
- The score increment seems to be constant for each function (e.g., 12.5 or 10).

```
def inchealthyScore():
    for i in range(len(medibase)):
        if (medibase[i][0] == "Healthy"):
            medibase[i][1] += 12.5

5 usages
def incfeverScore():
    for i in range(len(medibase)):
        if (medibase[i][0] == "Fever"):
            medibase[i][1] += 12.5

6 usages
def incfluScore():
    for i in range(len(medibase)):
        if (medibase[i][0] == "Flu"):
            medibase[i][1] += 12.5

6 usages
def inccovidScore():
    for i in range(len(medibase)):
        if (medibase[i][0] == "Covid-19"):
            medibase[i][1] += 12.5
```

1. Healthy Individual

The first diagnosis to check while conducting the checkup must always be whether the individual is healthy or not. An individual who is healthy has a body temperature from anywhere around 36.3ºC to 37.5ºC, a resting heart rate value from 60 BPM to 100 BPM, an oxygen level of 95% or higher and a glucose level ranging anywhere from 80 to 120 mg /dl. The healthy blood glucose range mentioned prior was for an individual who conducted a health checkup while fasting, i.e., not having eaten anything in the past 2 hours. If an individual completes a checkup after recently eating, the healthy blood glucose level must be within 120 to 180 mg /dl.

```
if(temperature >= 36.4 and temperature <= 37.5):
    inchealthyScore()
if(oxygen >= 95):
    inchealthyScore()
if (heart_rate >= 60 and heart_rate <= 100):
    inchealthyScore()
if(not_fasting == True):
    if(glucose >= 120 and glucose <= 180):
        inchealthyScore()
else:
    if (glucose >= 80 and glucose <= 120):
        inchealthyScore()
```

2. Fever

An individual will be identified to have Fever as their likely diagnosis if first of all their body temperature is greater or equal to 37.8ºC. If the oxygen level is equal to or less than 96% and the heart rate is greater than or equal to 90 BPM, the probability score of being diagnosed with Fever is increase. The Blood Glucose vital does not really change if the person has fever so the glucose range for being diagnosed with Fever is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg /dl if the person is not fasting.

```
if(temperature >= 37.8):
    incfeverScore()
if(oxygen <= 96):
    incfeverScore()
if (heart_rate >= 90):
    incfeverScore()
if(not_fasting == True):
    if(glucose >= 120 and glucose <= 180):
        incfeverScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incfeverScore()
```

3. Flu

An individual will be identified to have Flu as their likely diagnosis if first of all their body temperature falls below 36.2ºC or the temperature is above 37.8ºC. If the oxygen level turns out to be less than 95%, the score is increased further. The heart rate in this case is different for males and females. If the individual is a male and the heart rate is greater than 100 BPM, the probability score of being diagnosed with Fever is increased. The diagnosis score for females increases if their heart rate turns out be greater than 82 BPM. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Flu is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature > 37.8 or temperature < 36.2):
    incfluScore()
if(oxygen < 95):
    incfluScore()
if(sex == "Male"):
    if (heart_rate > 100):
        incfluScore()
elif (sex == "Female"):
    if (heart_rate > 82):
        incfluScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
        incfluScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incfluScore()
```

4. Covid-19

An individual will be identified to have Fever as their likely diagnosis if first of all their body temperature is greater or equal to 37.8ºC. If the oxygen level turns out to be less than 92%, the score is increased further. The heart rate in this case is different for males and females. If the individual is a male and the heart rate is greater than 100 BPM, the probability score of being diagnosed with Covid-19 is increased. The diagnosis score for females increases if their heart rate turns out be greater than 82 BPM. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Covid-19 is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature > 37.8):
    inccovidScore()
if(oxygen < 92):
    inccovidScore()
if(sex == "Male"):
    if (heart_rate > 100):
        inccovidScore()
elif (sex == "Female"):
    if (heart_rate > 82):
        inccovidScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
        inccovidScore()
else:
    if (glucose >= 80 and glucose <= 120):
        inccovidScore()
```

5. Common Cold

An individual will be identified to have a Common Cold as their likely diagnosis if first of all their body temperature falls below 36.2ºC or the temperature is above 37.8ºC. If the oxygen level turns out to be less than 97%, the score is increased further. The heart rate in this case, again is different for males and females. If the individual is a male and the heart rate is greater than 100 BPM, the probability score of being diagnosed with Common Cold is increased. The diagnosis score for females increases if their heart rate turns out be greater than 82 BPM. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Common Cold is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature > 37.8 or temperature < 36.2):
    inccoldScore()
if(oxygen < 97):
    inccoldScore()
if(sex == "Male"):
    if (heart_rate > 100):
        inccoldScore()
elif (sex == "Female"):
    if (heart_rate > 82):
        inccoldScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
        inccoldScore()
else:
    if (glucose >= 80 and glucose <= 120):
        inccoldScore()
```

6. Asthma

An individual will be identified to have Asthma as their likely diagnosis if first of all their body temperature is greater than 37.8ºC. The oxygen level needs to be less than 90% to be identified as Asthma as the oxygen saturation is the most significant factor in identifying this particular problem. The heart rate in this case is different for males and females. If the individual is a male and the heart rate is greater than 120 BPM, the probability score of being diagnosed with Asthma is increased. The diagnosis score for females increases if their heart rate turns out be greater than 90 BPM. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Asthma is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature > 37.8):
    incasthmaScore()
if(oxygen < 90):
    incasthmaScore()
if(sex == "Male"):
    if (heart_rate > 120):
        incasthmaScore()
elif (sex == "Female"):
    if (heart_rate > 90):
        incasthmaScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
         incasthmaScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incasthmaScore()
```

7. Hyperglycemia

An individual will be identified to have Hyperglycemia as their likely diagnosis if first of all their body temperature is greater than 37.6ºC. The oxygen saturation does not really change if an individual suffers from hyperglycemia and thus the hyperglycemia probability score can be increased if the oxygen level is greater than or equal to 95%. If the individual has a heart rate greater than 88 BPM, the probability score is further increased. However, the glucose body vital is the factor that most significantly identifies whether a person is experiencing hyperglycemia. If the blood glucose level falls turns out to be greater than 120 mg / dl for a fasting user, the score for hyperglycemia increases. For an individual who is not fasting, the score for Hyperglycemia increases if the checkup value of blood glucose turns out to be greater than 180 mg / dl.

```
if(temperature > 37.6):
    inchyperglycemiaScore()
if(oxygen >= 95):
    inchyperglycemiaScore()
if (heart_rate > 88):
    inchyperglycemiaScore()
if (not_fasting == True):
    if (glucose > 180):
        inchyperglycemiaScore()
else:
    if (glucose > 120):
        inchyperglycemiaScore()
```

8. Pre-Diabetes

An individual will be identified to have Pre-Diabetes as their likely diagnosis if first of all their body temperature is greater than 37.6ºC. The oxygen saturation does not really change if an individual suffers from pre-diabetes and thus the probability for this particular diagnosis score can be increased if the oxygen level is greater than or equal to 95%. If the individual has a heart rate greater than 88 BPM, the probability score is further increased. However, the glucose body vital is the factor that most significantly identifies whether a person is experiencing hyperglycemia. If the blood glucose level falls within the range of 110 - 140 mg / dl for a fasting user, the score for pre-diabetes increases. For an individual who is not fasting, the score for this illness increases if the checkup value of blood glucose turns out to be anywhere from 150 to 180 mg / dl.

```
if(temperature > 37.6):
    incprediabetesScore()
if(oxygen >= 95):
    incprediabetesScore()
if (heart_rate > 88):
    incprediabetesScore()
if (not_fasting == True):
    if (glucose >= 150 and glucose <= 180):
        incprediabetesScore()
else:
    if (glucose >= 110 and glucose <= 140):
        incprediabetesScore()
```

9. Hypoglycemia

An individual will be identified to have Hypoglycemia as their likely diagnosis if first of all their body temperature is greater than 37.6ºC. If the oxygen level turns out to be less than or equal to 94%, the probability score of hypoglycemia is increased further. If the individual has a heart rate greater than 90 BPM, the probability score is further increased. However, the glucose body vital is the factor that most significantly identifies whether a person is experiencing hypoglycemia. If the blood glucose level falls turns out to be less than 75 mg / dl for a fasting user, the score for hyperglycemia increases. For an individual who is not fasting, the score for Hypoglycemia increases if the checkup value of blood glucose turns out to be greater than 145 mg / dl.

```
if(temperature > 37.6):
    inchypoglycemiaScore()
if(oxygen <= 94):
    inchypoglycemiaScore()
if (heart_rate > 90):
    inchypoglycemiaScore()
if (not_fasting == True):
    if (glucose <= 145):
        inchypoglycemiaScore()
else:
    if (glucose <= 75):
        inchypoglycemiaScore()
```

10. Pneumonia

An individual will be identified to have Pneumonia as their likely diagnosis if first of all their body temperature is greater than 38.4ºC. If the oxygen level turns out to be less than or equal to 92%, the probability score of pneumonia is increased further. The heart rate of the individual is spiked quite high and only if the value is greater than 112.5 BPM, the probability score is increased. The glucose body vital is also a factor that changes when a person is suffering from pneumonia. If the blood glucose level falls turns out to be greater than 95 mg / dl for a fasting user, the score for pneumonia increases. For an individual who is not fasting, the score for Pneumonia increases if the checkup value of blood glucose is greater than 130 mg / dl.

```
if(temperature > 38.4):
    incpneumoniaScore()
if(oxygen <= 92):
    incpneumoniaScore()
if (heart_rate > 112.5):
    incpneumoniaScore()
if (not_fasting == True):
    if (glucose >= 130):
        incpneumoniaScore()
else:
    if (glucose >= 95):
        incpneumoniaScore()
```

11. Stress

An individual identified to have Stress will have a pretty standard body temperature value, ranging from 36ºC to 38ºC. The oxygen level and glucose levels also typically match the ranges of a healthy person, with the oxygen level

being anywhere from 95 to 98% and the glucose level falling within 80 to 120 mg /dl. For a person who is not fasting, the glucose value once again can fall anywhere from 120 to 180 mg /dl. However, the main body vital that changes in the diagnosis of Stress is the heart rate, The heart rate of an individual needs to be greater than 125 BPM to add to the score of Stress.

```
if(temperature >= 36 and temperature <= 38):
    incstressScore()
if(oxygen >= 95 and oxygen <= 98):
    incstressScore()
if (heart_rate >= 125):
    incstressScore()
if(not_fasting == True):
    if(glucose >= 120 and glucose <= 180):
        incstressScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incstressScore()
```

12. Arrhythmia

An individual will be identified to have Arrhythmia as their likely diagnosis if first of all their body temperature drops less than 36.3ºC. If the oxygen level turns out to be less than or equal to 94%, the probability score is increased further. The heart rate in this case can either increase or decrease above the typical normal ranges, with the score of Arrhythmia increasing if the heart rate is greater than 90 BPM or in the opposite way, less than 60 BPM. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Arrhythmia is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature < 36.3):
    incarrhythmiaScore()
if(oxygen <= 94):
    incarrhythmiaScore()
if (heart_rate >= 90 or heart_rate <= 60):
    incarrhythmiaScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
        incarrhythmiaScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incarrhythmiaScore()
```

13. Coronary Heart Disease

An individual will have a greater probability of being diagnosed with Coronary Artery Disease if first of all, the body temperature is greater than or equal to 38ºC or lower than 36.3ºC. If the oxygen level turns out to be less than or equal to 93%, the score is increased further. The heart rate in this case is always lower so if the heart value falls anywhere between 60 to 75 BPM, the diagnosis of this disease is made more probable. The blood glucose vital does not really change in this situation so the glucose range for being diagnosed with Coronary Artery Disease is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if (bmi > 25) or (re.findall("Diabetes", current_problem)) or (re.findall("Diabetes", family_history)):
    inccoronaryScore()
if(temperature >= 38 and temperature < 36.3):
    inccoronaryScore()
if(oxygen <= 93):
    inccoronaryScore()
if (heart_rate >= 60 and heart_rate <= 75):
    inccoronaryScore()
if(not_fasting == True):
    if(glucose >= 120 and glucose <= 180):
        inccoronaryScore()
else:
    if (glucose >= 80 and glucose <= 120):
        inccoronaryScore()
```

14. Lung Cancer

An individual will have a greater probability of being diagnosed with Lung Cancer if first of all, the body temperature is greater than 38ºC. Since this disease affects the lungs, the oxygen level is critical in identifying if the diagnosis. If the oxygen level turns out to be less than 90%, which is extremely low, the score is increased further. The heart rate in this case can either increase or decrease above the typical normal ranges, with the score of Lung Cancer increasing if the heart rate is greater than 90 BPM or in the opposite way, less than 65 BPM. The blood glucose vital also changes considerably with the glucose value typically greater than 112.5 mg / dl if the person is fasting, and above 170 mg / dl if the person is checked when they are not fasting.

```
if(temperature > 38):
    inclungcancerScore()
if(oxygen < 90):
    inclungcancerScore()
if (heart_rate > 90 or heart_rate < 65):
    inclungcancerScore()
if (not_fasting == True):
    if (glucose > 170):
        inclungcancerScore()
else:
    if (glucose > 112.5):
        inclungcancerScore()
```

15. Breast Cancer

A female will have a greater probability of being diagnosed with Breast Cancer if first of all, the body temperature is greater than 38.2ºC. Since this disease affects an area around the lungs, the oxygen level is critical in identifying if the diagnosis. If the oxygen level turns out to be less than 90%, which is extremely low, the score is increased further. The heart rate in this case increases from the standard healthy range with the score of Breast Cancer increasing if the heart rate is greater than 90 BPM. The blood glucose vital also changes considerably with the glucose value typically greater than 112.5 mg / dl if the person is fasting, and above 160 mg / dl if the person is checked when they are not fasting.

```
if(sex == "Female"):
    if (temperature > 38.2):
        incbreastcancerScore()
    if (oxygen < 90):
        incbreastcancerScore()
    if (heart_rate > 90):
        incbreastcancerScore()
    if (not_fasting == True):
        if (glucose > 160):
            incbreastcancerScore()
    else:
        if (glucose > 112.5):
            incbreastcancerScore()
```

16. Leukemia

An individual will have a greater probability of being diagnosed with Leukemia if first of all, the body temperature is greater than 38ºC. This disease affects the bloodstream, so the oxygen level is the most critical vital in identifying the diagnosis. If the oxygen level turns out to be less than 90%, which is extremely low, the score is increased further. The score of Leukemia increases if the heart rate of the individual falls within 72 to 90 BPM. The blood glucose vital does not really change so the glucose range for being diagnosed with Leukemia is the same as a healthy individual – 80 to 120 mg / dl if the person is fasting, and 120 to 180 mg / dl if the person is not fasting.

```
if(temperature > 38):
    incleukemiaScore()
if(oxygen < 90):
    incleukemiaScore()
if (heart_rate >= 72 or heart_rate <= 90):
    incleukemiaScore()
if (not_fasting == True):
    if (glucose >= 120 and glucose <= 180):
        incleukemiaScore()
else:
    if (glucose >= 80 and glucose <= 120):
        incleukemiaScore()
```

4.3.2 Summarized Diagnosis-Body Vital Decision Table

S.No	Diagnosis	Body Vital	Vital Range		Other Factors
1.	Healthy	Heart Rate	60 <= HR <= 100		-
		Body Temperature	36.4 <= Body Temp. <= 37.5		
		Blood Glucose Level	Fasting	Not Fasting	
			Glucose >= 80	Glucose >= 120	

			Glucose <= 120	Glucose <= 180	
		Blood Oxygen Level	Oxygen >= 95		
2.	**Fever**	**Heart Rate**	HR >= 90		-
		Body Temperature	Body Temp. >= 37.5		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 80 Glucose <= 120	Glucose >= 120 Glucose <= 180	
		Blood Oxygen Level	Oxygen <= 96		
3.	**Flu**	**Heart Rate**	**Male**	**Female**	-
			HR > 100	HR > 82	
		Body Temperature	Body Temp. > 37.8 OR Body Temp. < 36.2		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 80 Glucose <= 120	Glucose >= 120 Glucose <= 180	
		Blood Oxygen Level	Oxygen < 95		

<table>
<tr><td rowspan="6">4.</td><td rowspan="6">Covid-19</td><td rowspan="2">Heart Rate</td><td>Male</td><td>Female</td><td rowspan="6">Age > 45</td></tr>
<tr><td>HR > 100</td><td>HR > 82</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 37.8</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose >= 80
Glucose <= 120</td><td>Glucose >= 120
Glucose <= 180</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 92</td></tr>
<tr><td rowspan="6">5.</td><td rowspan="6">Common Cold</td><td rowspan="2">Heart Rate</td><td>Male</td><td>Female</td><td rowspan="6">-</td></tr>
<tr><td>HR > 100</td><td>HR > 82</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 37.8
OR
Body Temp. < 36.2</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose >= 80
Glucose <= 120</td><td>Glucose >= 120
Glucose <= 180</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 97</td></tr>
<tr><td rowspan="2">6.</td><td rowspan="2">Asthma</td><td rowspan="2">Heart Rate</td><td>Male</td><td>Female</td><td rowspan="2">-</td></tr>
<tr><td>HR > 110</td><td>HR > 90</td></tr>
</table>

<table>
<tr><td rowspan="5"></td><td rowspan="5"></td><td>Body Temperature</td><td colspan="2">Body Temp. > 37.8</td><td rowspan="5"></td></tr>
<tr><td rowspan="3">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose >= 80
Glucose <= 120</td><td>Glucose >= 120
Glucose <= 180</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 90</td></tr>
<tr><td rowspan="5">7.</td><td rowspan="5">Hyperglycemia</td><td>Heart Rate</td><td colspan="2">HR > 88</td><td rowspan="5">• Age > 45</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 37.6</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose > 120</td><td>Glucose > 180</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen >= 95</td></tr>
<tr><td rowspan="5">8.</td><td rowspan="5">Pre-Diabetes</td><td>Heart Rate</td><td colspan="2">HR > 88</td><td rowspan="5">• Age > 30</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 37.6</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose >= 110
Glucose <= 140</td><td>Glucose >= 150
Glucose <= 180</td></tr>
</table>

		Blood Oxygen Level	Oxygen >= 95		
9.	**Hypoglycemia**	**Heart Rate**	HR > 90		-
		Body Temperature	Body Temp. > 37.6		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose <= 75	Glucose >= 145	
		Blood Oxygen Level	Oxygen <= 94		
10.	**Pneumonia**	**Heart Rate**	HR > 112		-
		Body Temperature	Body Temp. > 38.4		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 95	Glucose >= 130	
		Blood Oxygen Level	Oxygen <= 92		
11.	**Stress**	**Heart Rate**	HR >= 125		-
		Body Temperature	36 <= Body Temp. <= 38		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 80	Glucose >= 120	

			Glucose <= 120	Glucose <= 180	
		Blood Oxygen Level	95 <= Oxygen <= 98		
12.	**Arrhythmia**	**Heart Rate**	HR >= 90 OR HR <= 60		• BMI > 25
		Body Temperature	Body Temp. < 36.3		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 80 Glucose <= 120	Glucose >= 120 Glucose <= 180	
		Blood Oxygen Level	Oxygen <= 94		
13.	**Coronary Artery Disease**	**Heart Rate**	60 <= HR <= 72		• BMI > 25 • History of Diabetes • Currently having Diabetes
		Body Temperature	Body Temp. >= 38 OR Body Temp. < 36.3		
		Blood Glucose Level	**Fasting**	**Not Fasting**	
			Glucose >= 80 Glucose <= 120	Glucose >= 120 Glucose <= 180	
		Blood Oxygen Level	Oxygen <= 93		
14.	**Lung Cancer**	**Heart Rate**	HR > 90 OR HR < 65		-

<table>
<tr><td rowspan="5"></td><td rowspan="5"></td><td>Body Temperature</td><td colspan="2">Body Temp. > 38</td><td rowspan="5"></td></tr>
<tr><td rowspan="3">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose > 112.5</td><td>Glucose > 170</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 90</td></tr>
<tr><td rowspan="5">15.</td><td rowspan="5">Breast Cancer</td><td>Heart Rate</td><td colspan="2">HR > 90</td><td rowspan="5">• Sex = Female
• Family History</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 38.2</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose > 112.5</td><td>Glucose > 160</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 90</td></tr>
<tr><td rowspan="5">16.</td><td rowspan="5">Leukemia</td><td>Heart Rate</td><td colspan="2">72 <= Heart Rate <= 90</td><td rowspan="5">• Family History</td></tr>
<tr><td>Body Temperature</td><td colspan="2">Body Temp. > 38</td></tr>
<tr><td rowspan="2">Blood Glucose Level</td><td>Fasting</td><td>Not Fasting</td></tr>
<tr><td>Glucose >= 80
Glucose <= 120</td><td>Glucose >= 80
Glucose <= 120</td></tr>
<tr><td>Blood Oxygen Level</td><td colspan="2">Oxygen < 90</td></tr>
</table>

4.3.3 Reference Symptoms

After the monitored values are received, the symptoms of each diagnosis need to be checked in order to completely guarantee and satisfy the diagnosis. A file called Symptoms.txt is referenced to check and evaluate each symptom and add to the diagnosis score in the medibase list.

```
Healthy:
Fever:headache,feeling tired
Flu:vomiting,sweating or shivering,coughing,chest pain,headache
Covid-19:sore throat,coughing,breathing difficulty,loss of appetite,feeling tired,headache
Common Cold:runny nose,sneezing,coughing,tired
Asthma:long term cough,breathing difficulty,suffocated,chest feels tight
Hyperglycemia:extreme thirst,increased appetite,vision is blurry,dry mouth,feeling tired
Pre Diabetes:extreme thirst,increased appetite,vision is blurry,dry mouth,feeling tired
Hypoglycemia:sweating or shivering,nauseous,light headed,feeling tired
Pneumonia:chest pain,cough,breathing difficulty,sweating or shivering
Stress:
Arrhythmia:chest pain,breathing difficulty,dizziness
Coronary Artery Disease:chest pain,breathing difficulty,swelling of arms and feet,feeling tired
Lung Cancer:long term cough,coughing blood,body swelling,chest pain
Breast Cancer:swelling of your breast,irritated around the breast,pain around the breast
Leukemia:nosebleeds,having bleeding gums,bone and joint pain,feeling tired
```

- Loop to process the **symptomfile**:
    - This part of the code seems to be iterating over lines in a file named **symptomfile**.
    - Each line is assigned to the variable **line**.
    - The **symp** list is initialized as an empty list to store symptoms associated with a medical condition.
- Line count and symptom counting:
    - The variable **linecount** is used to keep track of the line number being processed.
    - The loop increments **linecount** by 1 for each iteration, representing the line number.
- Updating **medibase** with symptom information:
    - The code updates the sub list for each medical condition in the **medibase** list with the following information:
        - The line count (i.e., the line number from the **symptomfile** associated with that medical condition).
        - The number of symptoms found for that medical condition.

- Matching medical conditions:
    - The code uses **re.findall()** to check if the name of each medical condition in **medibase** matches any content in the **line**.
    - If there is a match, it extracts the relevant symptom information from the line and appends it to the **symp** list.
- Counting symptoms:
    - The loop then counts the number of symptoms present in the **symp** list and stores that count in the **symp_count** variable.
- Updating the **medibase** list with symptom count:
    - After counting the symptoms, the **symp_count** is stored in the **medibase** sub list for the matching medical condition.

4.3.4 Summarized Diagnosis-Symptoms Table

S.No	Diagnosis	List of Symptoms
1.	**Healthy**	-
2.	**Fever**	i. Headache ii. Feeling Tired iii. Sweating or Shivering
3.	**Flu**	i. Vomiting ii. Sweating or Shivering iii. Coughing iv. Chest Pain v. Headache
4.	**Covid-19**	i. Sore Throat

		ii. Coughing iii. Breathing Difficulty iv. Loss of appetite v. Feeling Tired vi. Headache
5.	**Common Cold**	i. Runny Nose ii. Sneezing iii. Coughing iv. Tired
6.	**Asthma**	i. Long Term Cough ii. Breathing Difficulty iii. Suffocated iv. Chest Feels Tight
7.	**Hyperglycemia**	i. Extreme Thirst ii. Increased Appetite iii. Vision is Blurry iv. Dry Mouth v. Feeling Tired
8.	**Pre-Diabetes**	i. Extreme Thirst ii. Increased Appetite iii. Vision is Blurry iv. Dry Mouth v. Feeling Tired

9.	**Hypoglycemia**	i. Sweating or Shivering ii. Nauseous iii. Lightheaded iv. Feeling Tired
10.	**Pneumonia**	i. Chest Pain ii. Cough iii. Breathing Difficulty iv. Sweating or Shivering
11.	**Stress**	-
12.	**Arrhythmia**	i. Chest Pain ii. Breathing Difficulty iii. Dizziness
13.	**Coronary Artery Disease**	i. Chest Pain ii. Breathing Difficulty iii. Swelling of Arms and Feet iv. Feeling Tired
14.	**Lung Cancer**	i. Long Term Cough ii. Coughing Blood iii. Body Swelling iv. Chest Pain
15.	**Breast Cancer**	i. Swelling of your Breast ii. Irritated around the Breast iii. Pain around the Breast

16.	**Leukemia**	i. Nosebleeds ii. Bleeding Gums iii. Bone and Joint Pain iv. Feeling tired

4.3.5 Recommendation

The code then checks the user's health parameters and updates the scores accordingly for each medical condition based on the user's input. The code identifies the highest score among the medical conditions and selects the diagnosis with the highest score. If there are multiple diagnoses with the same high score, it stores them in the prediction list.

The code first checks whether the temporary diagnosis has symptoms alongside the name in the Symptoms file via user inputs.

If the user responds with "No" or indicates that they haven't been experiencing any symptoms, the code increments the symptom count for each medical condition that has only one associated symptom. It calculates the symptom score as:

(number of symptoms matched / total number of symptoms for the condition) * 50`

After updating the symptom scores for single-symptom conditions, the code identifies the medical condition with the highest score and assigns the diagnosis and solution for the identified condition:

If the user responds with "Yes" or indicates that they have been experiencing symptoms, the code proceeds with further symptom assessment. The user is asked to describe the symptoms:

The code then goes through each line of the and compares the user's symptoms with the symptoms listed for each medical condition in the found symptom line. If a symptom is matched, it increments the symptom count for the corresponding medical condition.

```
if (sympline[0][0] == ""):
    symptomQ = "Have you been experiencing any kind of symptoms?"
    database.child("Model").child("001").child("Symptom Question").set(symptomQ)

    symptomSet = False
    while (symptomSet == False):
        # sleep(3)
        symptomAns = database.child("Model").child("001").child("Symptom Answer").get()
        symptomA = symptomAns.val()
        if (symptomA != ""):
            symptom_answer = symptomA
            symptomSet = True

    print(symptom_answer)
    database.child("Model").child("001").child("Symptom Question").set("")
    database.child("Model").child("001").child("Symptom Answer").set("")

    if ((re.findall("No", symptom_answer)) or (re.findall("not been", symptom_answer))):
        for i in range(len(medibase)):
            if(medibase[i][4] == 1):
                medibase[i][5] += 1
                symptom_score = (medibase[i][5] / medibase[i][4]) * 50
                medibase[i][1] += symptom_score


else:
    index = 0
    not_symptom = 0
    while (index + 1 <= len(sympline[0])):
        symptomQ = "Have you been been suffering from " + sympline[0][index]
        database.child("Model").child("001").child("Symptom Question").set(symptomQ)

        symptomSet = False
        while (symptomSet == False):
            # sleep(3)
            symptomAns = database.child("Model").child("001").child("Symptom Answer").get()
            symptomA = symptomAns.val()
            if (symptomA != ""):
                symptom_answer = symptomA
                symptomSet = True
```

However, if the user indicates that they haven't been experiencing the specific symptom (by answering "No" to the symptom question), the `not_symptom` counter is incremented. If this happens for fewer than three symptoms, the code prints "Alright then" and continues assessing the remaining symptoms. But if the `not_symptom` counter reaches three, it means the user is unable to provide symptom information, and they are prompted to describe their symptoms:

```
if ((re.findall("Yes", symptom_answer)) or (re.findall("have been", symptom_answer)) or (re.findall("yes", symptom_answer))):
    filecount = 0
    for line in symptomfile:
        filecount += 1
        symptoms.append(re.split(",", line))
        for i in range(len(symptoms)):
            for j in range(len(symptoms[i])):
                if (re.findall(sympline[0][index], symptoms[i][j])):
                    medibase[filecount - 1][5] += 1
    for i in range(len(medibase)):
        symptom_score = (medibase[i][5] / medibase[i][4]) * 50
        medibase[i][1] += symptom_score
```

```
elif((re.findall("No", symptom_answer)) or (re.findall("not been", symptom_answer)) or (re.findall("no", symptom_answer))):
    not_symptom += 1
    print(not_symptom)
    if (not_symptom < 3):
        symptomQ = "Hmm, Alright Then"
        database.child("Model").child("001").child("Symptom Question").set(symptomQ)
        sleep(5)
    else:
        sympline.clear()
        filecount = 0
        symptomQ = "Can you then please describe the symptoms you are having?"
        database.child("Model").child("001").child("Symptom Question").set(symptomQ)

        symptomSet = False
        while (symptomSet == False):
            # sleep(3)
            symptomAns = database.child("Model").child("001").child("Symptom Answer").get()
            symptomA = symptomAns.val()
            if (symptomA != ""):
                symptom_answer = symptomA
                symptomSet = True
```

If there are multiple high-scoring diagnoses, it asks the user to provide additional symptoms to narrow down the diagnosis. It then updates the scores based on the new symptoms. Finally, the code selects the medical condition with the highest score as the identified diagnosis and presents the corresponding solution for the condition.

```
samefile = open("/home/user/Healthcare/Symptoms", "r")
for line in samefile:
    filecount += 1
    for i in range(len(medibase)):
        if (medibase[i][3] == filecount):
            cleaned_line = line.replace(medibase[i][0] + ":", "")
            cleaned_line = cleaned_line.replace("\n", "")
            sympline.append(re.split(",", cleaned_line))
for i in range(len(sympline)):
    for j in range(len(sympline[i])):
        if (re.findall(sympline[i][j], symptom_answer) and (sympline[i][j] != "")):
            medibase[i][5] += 1
for i in range(len(medibase)):
    symptom_score = (medibase[i][5] / medibase[i][4]) * 50
    medibase[i][1] += symptom_score
for i in range(len(medibase)):
    if (medibase[i][1] > final_high):
        final_high = medibase[i][1]
        diagnosis = medibase[i][0]
        solution = medibase[i][2]
database.child("Users").child(user).child("Last Diagnosis").set(diagnosis)
database.child("Users").child(user).child("Last Recommendation").set(solution)
database.child("Model").child("001").child("Complete Prediction").set(True)
```

Finally, the code selects the medical condition with the highest score as the identified diagnosis and presents the corresponding solution for the condition and displays it for the user and save the diagnosis with the recommendation for the user as part of the cloud storage.

4.3.6 Summarized Diagnosis - Recommendation Table

S.No	Diagnosis	Recommendation

1.	**Healthy**	Continue having a fun life :)
2.	**Fever**	Consider Aspirin or Ibuprofen Tablets for treatment.
3.	**Flu**	Consider taking Oseltamivir tablets or Flu Shots from a doctor.
4.	**Covid-19**	Isolate yourself, Drink hot water and inform medical facility.
5.	**Common Cold**	Oxymetazoline Nasal Spray with Benzonatate tablets.
6.	**Asthma**	Get an Aerosol or Ventolin Inhaler with Prednisolone tablets.
7.	**Hyperglycemia**	You require Prescribed Insulin Shots and Glucotrol tablets.
8.	**Pre-Diabetes**	Consider taking Metformin Tablets.
9.	**Hypoglycemia**	You require Timely Glucagon Shots and Metformin tablets.
10.	**Pneumonia**	You require Penicillin G tablets or Augmentin tablets.
11.	**Stress**	Consider taking Aspirin to reduce pain.
12.	**Arrhythmia**	Propafenone and Acebutolol Beta Blocker tablets
13.	**Coronary Artery Disease**	Niacin supplements, Ranolazine and Acebutolol Beta Blocker tablets

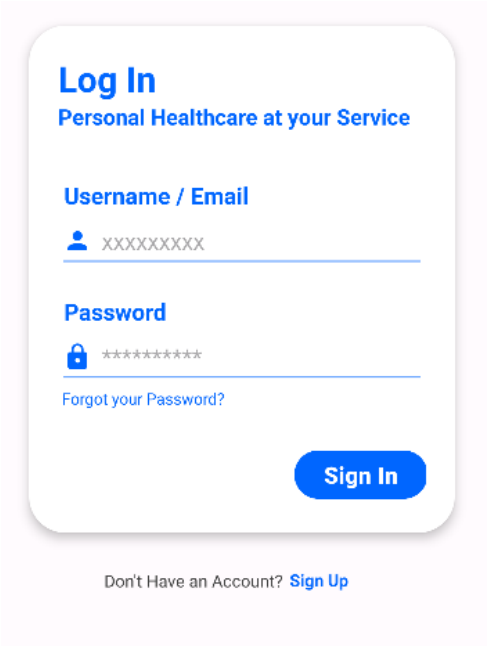
14.	**Lung Cancer**	Sadly, you might require chemotherapy. Cisplatin and Carboplatin can help but you need a doctor.
15.	**Breast Cancer**	Sadly, Chemotherapy is needed. Adriamycin, Cytoxan and Taxotere will help from the doctor.
16.	**Leukemia**	Sadly, Chemotherapy or Targeted Therapy is required. Doctors might also use immunotherapy.

4.4 Application Development

The overall function of this code is used to set up a connection to the Firebase Realtime Database using the Firebase Android SDK in Java. The code establishes a DatabaseReference object that can be used to interact with the database and perform various database operations, such as reading and writing data.

```
57 usages
DatabaseReference databaseReference = FirebaseDatabase.getInstance().getReferenceFromUrl("https://healthcare-c3b0a-default-rtdb.firebaseio
```

4.4.1 Login Activity



The Login Activity handles the login process. The Sign In button click is handled to perform the login process. The process on this page mainly checks if a registered user has entered both their username and password. If any of the fields are left empty, an error message is displayed to fill in all the fields. If both the fields are provided, it queries the Firebase Realtime Database to check if the entered username exists as the key in the "Users" node.

If the email/username exists, it retrieves all the user's information from the database, including their name, date of birth, email, sex, height, weight, current health issues, current medication, model, last heart rate, last oxygen level, last temperature, last glucose level, last diagnosis, and last recommendation.

The application then compares the entered password with the password retrieved from the database and only if the passwords match, the system launches the Homepage activity and passes the user's data as intent extras. If the password is incorrect, the login page shows a toast message to warn the user. If the entered email/username is not found in the database, the application activity shows a toast message to inform the user.

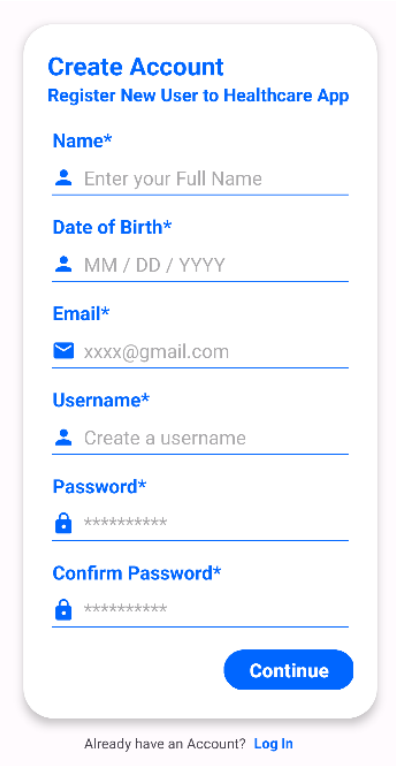
```
databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.hasChild(loginId))
        {
            final String getPassword = snapshot.child(loginId).child( path: "Password").getValue(String.class);
            final String getName = snapshot.child(loginId).child( path: "Name").getValue(String.class);
            final String getDOB = snapshot.child(loginId).child( path: "DOB").getValue(String.class);
            final String getEmail = snapshot.child(loginId).child( path: "Email").getValue(String.class);
            final String getSex = snapshot.child(loginId).child( path: "Sex").getValue(String.class);
            final String getHeight = String.valueOf(snapshot.child(loginId).child( path: "Height").getValue(Double.class));
            final String getWeight = String.valueOf(snapshot.child(loginId).child( path: "Weight").getValue(Double.class));
            final String getCurrentIssues = snapshot.child(loginId).child( path: "Current Issues").getValue(String.class);
            final String getCurrentMedication = snapshot.child(loginId).child( path: "Current Medication").getValue(String.class);
            final String getModel = snapshot.child(loginId).child( path: "Model").getValue(String.class);
            final String getLastHeart = String.valueOf(snapshot.child(loginId).child( path: "Last Heart").getValue(Integer.class));
            final String getLastO2 = String.valueOf(snapshot.child(loginId).child( path: "Last O2").getValue(Integer.class));
            final String getLastTemp = String.valueOf(snapshot.child(loginId).child( path: "Last Temp").getValue(Double.class));
            final String getLastGlucose = String.valueOf(snapshot.child(loginId).child( path: "Last Glucose").getValue(Double.class));
            final String getLastDiagnosis = snapshot.child(loginId).child( path: "Last Diagnosis").getValue(String.class);
            final String getLastRecommendation = snapshot.child(loginId).child( path: "Last Recommendation").getValue(String.class);
```

```
if (getPassword.equals(loginPassword))
{

    Intent intent = new Intent( packageContext: LoginActivity.this, Homepage.class);
    intent.putExtra( name: "user", loginId);
    intent.putExtra( name: "name", getName);
    intent.putExtra( name: "dob", getDOB);
    intent.putExtra( name: "email", getEmail);
    intent.putExtra( name: "pwd", getPassword);
    intent.putExtra( name: "sex", getSex);
    intent.putExtra( name: "height", getHeight);
    intent.putExtra( name: "weight", getWeight);
    intent.putExtra( name: "issues", getCurrentIssues);
    intent.putExtra( name: "medication", getCurrentMedication);
    intent.putExtra( name: "model", getModel);
    intent.putExtra( name: "last_heart", getLastHeart);
    intent.putExtra( name: "last_o2", getLastO2);
    intent.putExtra( name: "last_temp", getLastTemp);
    intent.putExtra( name: "last_glucose", getLastGlucose);
    intent.putExtra( name: "last_diagnosis", getLastDiagnosis);
    intent.putExtra( name: "last_recommendation", getLastRecommendation);
    startActivity(intent);
    finish();
}
else
{
    Toast.makeText( context: LoginActivity.this,  text: "Incorrect Password Entered.", Toast.LENGTH_SHORT).show();
}
```

Individuals who do not have a registered account already, can click on the Sign-Up Link to navigate to the registration screen.

```
signUpLink.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent( packageContext: LoginActivity.this, RegisterActivity1.class));
    }
});
```

4.4.2 Register User Profile



A new application user who does not have an account needs to be registered. An unregistered application user is required to fill in their name, date of birth, email, desired username, password, and confirm password to start the registration process. Once the Continue button click is clicked, the registration process begins.

The application checks if any of the required fields are empty, and if so, it displays a toast message to ask the user to fill in all fields. There is also a verification process to ensure that the entered password matches the confirmed password.

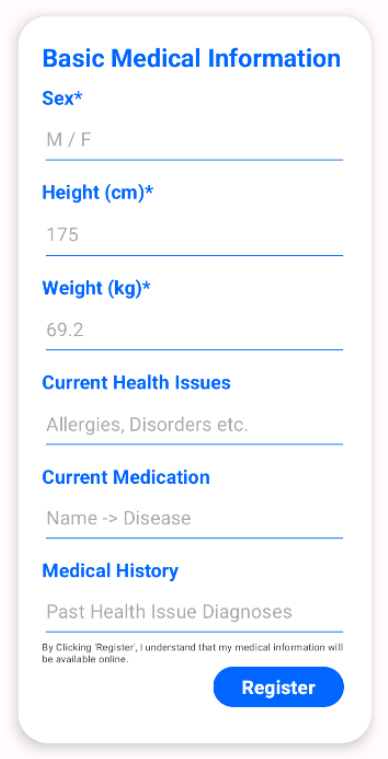
If all required fields are filled, the code checks if the desired username is already taken by querying the Firebase Realtime Database under the "Users" node. If the username is available, it proceeds to create a new user entry in the Firebase database under the "Users" node with the provided name, date of birth, email, and password. Once the Firebase database is updated, the app follows by launching the second Register Activity and passes the username as an intent extra to proceed with the next step of registration.

The Login Link can be clicked back to navigate back to the login screen if the user already has an account and wants to log in.

```
databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.hasChild(registerUsername))
        {
            Toast.makeText( context: RegisterActivity1.this,  text: "Username has already been registered", Toast.LENGTH_SHORT).show();
        }
        else
        {
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Name").setValue(registerName);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "DOB").setValue(registerDOB);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Email").setValue(registerEmail);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Password").setValue(registerPassword);

            Intent intent = new Intent( packageContext: RegisterActivity1.this, RegisterActivity2.class);
            intent.putExtra( name: "keyUsername", registerUsername);
            startActivity(intent);
            finish();
        }
    }
```

```
loginLink.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        startActivity(new Intent( packageContext: RegisterActivity1.this, LoginActivity.class));
    }
});
```

4.4.3 Register Medical Profile



The code in this specific activity handles the second step of user registration - updating the medical information of the user in the Firebase Realtime Database. It helps create a comprehensive user profile for healthcare-related services. After retrieving the username, from the previous account registration, the user is required to fill in their sex, measured height, measured weight, current issues, current medication, and disease history. If all the required fields (sex, height, weight) are filled, The Register button can be clicked to complete the registration process.

If all required fields are filled, the code updates the user's medical information in the Firebase Realtime Database under the "Users" node with the provided sex, height, weight, and other details.

```
databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.hasChild(username))
        {
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Sex").setValue(registerSex);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Height").setValue(registerHeight);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Weight").setValue(registerWeight);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Model").setValue("001");
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last Temp").setValue(-1);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last Heart").setValue(-1);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last O2").setValue(-1);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last Glucose").setValue(-1);
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last Diagnosis").setValue("");
            databaseReference.child( pathString: "Users").child(username).child( pathString: "Last Recommendation").setValue("");

            if (!registerIssues.isEmpty())
            {
                databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Issues").setValue(registerIssues);
            }
            else
            {
                databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Issues").setValue("");
            }
```

- Fields like "Model," "Last Temp," "Last Heart," "Last O2," "Last Glucose," "Last Diagnosis," and "Last Recommendation" are initialized with default or empty values.

- If the user provided additional information in fields like "Current Issues," "Current Medication," and "Medical History," the code updates these fields in the database accordingly.

Once the registration process is complete, the user is informed with a toast message and then redirected to the login screen to sign in with the newly registered account.

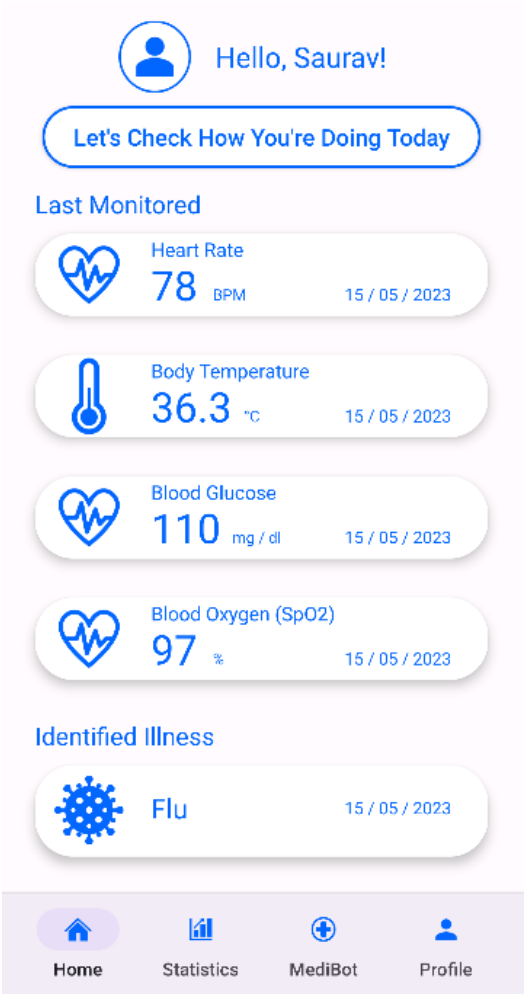
```
    if (!registerMedication.isEmpty())
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Medication").setValue(registerMedicat
    }
    else
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Medication").setValue("");
    }

    if (!registerHistory.isEmpty())
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Medical History").setValue(registerHistory);
    }
    else
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Medical History").setValue("");
    }

    Toast.makeText( context: RegisterActivity2.this,  text: "User Successfully Registered!", Toast.LENGTH_SHORT).show();
    startActivity(new Intent( packageContext: RegisterActivity2.this, LoginActivity.class));
    finish();

}
else
{
    Toast.makeText( context: RegisterActivity2.this,  text: "Could not Update Medical Information.", Toast.LENGTH_SHORT).show();
}
```

4.4.4 Homepage

The homepage of the application is basically a navigation page for the user that allows users to move to different fragments of the application. The homepage is coded with a navigation bar that allows users to move to view health statistics, go to the MediBot Activity or view and update profiles.

```
<?xml version="1.0" encoding="utf-8"?>
<menu xmlns:android="http://schemas.android.com/apk/res/android">

    <item
        android:id="@+id/home"
        android:title="Home"
        android:icon="@drawable/home" />

    <item
        android:id="@+id/stats"
        android:title="Statistics"
        android:icon="@drawable/stats" />

    <item
        android:id="@+id/bot"
        android:title="MediBot"
        android:icon="@drawable/medi" />

    <item
        android:id="@+id/profile"
        android:title="Profile"
        android:icon="@drawable/baseline_person_24" />

</menu>
```

The code in the Homepage presents the home screen of the healthcare app, displaying the user's personal information and last recorded health data, and provides options to access the chatbot and view health statistics. The data is fetched from the Firebase Realtime Database, and the user can interact with the app to access different features. The page shows the user's last recorded heart rate, oxygen level, temperature, glucose level, and diagnosis fetched from the real-time Database.



```
databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        if (snapshot.hasChild(user))
        {
            final String lastHeartVal = String.valueOf(snapshot.child(user).child( path: "Last Heart").getValue(Integer.class));
            lastHeart.setText(lastHeartVal);

            final String lastO2Val = String.valueOf(snapshot.child(user).child( path: "Last O2").getValue(Integer.class));
            lastO2.setText(lastO2Val);

            final String lastTempVal = String.valueOf(snapshot.child(user).child( path: "Last Temp").getValue(Double.class));
            lastTemp.setText(lastTempVal);

            final String lastGlucoseVal = String.valueOf(snapshot.child(user).child( path: "Last Glucose").getValue(Double.class));
            lastGlucose.setText(lastGlucoseVal);

            final String diagnosis = snapshot.child(user).child( path: "Last Diagnosis").getValue(String.class);
            lastIllness.setText(diagnosis);
        }
    }
```

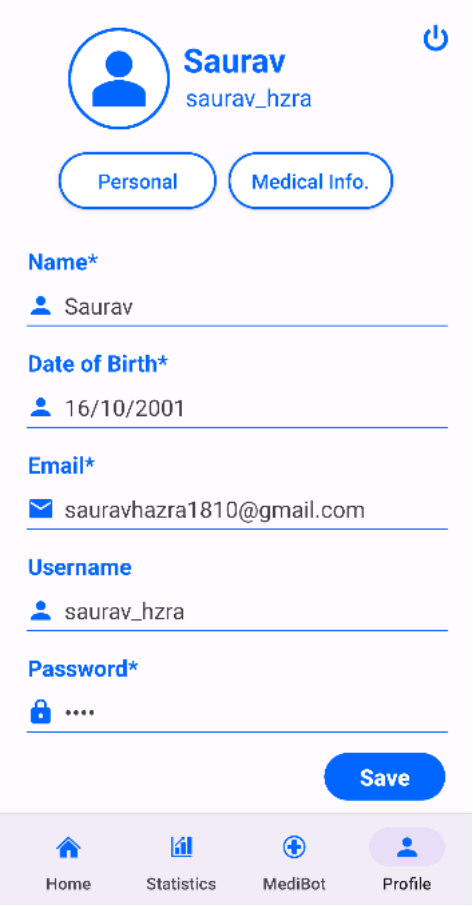
The homepage shows the user's last recorded heart rate, oxygen level, temperature, glucose level, and diagnosis fetched from the Firebase Realtime Database. If the last recorded value is not available, it displays a default placeholder text. The last monitored values can also be accessed as buttons to view the corresponding body vital statistic page.

```
heartStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new StatsFragment()); }
});

o2StatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new O2StatsFragment()); }
});

tempStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new TempStatsFragment()); }
});

glucoseStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new GlucoseStatsFragment()); }
});
```

4.4.5 User Profile Page

The User Profile page allows users to update their personal account information through the app. The page gathers all the information from the Firebase Realtime Database once the user logs in and carries this all this information across all the application pages. On the User Profile Page all the account and

personal information is displayed and can be edited except for the username. All the fields are filed in with the appropriate information and can be changed.



If the "Update" button is pressed, the application allows the user to modify their personal information. When the button is clicked:

- It checks if all the required fields are filled. If not, it displays a toast message to ask the user to fill in all required fields.
- If all required fields are filled, it updates the user's personal information in the Firebase Realtime Database under the "Users" node with the provided name, birthdate, email, and password.
- After successful update, it displays a toast message confirming the update and asks the user to log in and out to present the updated information.

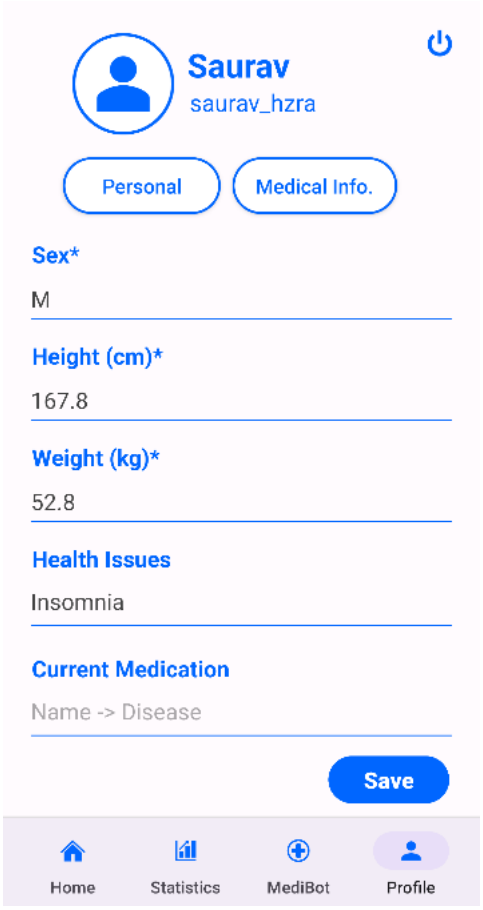
```
final String registerName = nameField.getText().toString();
final String registerDOB = dobField.getText().toString();
final String registerEmail = emailField.getText().toString();
final String registerUsername = usernameField.getText().toString();
final String registerPassword = passwordField.getText().toString();

if (registerName.isEmpty() || registerDOB.isEmpty() || registerEmail.isEmpty() || registerUsername.isEmpty() || registerPassword.isEmpty())
{
    Toast.makeText(getActivity(), text: "Please fill in all the fields.", Toast.LENGTH_SHORT).show();
}
else
{
    databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {

            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Name").setValue(registerName);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "DOB").setValue(registerDOB);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Email").setValue(registerEmail);
            databaseReference.child( pathString: "Users").child(registerUsername).child( pathString: "Password").setValue(registerPassword);

            Toast.makeText(getActivity(), text: "Personal Information Successfully Updated. Information will be refreshed once you Log Back I
            home.replaceFragment(new ProfileFragment());
        }
```

The user can click the "Logout" button to log out and navigate to the login screen at any time. The user is also presented with “Personal” and "Medical" buttons to switch between personal and medical profile sections.

```
logout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(), LoginActivity.class);
        startActivity(intent);
    }
});

personal.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new ProfileFragment()); }
});

medical.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new MediProfileFragment()); }
});
```

4.4.6 Medical Profile Page

The Medical Profile page allows users to update their medical information to keep their checkups as accurate and informed as possible. The page gathers all the required medical factors from the Firebase Realtime Database once the user logs in and carries this all this information across to this application page. On the Medical Profile Page, the medical information is displayed and can also be edited through the text fields. All the fields are filled in with the appropriate information and can be changed.



If the "Update" button is pressed, the application allows the user to modify their medical information. When the button is clicked:

- It checks if all the required fields (Sex, Height and Weight) are filled. If not, it displays a toast message to ask the user to fill in all required fields.
- If all required fields are filled, it updates the user's medicall information in the Firebase Realtime Database under the "Users" node with the provided name, birthdate, email, and password.

After a successful update, it displays a toast message confirming the update and asks the user to log in and out to present the updated information.

```
if (snapshot.hasChild(username))
{
    databaseReference.child( pathString: "Users").child(username).child( pathString: "Sex").setValue(registerSex);
    databaseReference.child( pathString: "Users").child(username).child( pathString: "Height").setValue(registerHeight);
    databaseReference.child( pathString: "Users").child(username).child( pathString: "Weight").setValue(registerWeight);
    if (!registerIssues.isEmpty())
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Issues").setValue(registerIssues);
    }
    else
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Issues").setValue("");
    }

    if (!registerMedication.isEmpty())
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Medication").setValue(registerMedication);
    }
    else
    {
        databaseReference.child( pathString: "Users").child(username).child( pathString: "Current Medication").setValue("");
    }

    Toast.makeText(getActivity(), text: "Medical Information Successfully Updated. Information will be refreshed once you Log Back In.",
    home.replaceFragment(new MediProfileFragment());

}
else
{
    Toast.makeText(getActivity(), text: "Could not Update Medical Information.", Toast.LENGTH_SHORT).show();
}
```

The user can click the "Logout" button to log out and navigate to the login screen at any time. The user is also presented with “Personal” and "Medical" buttons to switch between personal and medical profile sections.

```
logout.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        Intent intent = new Intent(getActivity(), LoginActivity.class);
        startActivity(intent);
    }
});

personal.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new ProfileFragment()); }
});

medical.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new MediProfileFragment()); }
});
```

4.4.7 Statistics Page

All the Stats Pages – Heart Rate, Body Temperature, Blood Glucose, O2 Statistics Pages efficiently handles fetching the individual body vital data for the last seven days from the Firebase Realtime Database and dynamically updates

the line chart and statistical information on the UI. This allows users to visualize and track their heart rate trends over time.

The **createGraph** method sets up the line chart's appearance, including its description, X-axis, and Y-axis formatting. It also prepares the data entries for the line chart. The line chart data entries are represented by a List of Entry objects (x, y values). A **LineDataSet** is created using these entries to customize the appearance of the line on the chart. All the values must be converted to Float datatypes.

```
Description description = new Description();
description.setText("Heart Rate");
description.setPosition( x: 150f,  y: 15f);

lineChart.setDescription(description);
lineChart.getAxisRight().setDrawLabels(false);

xValues = Arrays.asList(formattedSevenDate, formattedSixDate, formattedFiveDate, formattedFourDate, formattedThreeDate, formattedTwoDate,

XAxis xAxis = lineChart.getXAxis();
xAxis.setPosition(XAxis.XAxisPosition.BOTTOM);
xAxis.setValueFormatter(new IndexAxisValueFormatter(xValues));
xAxis.setLabelCount(7);
xAxis.setGranularity(1f);

YAxis yAxis = lineChart.getAxisLeft();
yAxis.setAxisMaximum(120f);
yAxis.setAxisMinimum(50f);
yAxis.setAxisLineWidth(2f);
yAxis.setAxisLineColor(Color.BLACK);
yAxis.setLabelCount(7);
```

The code uses the **LocalDateTime and DateTimeFormatter** classes to obtain the current date and format it into a specific pattern (dd-MM-yyyy). It also calculates dates for the last seven days and formats them similarly. These dates will likely be used for data retrieval and display purposes.

```
if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
    recordedDate = LocalDateTime.now();
    sevenDate = LocalDateTime.now().minusDays(6);
    sixDate = LocalDateTime.now().minusDays(5);
    fiveDate = LocalDateTime.now().minusDays(4);
    fourDate = LocalDateTime.now().minusDays(3);
    threeDate = LocalDateTime.now().minusDays(2);
    twoDate = LocalDateTime.now().minusDays(1);
    dateFormat = DateTimeFormatter.ofPattern("dd-MM-yyyy");
    formattedDate = recordedDate.format(dateFormat);
    formattedSevenDate = sevenDate.format(dateFormat);
    formattedSixDate = sixDate.format(dateFormat);
    formattedFiveDate = fiveDate.format(dateFormat);
    formattedFourDate = fourDate.format(dateFormat);
    formattedThreeDate = threeDate.format(dateFormat);
    formattedTwoDate = twoDate.format(dateFormat);
}
```

This method creates a line graph using the library "MPAndroidChart." It sets up the X-axis labels using the dates from **xValues** and the Y-axis values using heart rate data from the **readDay7** to **readDay1** variables. The application will fetch all the body vital data for each of the last seven days (day 1 to day 7) from the Firebase Realtime Database. The data is read from the "Monitored Data"

node under the user's node, corresponding to each date. It then configures the appearance of the line chart, such as line color and axis text color. The **LineDataSet** object represents the actual data set to be displayed on the chart. Finally, the method sets the created **LineData** object to the **LineChart**, and calls **invalidate ()** to redraw the chart.

```
day7Val = readDay7.floatValue();
day6Val = readDay6.floatValue();
day5Val = readDay5.floatValue();
day4Val = readDay4.floatValue();
day3Val = readDay3.floatValue();
day2Val = readDay2.floatValue();
day1Val = readDay1.floatValue();

List<Entry> entries = new ArrayList<>();
entries.add(new Entry( x: 0, day7Val));
entries.add(new Entry( x: 1, day6Val));
entries.add(new Entry( x: 2, day5Val));
entries.add(new Entry( x: 3, day4Val));
entries.add(new Entry( x: 4, day3Val));
entries.add(new Entry( x: 5, day2Val));
entries.add(new Entry( x: 6, day1Val));

LineDataSet dataSet = new LineDataSet(entries, user);
dataSet.setColor(Color.BLUE);

LineData lineData = new LineData(dataSet);

lineChart.setData(lineData);
lineChart.invalidate();
```

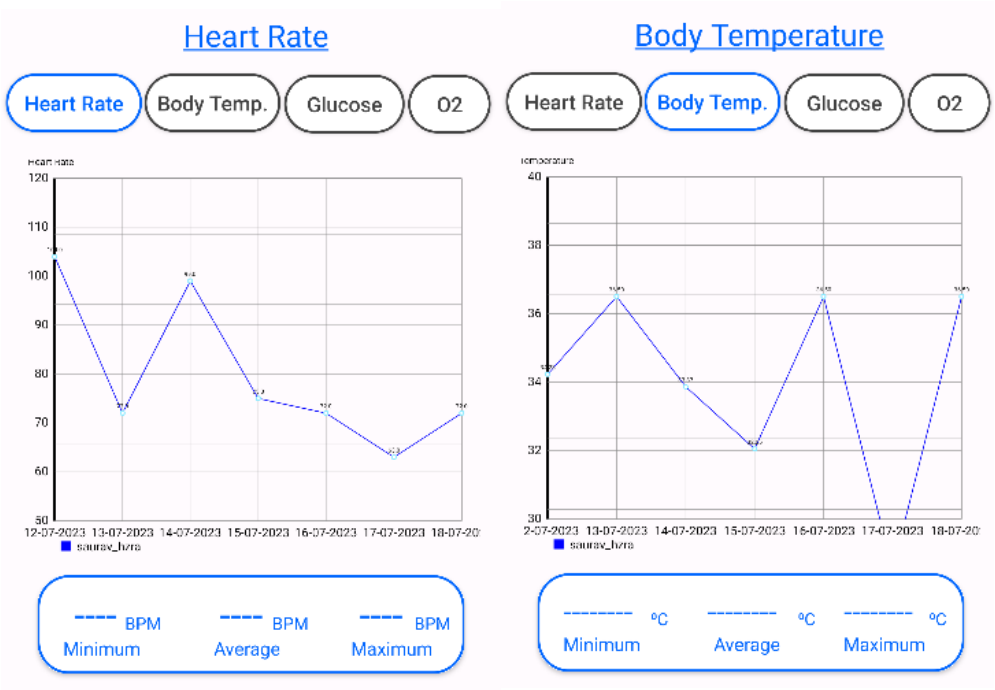
```
if (snapshot.hasChild(user)) {
    Log.d(TAG, msg: "I DO LOOK THROUGH DAY 2");

    databaseReference.child( pathString: "Monitored Data").child(user).child(formattedTwoDate).addListenerForSingleValueEvent(new Value
        @Override
        public void onDataChange(@NonNull DataSnapshot datasnapshot) {
            if (datasnapshot.hasChildren()) {
                Log.d(TAG, msg: "I COME TO DAY2");
                String time = datasnapshot.getValue().toString();
                final String timestamp = time.substring(1, 9);
                databaseReference.child( pathString: "Monitored Data").child(user).child(formattedTwoDate).child(timestamp).addListener
                    @Override
                    public void onDataChange(@NonNull DataSnapshot datasnap) {
                        if (datasnap.hasChild( path: "Glucose"))
                        {
                            Log.d(TAG, msg: "I COME TO DAY2 FOR SURE");
                            readDay2 = datasnap.child( path: "Glucose").getValue(Double.class);
                            Log.d(TAG, String.valueOf(readDay2));
                            createGraph();
                        }
                    }
```

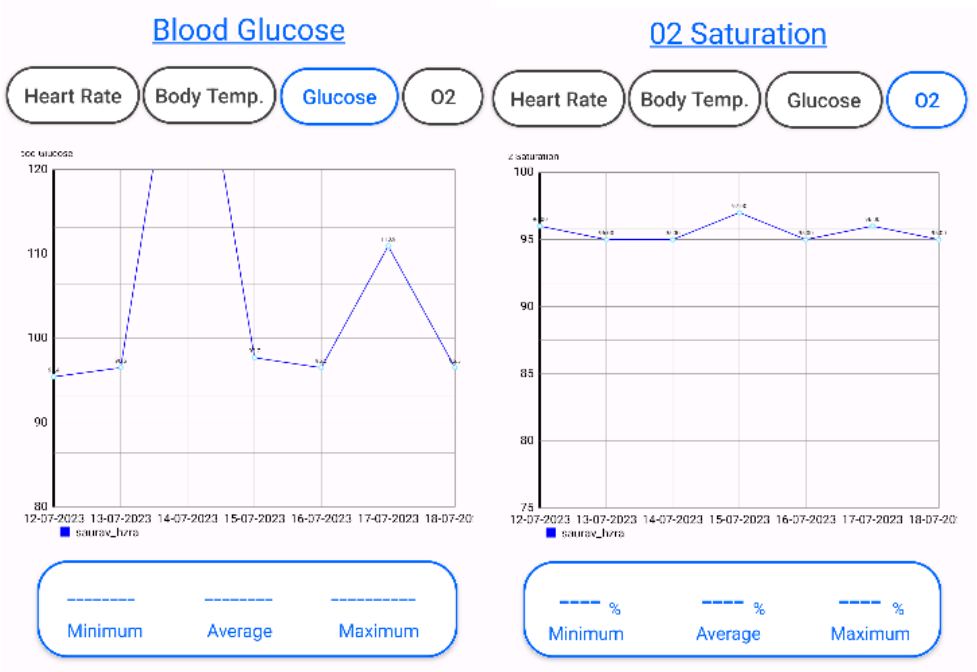
This application also calculates the minimum, maximum, and average heart rate values based on the data collected from the seven days (**readDay7** to **readDay1**). It stores the results in the **min**, **max**, and **avg** variables.

```
databaseReference.child( pathString: "Monitored Data").child(user).child(formattedSixDate).addListenerFor
    @Override
    public void onDataChange(@NonNull DataSnapshot datasnapshot) {
        if (datasnapshot.hasChildren()) {

            String time = datasnapshot.getValue().toString();
            final String timestamp = time.substring(1, 9);
            databaseReference.child( pathString: "Monitored Data").child(user).child(formattedSixDate).ch
                @Override
                public void onDataChange(@NonNull DataSnapshot datasnap) {
                    if (datasnap.hasChild( path: "Glucose"))
                    {
                        final Double day6 = datasnap.child( path: "Glucose").getValue(Double.class);
                        avgTotal += day6;
                        if (day6 < min)
                        {
                            min = day6;
                        }
                        if (day6 > max)
                        {
                            max = day6;
                        }
                        minRecorded.setText(String.valueOf(min));
                        avg = Double.valueOf(df.format( number: avgTotal / 7));;
                        avgRecorded.setText(String.valueOf(avg));
                        maxRecorded.setText(String.valueOf(max));
                    }
                }
```

The code can display individual fragments for each body vital for a specific user, fetched from a Firebase Realtime Database, and allows the user to switch between the parameters using buttons.

```
heartStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new StatsFragment()); }
});

o2StatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new O2StatsFragment()); }
});

tempStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new TempStatsFragment()); }
});

glucoseStatsButton.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) { home.replaceFragment(new GlucoseStatsFragment()); }
});
```





4.4.8 MediBot Page

This code defines a custom RecyclerView Adapter called `MessageAdapter`, which is used to display chat messages in a chat interface. RecyclerView is a commonly used UI component in Android that efficiently displays large sets of data in a list or grid format.

- `chatModelArrayList`: An ArrayList of `ChatModel` objects that holds the chat messages to be displayed in RecyclerView.
- ‘context`: The Context of the activity or fragment that uses this adapter.
- Constructor: The constructor is defined to receive an ArrayList of `ChatModel` objects and the Context. It is used to pass the data to the adapter when it is initialized.

```
42 usages
public ChatModel(String message, String sender)
{
    this.message = message;
    this.sender = sender;
}
```

```
public class Message {

    3 usages
    private String message;

    public Message(String message) { this.message = message; }

    public String getMessage() { return message; }

    public void setMessage(String message) { this.message = message; }

}
```

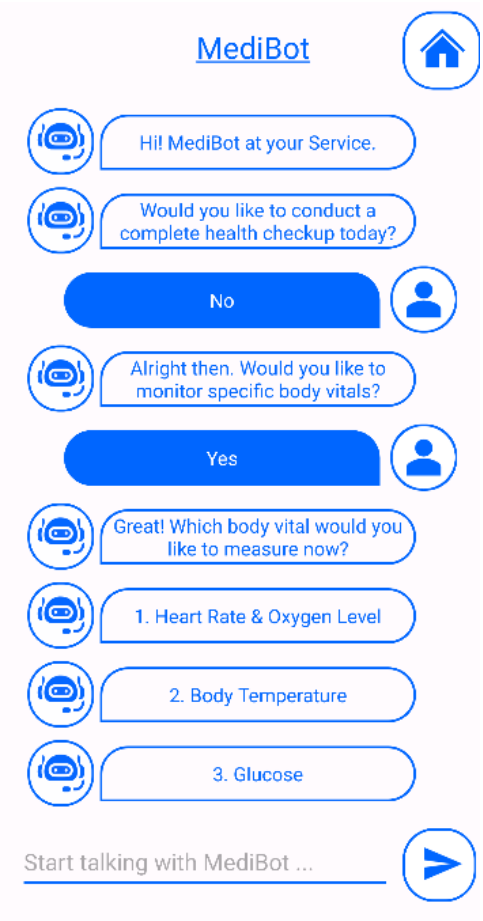
- onBindViewHolder Method: This method is called to bind data to the views of each item in the RecyclerView. It receives the ViewHolder and the position of the item to be bound.
    - The method retrieves the `ChatModel` object corresponding to the current position from the `chatModelArrayList`.
    - It uses another switch statement based on the sender of the chat message ("user" or "bot") to decide which ViewHolder to use (`UserViewHolder` or `BotViewHolder`).
    - The appropriate ViewHolder is cast from the generic `RecyclerView.ViewHolder` to the specific `UserViewHolder` or `BotViewHolder`, and the message text is set to the corresponding TextView (`userMsg` or `botMsg`).

```
4 usages
private ArrayList<ChatModel> chatModelArrayList;
1 usage
private Context context;

1 usage
public MessageAdapter(ArrayList<ChatModel> chatModelArrayList, Context context) {
    this.chatModelArrayList = chatModelArrayList;
    this.context = context;
}

@NonNull
@Override
public RecyclerView.ViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
    View view;
    switch (viewType) {
        case 0:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_container_sent_message, parent,  attachToRoot: false);
            return new UserViewHolder(view);

        case 1:
            view = LayoutInflater.from(parent.getContext()).inflate(R.layout.item_container_received_message, parent,  attachToRoot: false);
            return new BotViewHolder(view);
    }
    return null;
}
```

- getItemViewType Method: This method determines the view type of each item in the RecyclerView based on the sender of the chat message ("user" or "bot").
    - If the sender is "user," it returns `0`, which corresponds to the view type for the layout representing a message sent by the user.
    - If the sender is "bot," it returns `1`, which corresponds to the view type for the layout representing a message received from the bot.
    - If the sender is neither "user" nor "bot," it returns `-1`, which indicates an unknown view type (though this is not used in this specific adapter).

```
2 usages
public static class UserViewHolder extends RecyclerView.ViewHolder {
    2 usages
    TextView userMsg;
    1 usage
    public UserViewHolder(@NonNull View itemView) {
        super(itemView);
        userMsg = itemView.findViewById(R.id.sentMessage);
    }

}

2 usages
public static class BotViewHolder extends RecyclerView.ViewHolder {
    2 usages
    TextView botMsg;
    1 usage
    public BotViewHolder(@NonNull View itemView) {
        super(itemView);
        botMsg = itemView.findViewById(R.id.receivedMessage);
    }

}
```

Overall, this `MessageAdapter` class facilitates the display of chat messages in a RecyclerView, with each message's layout determined based on the sender (user or the chatbot).



4.5 Chatbot Development

To develop the medical chatbot, coding is done in two languages. Since the application is made in Android Studio, Java is required to develop heavy aspects of the chatbot. But the chatbot coding also requires Python since the monitoring and the recommendation is done in Python. The input prompts for the recommendation and checkup will be overall completed through the application and thus the coding with both these languages need to be connected through the Firebase connection.

4.5.1 Python Regex Function

The chatbot references the Symptoms file that contains some text data and uses the re.findall () function to search for lines that have the word a diagnosis in them, retrieve this information and then prompt the user to answer a question regarding if they are experiencing a symptom or not.

```python
import re
symptomfile = open("Symptoms.txt", "r")

identified_diagnosis = "Healthy"
sympline = []
for oneline in symptomfile:
    found = identified_diagnosis
    if(re.findall(found, oneline)):
        print(oneline)
        cleaned_line = oneline.replace(found + ":", "")
        cleaned_line2 = cleaned_line.replace("\n", "")
        sympline.append(re.split(",", cleaned_line2))
```

4.5.2 Python YamL Library

The YAML library contains a list of questions and prompts as keys and their corresponding responses as values. These keys and values are stored as a Python dictionary which the chatbot module uses to provide appropriate responses to user input. The coded chatbot module is pretty straight-forward, simply capable of answering 'yes' or 'no' questions based on predefined responses in the YAML file.

The main function of the chatbot calls the load_responses () function to load the responses from the YAML file and stores them in the responses variable. The chatbot enters a loop where it continuously takes user input using input. After the input has been registered, the code calls get_response () with the user's input and responds according to the stored values in the dictionary, printing the chatbot's response.

The get_response () function takes two arguments - the user's input and all the responses (the dictionary of responses loaded from the YAML file). The code first converts the user input into complete lowercase for case-insensitive comparison. If the user input exists as a key in the responses dictionary, the method returns the corresponding value (the chatbot's response). If not, it returns a default "I'm sorry, I do not understand. Please answer with 'yes' or 'no'."

```python
import yaml
1 usage
def load_responses():
    with open('responses.yaml', 'r') as file:
        return yaml.safe_load(file)


1 usage
def get_response(user_input, responses):
    user_input = user_input.lower()
    if user_input in responses:
        return responses[user_input]
    return "I'm sorry, I don't understand. Please answer with 'yes' or 'no'."


def main():
    responses = load_responses()

    print("Chatbot: Hi! I'm a simple chatbot. You can ask me 'yes' or 'no' questions.")
    while True:
        user_input = input("You: ")
        if user_input.lower() == 'exit':
            print("Chatbot: Goodbye!")
            break
        response = get_response(user_input, responses)
        print("Chatbot:", response)
```

4.5.3 Java Development

The MediBot chatbot interacts with the user to conduct a health checkup. It communicates with the user through a simple chat-like interface and performs different health checkup tasks based on the user's responses.

`startBot()` Method:

- This method is the starting point of the chatbot's conversation.

- It greets the user and asks if they want to conduct a complete health checkup.

- It sets up a click listener on the send button (`sendMsgBtn`) to handle user responses.

```
private void startBot() {

    databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Username").setValue(username);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Hi! MediBot at your Service.", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 3000);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Would you like to conduct a complete health checkup today?", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 6000);
```

`checkup()` Method:

- This method initiates a complete health checkup.

- It also includes arrays (`checkupYes` and `checkupNo`) that contain possible responses indicating "yes" or "no" to specific questions.

```
10 usages
String[] checkupYes = {"yes", "i do", "want to", "would", "i'd", "like"};
10 usages
String[] checkupNo = {"no", "i do not", "don't", "would not", "not", "wouldn't"};
```

- It sets `conductCheckup` and `compCheckup` to true to indicate that a complete checkup is being conducted.

- It calls `heartO2Monitor()` to start monitoring heart rate and oxygen level.

```
private void heartO2Monitor() {

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Okay. Let's measure your heart rate and blood oxygen level.", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 3000);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Place your index finger on the Pulse Sensor.", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 4500);

    if (heart == -1 && o2 == -1) {
        heart = 1;
        o2 = 1;
        conductCheckup = true;

        databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Conduct Checkup").setValue(conductCheckup);
                databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Heart Checkup").setValue(heart);
                databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "O2 Checkup").setValue(o2);
            }
```

`heartO2Monitor()`, `tempMonitor()`, and `glucoseMonitor()` are responsible for monitoring heart rate and oxygen level, body temperature, and glucose level, respectively.

- They show appropriate messages to guide the user on how to conduct the measurement and record the respective measurements.

- After recording measurements, they update the Firebase database to indicate that the respective checkup is complete.

`heartO2Wait()`, `tempWait()`, and `glucoseWait()` methods represent waiting for the measurements to be completed before proceeding with the next step of the checkup. The complete implementation may contain these methods to handle such situations.

If the temperature data is available (temp == 0), it shows the body temperature value (**tempValue**) to the user after a delay of 2.5 seconds. Then, after a delay of 3.5 seconds, it proceeds to the **completeCheckup()** method to check if all the necessary health parameters have been collected.

```
if (heart != 0 && o2 != 0) {
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Monitoring Heart Rate and Oxygen Level .....", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 3000);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() { heartO2Wait(); }
    }, delayMillis: 4500);

} else {
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Heart Rate : " + heartValue + " BPM", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 3000);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Oxygen Level : " + o2Value + " %", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
```

This method, **completeCheckup()**, checks if all the health parameters (**heartCheck**, **tempCheck**, **glucoseCheck**, **o2Check**) have been collected (**compCheckup**). If **compCheckup** is true, it means all parameters are collected, and the chatbot can proceed with making predictions based on the data.

If all parameters are available (heartCheck == 0, tempCheck == 0, glucoseCheck == 0, o2Check == 0), it calls the **setPrediction()** method after a delay of 2 seconds to make predictions.

```
retrieveParam();
if (compCheckup.equals(true)) {
    if (heartCheck == 0 && tempCheck == 0 && glucoseCheck == 0 && o2Check == 0) {
        handler.postDelayed(new Runnable() {
            @Override
            public void run() { setPrediction(); }
        }, delayMillis: 2000);
    }
    if (heartCheck == 0 && tempCheck == -1 && glucoseCheck == -1 && o2Check == 0) {
        glucoseMonitor();
    }
    if (heartCheck == 0 && tempCheck == -1 && glucoseCheck == 0 && o2Check == 0) {
        tempMonitor();
    }
} else {
```

```
handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        chatModelArrayList.add(new ChatModel( message: "Great! Which body vital would you like to measure now?", BOT_KEY));
        messageAdapter.notifyDataSetChanged();
    }
}, delayMillis: 2000);

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        chatModelArrayList.add(new ChatModel( message: "1. Heart Rate & Oxygen Level", BOT_KEY));
        messageAdapter.notifyDataSetChanged();
    }
}, delayMillis: 3000);

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        chatModelArrayList.add(new ChatModel( message: "2. Body Temperature", BOT_KEY));
        messageAdapter.notifyDataSetChanged();
    }
}, delayMillis: 4000);

handler.postDelayed(new Runnable() {
    @Override
    public void run() {
        chatModelArrayList.add(new ChatModel( message: "3. Glucose", BOT_KEY));
        messageAdapter.notifyDataSetChanged();
    }
```

`indCheckup()` Method:

- This method is called when the user wants to monitor specific body vitals.

- It asks the user which body vital they want to measure (heart rate, temperature, or glucose) and proceeds accordingly.

`chooseMonitor()` Method:

- This method is called to handle the user's choice of monitoring a specific body vital (heart rate, temperature, or glucose).

This method, **glucoseMonitor()**, is responsible for monitoring the body temperature. It first calls **retrieveParam()** to get the current health parameters from the database. If the temperature data (**temp**) is not equal to 0 (i.e., data is not available), it shows the message "Monitoring Body Glucose ....." after a delay of 3 seconds using **handler.postDelayed()**. After another delay of 4.5 seconds, it calls **glucoseMonitor()** again to repeat the temperature monitoring process until the temperature data becomes available (temp == 0).

This method, **completeCheckup()**, checks if all the health parameters (**heartCheck**, **tempCheck**, **glucoseCheck**, **o2Check**) have been collected (**compCheckup**). If **compCheckup** is true, it means all parameters are collected, and the chatbot can proceed with making predictions based on the data.

If some parameters are missing, it checks which ones are missing and takes appropriate action. For example, if **heartCheck == 0** and **tempCheck == -1**

and **glucoseCheck == -1** and **o2Check == 0**, it means the chatbot has collected heart and oxygen data but needs temperature and glucose data. In this case, it proceeds to glucose monitoring.

```
if ((fastingChoice.toLowerCase().contains("yes")) || (fastingChoice.toLowerCase().contains("have"))) {
    databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Fasting").setValue(false);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}

if ((fastingChoice.toLowerCase().contains("no")) || (fastingChoice.toLowerCase().contains("have not"))) {
    databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Fasting").setValue(true);
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
}
```

```
if (glucose == -1) {
    glucose = 1;
    conductCheckup = true;

    databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Conduct Checkup").setValue(conduct
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Glucose Checkup").setValue(glucose
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
```

This method, **setPrediction()**, is responsible for making predictions based on the collected health data. It retrieves the user's last diagnosis and recommendation from the database after a delay of 1 second using **handler.postDelayed()**. Then, after certain delays, it displays the diagnosis and recommendation to the user using **chatModelArrayList.add()** and **messageAdapter.notifyDataSetChanged()**.

This method, **endMediBot()**, is responsible for completing the monitoring process. It records the current date and time, saves monitored data and user

information to the database, resets parameters, and ends the chatbot interaction.

```
private void endMediBot() {

    String formattedTime = "", formattedDate = "";
    LocalDateTime recordedTime, recordedDate;
    DateTimeFormatter timeFormat, dateFormat;

    if (android.os.Build.VERSION.SDK_INT >= android.os.Build.VERSION_CODES.O) {
        recordedTime = LocalDateTime.now();
        timeFormat = DateTimeFormatter.ofPattern("HH-mm-ss");
        recordedDate = LocalDateTime.now();
        dateFormat = DateTimeFormatter.ofPattern("dd-MM-yyyy");
        formattedTime = recordedTime.format(timeFormat);
        formattedDate = recordedDate.format(dateFormat);
    }

    retrieveParam();
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Monitoring is completed.", BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 2000);
```

The application retrieves the current date and time using **LocalDateTime** and formats them into strings (**formattedDate** and **formattedTime**) to use as keys for storing monitored data in the database.

- It adds a message to the **chatModelArrayList** to inform the user that monitoring is completed.
- It updates the user's last recorded health parameters (**heartValue**, **o2Value**, **tempValue**, **glucoseValue**) in the Firebase database under their respective fields (**Last Heart**, **Last O2**, **Last Temp**, **Last Glucose**).
- It stores the monitored data (**heartValue**, **o2Value**, **tempValue**, **glucoseValue**) under the user's node in the Firebase database, using the date and time keys (**formattedDate** and **formattedTime**) to organize the data under appropriate timestamps.

```
1 usage
private void getPrediction() {

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {

            databaseReference.child( pathString: "Users").addListenerForSingleValueEvent(new ValueEventListener() {
                @Override
                public void onDataChange(@NonNull DataSnapshot snapshot) {
                    if (snapshot.hasChild(username)) {
                        diagnosis = snapshot.child(username).child( path: "Last Diagnosis").getValue(String.class);
                        recommendation = snapshot.child(username).child( path: "Last Recommendation").getValue(String.class);
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {

                }

            });

        }
    }, delayMillis: 1000);
```

```
    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Identified Diagnosis : " + diagnosis, BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 2500);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() {
            chatModelArrayList.add(new ChatModel( message: "Recommendation : " + recommendation, BOT_KEY));
            messageAdapter.notifyDataSetChanged();
        }
    }, delayMillis: 4000);

    handler.postDelayed(new Runnable() {
        @Override
        public void run() { endMediBot(); }
    }, delayMillis: 6000);
```

This function is involved in handling predictions related to the user's health condition. It checks whether compPrediction is false or true. If compPrediction is false, it means there are more questions to ask the user for a complete prediction. It performs the following tasks:

- It checks whether the symptomQuestion is not empty. If it is empty, it retrieves parameters (retrieveParam()) and recursively calls setPrediction() to continue asking more questions.

- If symptomQuestion is not empty, it checks if the questionAsked is the same as symptomQuestion. If it is, it recursively calls setPrediction() to continue asking more questions.
- If the above conditions are not met, it adds the symptomQuestion to the chatModelArrayList to ask the user a question.
- It sets up a click listener on the send button to handle the user's response to the question.
- Once the user responds (symptomResponse), it updates the Symptom Answer field in the Model node in the Firebase database with the user's response and recursively calls setPrediction() to continue the prediction process.

```
databaseReference.child( pathString: "Monitored Data").addListenerForSingleValueEvent(new ValueEventListener() {
    @Override
    public void onDataChange(@NonNull DataSnapshot snapshot) {
        databaseReference.child( pathString: "Monitored Data").child(username).child(finalFormattedDate).child(finalFormattedTime).child( pathSt
        databaseReference.child( pathString: "Monitored Data").child(username).child(finalFormattedDate).child(finalFormattedTime).child( pathSt
        databaseReference.child( pathString: "Monitored Data").child(username).child(finalFormattedDate).child(finalFormattedTime).child( pathSt
        databaseReference.child( pathString: "Monitored Data").child(username).child(finalFormattedDate).child(finalFormattedTime).child( pathSt
    }
```

Finally, it calls the **setFinalParam()** function to reset certain parameters in the **Model** node of the database to prepare for the next monitoring session.

```
1 usage
private void setFinalParam() {

    databaseReference.child( pathString: "Model").addListenerForSingleValueEvent(new ValueEventListener() {
        @Override
        public void onDataChange(@NonNull DataSnapshot snapshot) {
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Conduct Checkup").setValue(false);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Complete Checkup").setValue(false);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Complete Prediction").setValue(false);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Fasting").setValue(true);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Heart Checkup").setValue(-1);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Temperature Checkup").setValue(-1);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Glucose Checkup").setValue(-1);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "O2 Checkup").setValue(-1);
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Symptom Question").setValue("");
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Symptom Answer").setValue("");
            databaseReference.child( pathString: "Model").child(checkupModel).child( pathString: "Username").setValue("");
        }

        @Override
        public void onCancelled(@NonNull DatabaseError error) {

        }
    });
```

CHAPTER 5: SYSTEM TESTING

5.1 Unit Testing

5.1.1 Test Plan

Test Case ID	Importance Level	Test Description
IoT Architecture		
IOT001	High	Record & Store Heart Rate
IOT002	High	Record & Store Blood Oxygen Level
IOT003	High	Record & Store Body Temperature
IOT004	High	Record & Store Blood Glucose Level
IOT005	High	Transfer Data Online (Through the same internet connection.)
IOT006	High	Connect Multiple Sensors on the same Raspberry Pi
Application Interface		
APP001	High	Login to the Personal Healthcare Application
APP002	High	Register User Account
APP003	High	Register Medical Information
APP004	Low	Update User Account Details
APP005	Low	Update Medical Information

APP006	Medium	Set New Password from Forget Password Interface
APP007	High	Retrieve Personal Monitored Data
APP008	Medium	Display Heart Rate Graph
APP009	Medium	Display Oxygen Saturation Graph
APP010	Medium	Display Body Temperature Graph
APP011	Medium	Display Blood Glucose Graph
APP012	High	Access Medical Chatbot
Disease Identification & Recommendation		
DIR001	High	Identify Healthy Person
DIR002	High	Identify Fever & Recommend Solution
DIR003	High	Identify Flu & Recommend Solution
DIR004	High	Identify Covid-19 & Recommend Solution
DIR005	High	Identify Common Cold & Recommend Solution
DIR006	High	Identify Asthma & Recommend Solution
DIR007	High	Identify Hyperglycemia & Recommend Solution

DIR008	High	Identify Pre-Diabetes & Recommend Solution
DIR009	High	Identify Hypoglycemia & Recommend Solution
DIR010	High	Identify Pneumonia & Recommend Solution
DIR011	High	Identify Stress & Recommend Solution
DIR012	High	Identify Arrhythmia & Recommend Solution
DIR013	High	Identify Coronary Artery Disease & Recommend Solution
DIR014	High	Identify Lung Cancer & Recommend Solution
DIR015	High	Identify Breast Cancer for Females & Recommend Solution
DIR016	High	Identify Leukemia & Recommend Solution
Medical Chatbot		
CBT001	High	Conduct Checkup
CBT002	Medium	Accept Message from User
CBT003	High	Deliver Appropriate & Timely Response
CBT004	High	Activate Correct Sensor

CBT005	Medium	End Checkup
CBT006	High	Deliver any Identified Problem and its Recommended Solution

5.1.2 Test Cases

IoT Architecture

Test Case ID	IOT001
Test Description	Record & Store Heart Rate
Test Objective	st whether the heart rate of an individual can be red. sure that the recorded value is stored with correct date plication retrieval and future analysis.
Prerequisites	Users must use the health application to conduct an overall health checkup or only choose to record the heart rate.
Test Data	• Heart Rate Value
Test Procedure	1. Activate MAX30102 sensor from the chatbot or manually and conduct the monitoring process for the specific sensor.
Expected Result	1. The MAX30102 sensor starts recording and calculate average value of 10 recordings. 2. Firebase Realtime database is updated when monitoring has ended.
Actual Result	The user was able to successfully record their heart rate and the information was saved appropriately in the cloud storage.

Status	Pass

Test Case ID	IOT002
Test Description	Record & Store Blood Oxygen Level
Test Objective	1. To test whether the blood oxygen level of an individual can be monitored. 2. To ensure that the recorded value is stored with correct date for application retrieval and future analysis.
Prerequisites	Users must use the health application to conduct an overall health checkup or only choose to record the blood oxygen level.
Test Data	• Blood Oxygen Value
Test Procedure	1. Activate MAX30102 sensor from the chatbot or manually and conduct the monitoring process for the specific sensor.
Expected Result	1. The MAX30102 sensor starts recording and calculate average value of 10 recordings. 2. Firebase Realtime database is updated when monitoring has ended.
Actual Result	The user was able to successfully record their blood oxygen saturation and the information was saved appropriately in the cloud storage.
Status	Pass

Test Case ID	IOT003
Test Description	Record & Store Body Temperature
Test Objective	1. To test whether the body temperature of an individual can be monitored. 2. To ensure that the recorded value is stored with correct date for application retrieval and future analysis.
Prerequisites	Users must use the health application to conduct an overall health checkup or only choose to record the body temperature.
Test Data	• Body Temperature Value
Test Procedure	1. Activate MLX90614 sensor from the chatbot or manually and conduct the monitoring process for the specific sensor.
Expected Result	1. The MLX90614 sensor starts recording and gets the temperature value at the head / mouth of the tester. 2. Firebase Realtime database is updated when monitoring has ended.
Actual Result	The user was able to successfully record their body temperature value and the information was saved appropriately in the cloud storage.
Status	Pass

Test Case ID	IOT004
Test Description	Record & Store Blood Glucose Level

Test Objective	1. To test whether the blood glucose level of an individual can be monitored. 2. To ensure that the recorded value is stored with correct date for application retrieval and future analysis.
Prerequisites	Users must use the health application to conduct an overall health checkup or only choose to record the blood glucose level.
Test Data	• Blood Glucose Value
Test Procedure	1. Activate IR sensor from the chatbot or manually and conduct the monitoring process for the specific sensor.
Expected Result	1. The IR starts recording the glucose values when the individual's finger is brought close and calculates the average value of 10 recordings. 2. Firebase Realtime database is updated when monitoring has ended.
Actual Result	The user was able to successfully record their blood glucose value and the information was saved appropriately in the cloud storage.
Status	Pass

Test Case ID	IOT005
Test Description	Transfer Data Online (Through the same internet connection.)

Test Objective	1. To test whether the monitored information can be picked up remotely from the healthcare application. 2. To ensure that the monitored data is never lost.
Prerequisites	A sensor must be activated and must monitor a single / multiple body parameters set by the user.
Test Data	• Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value
Test Procedure	1. Activate a sensor from the chatbot or manually and conduct the monitoring process for the specific sensor. 2. Make sure the process is ended and check if the cloud storage is updated.
Expected Result	1. The recorded body vitals are sent to the Firebase Realtime database. 2. The recorded information is displayed in the application.
Actual Result	The user was able to successfully transfer data online. Upon completion of the monitoring process, the information was updated in the application after refreshing.
Status	Pass

Test Case ID	IOT006

Test Description	Connect Multiple Sensors on the same Raspberry Pi
Test Objective	1. To check whether multiple body vitals can be recorded together for a complete health checkup. 2. To identify any bugs or issues with the IoT architecture setup.
Prerequisites	The tests for monitoring and storing individual body vitals from sensors are successful.
Test Data	• Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value
Test Procedure	1. Build the model, connect, and coordinate the GPIO pins of the Raspberry Pi 4B+ Model. 2. Activate all sensors one after the other from the chatbot conduct the health checkup process for each sensor. 3. Test different combinations and orders of monitoring processes.
Expected Result	There is no error with the setup of the sensors. Each sensor and its program are identified on the Raspberry Pi and the system can record all the body vitals no matter the order.
Actual Result	The body vitals have to be recorded in a specific order: • Heart Rate – O2 – Blood Glucose – Body Temperature The sensors had to be connected to different buses to connect to the Raspberry Pi model and the pin setup

	cleared by the program each time, but the system can record all the parameters.
Status	Pass

Application Interface

Test Case ID	APP001
Test Description	Login to the Personal Healthcare Application
Test Objective	1. To ensure that a new user can login to their account to monitor their health statistics. 2. To make sure that the user can make use of the IoT model. 3. To safeguard private medical information to themselves.
Prerequisites	Healthcare Application is downloaded, and the user has a previously registered account.
Test Data	• Username – saurav_hzra • Password – 1810
Test Procedure	1. User fills in the fields with their username and corresponding password. 2. User presses the ‘Login’ button. 3. The information on the fields is checked with the Firebase database to identify if the information matches with saved data.
Expected Result	1. The healthcare application opens up. 2. User is directed to the Homepage and access.

	3. User's monitored information is retrieved.
Actual Result	The user accesses the healthcare application, with their personal information retrieved and access all the system's functionalities.
Status	Pass

Test Case ID	APP002
Test Description	Register User Account
Test Objective	1. To ensure that a new user can register an account to monitor their health statistics. 2. To make sure that the registered user can make use of the IoT model.
Prerequisites	Healthcare Application is downloaded and opened for use. The user has not registered an account previously.
Test Data	• Name - Saurav • Date of Birth – 16/10/2001 • Email – saurav.hzra@gmail.com • Username – saurav_hzra • Password – 1810
Test Procedure	1. User fills in all the fields with their information. 2. User presses the 'Continue' button. 3. The information from the fields is saved as a new record in the Firebase database.

Expected Result	1. User's account details are saved, and the user is directed to the next registration page to update their medical information.
Actual Result	1. User's account details are saved, and the user is directed to the next registration page to update their medical information.
Status	Pass

Test Case ID	APP003
Test Description	Register Medical Information
Test Objective	1. To ensure that a new user can register their medical information to monitor their health statistics. 2. To make sure that the registered user can make use of this medical information when using the IoT model.
Prerequisites	Healthcare Application is downloaded and opened for use. The user is in the process of registering an account.
Test Data	• Sex - M • Height – 167.9 cm • Weight – 53.2 • Medical History – “” • Current Illness – “”
Test Procedure	1. User fills in all the fields with their information.

	2. User presses the 'Register' button. 3. The information from the fields is updated with new fields on the record corresponding to the user's account.
Expected Result	1. User's account details are saved. 2. The user receives a message to indicate that the account is made. 3. User is directed back to the Login page.
Actual Result	User's account details are saved, and the user is directed to the login page so they can access the healthcare application and its functions.
Status	Pass

Test Case ID	APP004
Test Description	Update User Account Details
Test Objective	1. To ensure that a registered user can update their account details to monitor their health statistics. 2. To make sure that the registered user can make use of the IoT model and application.
Prerequisites	The user is previously registered and is on the User Profile page of the application
Test Data	• Name - Saurav • Date of Birth – 16/10/2001 • Email – saurav.hzra@gmail.com • Password – 1810

Test Procedure	1. User updates the fields with changed information. 2. User presses the Update' button. 3. The information from the fields is updated with new fields on the record corresponding to the user's account.
Expected Result	1. User's account details are updated on the Firebase database. 2. The user receives a message indicating that the user account has been updated.
Actual Result	User's account details are saved, and the user can access their updated account information after re-logging into the healthcare application. However, the user cannot change their username.
Status	Pass

Test Case ID	APP005
Test Description	Update Medical Information
Test Objective	1. To ensure that a registered user can update their medical information. 2. To make sure that the registered user can make use of the IoT model and application with up-to-date medical information.
Prerequisites	The user is previously registered and is on the User Profile page of the application
Test Data	• Sex - M • Height – 167.9 cm

	• Weight – 53.2 • Medical History – “” • Current Illness – “”
Test Procedure	1. User updates the fields with changed information. 2. User presses the Update’ button. 3. The information from the fields is updated on the record corresponding to the user’s account.
Expected Result	1. User’s personal details are updated on the Firebase database. 2. The user receives a message indicating that the medical information has been updated.
Actual Result	User’s medical details are saved, and the user can access their updated medical information after re-logging into the healthcare application.
Status	Pass

Test Case ID	APP006
Test Description	Set New Password from Forget Password Interface
Test Objective	1. To ensure that the user can retrieve their account information and previously recorded body vital values. 2. To make sure there is a method to
Prerequisites	The user of the healthcare application must be registered but does not remember their password to log in.

Test Data	• Username • Email
Test Procedure	1. User enters their email to receive a One Time Password (OTP). 2. User enters the OTP and if they match, the user is allowed to create their new password. 3. The new password is updated with the user account on the Firebase Database
Expected Result	1. The password on the cloud database is updated. 2. User can login using the existing username and the newly set password.
Actual Result	1. The password on the cloud database is updated. 2. User can login using the existing username and the newly set password.
Status	Pass

Test Case ID	APP007
Test Description	Retrieve Personal Monitored Data
Test Objective	1. To check whether the user can securely collect their own previously monitored, stored information. 2. To ensure there is no chance of data invasion.
Prerequisites	The user must be a registered user that can login to the healthcare application.

Test Data	• Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value • User Account Details • Personal Information
Test Procedure	1. The user's username is referenced to retrieve all the information from the Firebase database. 2. The application retrieves and presents all the data associated with the username in the appropriate locations
Expected Result	1. The user's past monitored values can be displayed through the Body Vital graphs. 2. The user retrieves their account and medical information and places it on Profile page.
Actual Result	1. The user's past monitored values can be displayed through the Body Vital graphs. 2. The user retrieves their account and medical information and places it on Profile page.
Status	Pass

Test Case ID	APP008
Test Description	Display Heart Rate Graph
Test Objective	1. To check whether the application can display all the Heart Rate values of the user in this week.

	2. To check whether the system can report the minimum, maximum and average of the Heart Rate values.
Prerequisites	The user is logged in and presses the Heart Rate Statistics option or the “Stats” button on the Navigation Bar.
Test Data	• Heart Rate Values – Day 1, Day 2, Day 3, Day 4, Day 5, Day 6, Day 7
Test Procedure	1. User opens the Heart Rate page. 2. The application retrieves the current date and pulls all the stored heart rate values for all the days in the prior week. 3. The minimum and maximum values are set after going through each data item and the average is calculated.
Expected Result	1. A line graph with Heart Rate on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Heart Rate values. 3. The minimum, maximum and average is identified and displayed on the application page.
Actual Result	1. A line graph with Heart Rate on the Y-axis and Dates in the week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Heart Rate values. 3. The minimum, maximum and average is identified and displayed on the application page.

Status	Pass

Test Case ID	APP009
Test Description	Display Oxygen Saturation Graph
Test Objective	1. To check whether the application can display all the Blood Oxygen Level values of the user in this week. 2. To check whether the system can report the minimum, maximum and average of the Blood Oxygen values.
Prerequisites	The user is logged in and presses the O2 Statistics option on the Statistics page.
Test Data	• Blood Oxygen Values – Day 1, Day 2, Day 3, Day 4, Day 5, Day 6, Day 7
Test Procedure	1. User opens the O2 page. 2. The application retrieves the current date and pulls all the stored blood oxygen values for all the days in the prior week. 3. The minimum and maximum values are set after going through each data item and the average is calculated.
Expected Result	1. A line graph with O2 Saturation on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Blood Oxygen Level values.

	3. The minimum, maximum and average is identified and displayed on the application page.
Actual Result	1. A line graph with O2 Saturation on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Blood Oxygen Level values. 3. The minimum, maximum and average is identified and displayed on the application page.
Status	Pass

Test Case ID	APP010
Test Description	Display Body Temperature Graph
Test Objective	1. To check whether the application can display all the body temperature values of the user in this week. 2. To check whether the system can report the minimum, maximum and average of the Body Temperature values.
Prerequisites	The user is logged in and presses the Body Temp. Statistics option on the Statistics page.
Test Data	• Body Temperature Values – Day 1, Day 2, Day 3, Day 4, Day 5, Day 6, Day 7
Test Procedure	1. User opens the Body Temp. page.

	2. The application retrieves the current date and pulls all the stored body temperature values for all the days in the prior week. 3. The minimum and maximum values are set after going through each data item and the average is calculated.
Expected Result	1. A line graph with Body Temperature on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Body Temperature values. 3. The minimum, maximum and average is identified and displayed on the application page.
Actual Result	1. A line graph with Body Temperature on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Body Temperature values. 3. The minimum, maximum and average is identified and displayed on the application page.
Status	Pass

Test Case ID	APP011
Test Description	Display Blood Glucose Graph
Test Objective	1. To check whether the application can display all the Blood Glucose values of the user in this week.

	2. To check whether the system can report the minimum, maximum and average of the Blood Glucose values.
Prerequisites	The user is logged in and presses the Glucose Statistics option on the Statistics page.
Test Data	• Blood Glucose Values - Day 1, Day 2, Day 3, Day 4, Day 5, Day 6, Day 7
Test Procedure	1. User opens the Blood Glucose. page. 2. The application retrieves the current date and pulls all the stored blood glucose values for all the days in the prior week. 3. The minimum and maximum values are set after going through each data item and the average is calculated.
Expected Result	1. A line graph with Blood Glucose on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Blood Glucose values. 3. The minimum, maximum and average is identified and displayed on the application page.
Actual Result	1. A line graph with Blood Glucose on the Y-axis and Dates in the last week on the X-axis is produced. 2. The line is a straight-line graph which connects through all plotted points – recorded Blood Glucose values. 3. The minimum, maximum and average is identified and displayed on the application page.

Status	Pass

Test Case ID	APP012
Test Description	Access Medical Chatbot
Test Objective	1. To check whether the user can access the medical chatbot. 2. To ensure the use can conduct a health checkup and monitor body vitals using the IoT model.
Prerequisites	The user must be a registered user that can login to the healthcare application.
Test Data	• MediBot Button Click
Test Procedure	1. User presses the “MediBot” button the navigation bar or presses the checkup button the homepage.
Expected Result	1. The user is directed to the Chatbot Page. 2. The MediBot starts the conversation and asks user if they would like to conduct a health checkup.
Actual Result	The user can access the MediBot and interact by typing messages to the chatbot. The chatbot starts a conversation and asks whether the user would like to conduct a checkup.
Status	Pass

Disease Identification & Recommendation

Test Case ID	DIR001
Test Description	Identify Healthy Person
Test Objective	1. To check if the expert system can accept monitored values and identify “Healthy” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed and there are no symptoms experienced and stated by the patient.
Test Data	• Heart Rate Value – 75 BPM • Oxygen Level Value – 98 % • Body Temperature Value - 36.9 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “No symptoms”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Healthy” as Diagnosis 2. Identifies “Continue having a fun life :)” as Recommendation.
Actual Result	1. Identifies “Healthy” as Diagnosis

	2. Identifies “Continue having a fun life :)” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR002
Test Description	Identify Fever & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Fever” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 92 BPM • Oxygen Level Value – 95 % • Body Temperature Value – 37.8 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input – “Headache”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches.

	3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Fever” as Diagnosis 2. Identifies “Consider Aspirin or Ibuprofen Tablets for treatment.” as Recommendation.
Actual Result	1. Identifies “Fever” as Diagnosis 2. Identifies “Consider Aspirin or Ibuprofen Tablets for treatment.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR003
Test Description	Identify Flu & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Flu” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 104 BPM • Oxygen Level Value – 94 % • Body Temperature Value – 35.87 ºC

	• Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Vomiting”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Flu” as Diagnosis 2. Identifies “Consider taking Oseltamivir tablets or Flu Shots from a doctor.” as Recommendation.
Actual Result	1. Identifies “Flu” as Diagnosis 2. Identifies “Consider taking Oseltamivir tablets or Flu Shots from a doctor.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR004
Test Description	Identify Covid-19 & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Covid-19” as a diagnosis of a complete health checkup.

	2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 104 BPM • Oxygen Level Value – 91 % • Body Temperature Value - 38.16 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Sore Throat”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Covid-19” as Diagnosis 2. Identifies “Isolate yourself, Drink hot water and inform medical facility.” as Recommendation.
Actual Result	1. Identifies “Covid-19” as Diagnosis 2. Identifies “Isolate yourself, Drink hot water and inform medical facility.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR005
Test Description	Identify Common Cold & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Common Cold” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 101 BPM • Oxygen Level Value – 95 % • Body Temperature Value - 35.87 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Runny Nose”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Common Cold” as Diagnosis 2. Identifies “Oxymetazoline Nasal Spray with Benzonatate tablets.” as Recommendation.
Actual Result	1. Identifies “Common Cold” as Diagnosis 2. Identifies “Oxymetazoline Nasal Spray with Benzonatate tablets.” as Recommendation.

	The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR006
Test Description	Identify Asthma & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Asthma” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 112 BPM • Oxygen Level Value – 88 % • Body Temperature Value – 37.94 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Breathing Difficulty”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.

Expected Result	1. Identifies “Asthma” as Diagnosis 2. Identifies “Get an Aerosol or Ventolin Inhaler with Prednisolone tablets.” as Recommendation.
Actual Result	1. Identifies “Asthma” as Diagnosis 2. Identifies “Get an Aerosol or Ventolin Inhaler with Prednisolone tablets.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR007
Test Description	Identify Hyperglycemia & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Hyperglycemia” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 92 BPM • Oxygen Level Value – 97 % • Body Temperature Value – 37.8 ºC

	• Blood Glucose Value - 126.7 mg / dl • Symptom Input - “Extreme Thirst”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Hyperglycemia” as Diagnosis 2. Identifies “You require Prescribed Insulin Shots and Glucotrol tablets.” as Recommendation.
Actual Result	1. Identifies “Hyperglycemia” as Diagnosis 2. Identifies “You require Prescribed Insulin Shots and Glucotrol tablets.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR008
Test Description	Identify Pre-Diabetes & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Pre-Diabetes” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.

Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 92 BPM • Oxygen Level Value – 97 % • Body Temperature Value – 37.8 ºC • Blood Glucose Value - 96.89 mg / dl • Symptom Input - “Extreme Thirst”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Pre-Diabetes” as Diagnosis 2. Identifies “Consider taking Metformin Tablets.” as Recommendation.
Actual Result	1. Identifies “Pre-Diabetes” as Diagnosis 2. Identifies “Consider taking Metformin Tablets.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR009

Test Description	Identify Hypoglycemia & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Hypoglycemia” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 92 BPM • Oxygen Level Value – 94 % • Body Temperature Value – 37.69 ºC • Blood Glucose Value – 72.76 mg / dl • Symptom Input - “Sweating or Shivering”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Hypoglycemia” as Diagnosis 2. Identifies “You require Timely Glucagon Shots and Metformin tablets.” as Recommendation.
Actual Result	1. Identifies “Hypoglycemia” as Diagnosis 2. Identifies “You require Timely Glucagon Shots and Metformin tablets.” as Recommendation.

	The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR010
Test Description	Identify Pneumonia & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Pneumonia” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 75 BPM • Oxygen Level Value – 97 % • Body Temperature Value - 36.9 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Chest Pain”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.

Expected Result	1. Identifies “Pneumonia” as Diagnosis 2. Identifies “You require Penicillin G tablets or Augmentin tablets.” as Recommendation.
Actual Result	1. Identifies “Pneumonia” as Diagnosis 2. Identifies “You require Penicillin G tablets or Augmentin tablets.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR011
Test Description	Identify Stress & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Stress” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 125 BPM • Oxygen Level Value – 97 % • Body Temperature Value - 36.9 ºC • Blood Glucose Value - 98.7 mg / dl

	• Symptom Input - “No symptoms”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Stress” as Diagnosis 2. Identifies “Consider taking Aspirin to reduce pain.” as Recommendation.
Actual Result	1. Identifies “Stress” as Diagnosis 2. Identifies “Consider taking Aspirin to reduce pain.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR012
Test Description	Identify Arrhythmia & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Arrhythmia” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.

Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 97 BPM • Oxygen Level Value – 94 % • Body Temperature Value - 34.67 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Chest Pain”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Arrhythmia” as Diagnosis 2. Identifies “Propafenone and Acebutolol Beta Blocker tablets” as Recommendation.
Actual Result	1. Identifies “Arrhythmia” as Diagnosis 2. Identifies “Propafenone and Acebutolol Beta Blocker tablets” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR013

Test Description	Identify Coronary Artery Disease & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Coronary Artery Disease” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 65 BPM • Oxygen Level Value – 92 % • Body Temperature Value - 38.5 ºC • Blood Glucose Value - 98.7 mg / dl • Symptom Input - “Breathing Difficulty”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Coronary Artery Disease” as Diagnosis 2. Identifies “Niacin supplements, Ranolazine and Acebutolol Beta Blocker tablets” as Recommendation.
Actual Result	1. Identifies “Coronary Artery Disease” as Diagnosis

	2. Identifies “Niacin supplements, Ranolazine and Acebutolol Beta Blocker tablets” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR014
Test Description	Identify Lung Cancer & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Lung Cancer” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 95 BPM • Oxygen Level Value – 87 % • Body Temperature Value - 38.1 ºC • Blood Glucose Value - 115.7 mg / dl • Symptom Input - “Coughing blood”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches.

	3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Lung Cancer” as Diagnosis 2. Identifies “Sadly you might require chemotherapy. Cisplatin and Carboplatin can help but you need a doctor.” as Recommendation.
Actual Result	1. Identifies “Lung Cancer” as Diagnosis 2. Identifies “Sadly you might require chemotherapy. Cisplatin and Carboplatin can help but you need a doctor.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR015
Test Description	Identify Breast Cancer & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Breast Cancer” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed on a female, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 95 BPM • Oxygen Level Value – 87 %

	• Body Temperature Value - 38.1 ºC • Blood Glucose Value - 115.7 mg / dl • Symptom Input - “Pain around the breast”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Breast Cancer” as Diagnosis 2. Identifies “Sadly Chemotherapy is needed. Adriamycin, Cytoxan and Taxotere will help from the doctor.” as Recommendation.
Actual Result	1. Identifies “Breast Cancer” as Diagnosis 2. Identifies “Sadly Chemotherapy is needed. Adriamycin, Cytoxan and Taxotere will help from the doctor.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Test Case ID	DIR016
Test Description	Identify Leukemia & Recommend Solution
Test Objective	1. To check if the expert system can accept monitored values and identify “Leukemia” as a diagnosis of a complete health checkup. 2. To ensure that the correct recommendation is also provide for the diagnosis.
Prerequisites	An entire health check-up is completed, and symptoms experienced by the patient have been input to the chatbot.
Test Data	• Heart Rate Value – 75 BPM • Oxygen Level Value – 87 % • Body Temperature Value - 38.1 ºC • Blood Glucose Value - 93.7 mg / dl • Symptom Input - “Bleeding gums”
Test Procedure	1. Each body vital test value is compared to check which diagnoses’ range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies “Leukemia” as Diagnosis 2. Identifies “Sadly Chemotherapy or Targeted Therapy is required. Doctors might also use immunotherapy.” as Recommendation.
Actual Result	1. Identifies “Leukemia” as Diagnosis

	2. Identifies “Sadly Chemotherapy or Targeted Therapy is required. Doctors might also use immunotherapy.” as Recommendation. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

Medical Chatbot

Test Case ID	CBT001
Test Description	Conduct Checkup
Test Objective	1. To ensure that the user can use the application to conduct a checkup and monitor body vitals. 2. To check if the chatbot functions as intended.
Prerequisites	The “MediBot” navigation button is selected to access the medical chatbot.
Test Data	• User Message – “Yes, I want to have a checkup”
Test Procedure	1. User responds to chatbot messages. 2. Messages are relayed on to the Firebase storage to connect to the IoT hardware.
Expected Result	1. The MediBot indicates to the user that the heart rate measurement can begin first. 2. First Sensor is activated.
Actual Result	The chatbot informs the user that the checkup has started by first activating the MAX30102 sensor.

Status	Pass

Test Case ID	CBT002
Test Description	Accept Message from User
Test Objective	1. To ensure that the user can message using the chatbot and respond to MediBot messages to monitor body vitals.
Prerequisites	A complete health checkup is on-going, or the user has requested to monitor a specific body vital.
Test Data	• User Message – “Yes”
Test Procedure	1. User types in message in the textbox and sends it. 2. Message is relayed on to the Firebase storage to gather a MediBot response.
Expected Result	1. Message is delivered.
Actual Result	The exact message that was typed in by the user pops up in the messaging interface to show it was delivered for processing.
Status	Pass

Test Case ID	CBT003

Test Description	Deliver Appropriate & Timely Response
Test Objective	1. To ensure that the chatbot analyses user messages correctly. 2. To check whether the chatbot messages come in chronological order, in a timely manner.
Prerequisites	A health checkup is on-going, and the user has already sent one or more messages during the process.
Test Data	• User Message – “Yes, I want to have a checkup”
Test Procedure	1. Message relayed onto the Firebase storage is analysed for key words. 2. From the key words found, the MediBot response is generated.
Expected Result	1. MediBot Message – “Alright, let us start the checkup.” 2. The message is delivered in a matter of 2-3 seconds.
Actual Result	The correct message is delivered. However, the chatbot response might take extra time to appear due to internet delay.
Status	Pass

Test Case ID	CBT004
Test Description	Activate Correct Sensor

Test Objective	1. To ensure that the chatbot analyses user messages correctly. 2. To check if the sensors can be activated from the chatbot interface.
Prerequisites	The user wants to conduct a health checkup or monitor individual body vitals.
Test Data	• User Message – “Yes, I want to have a checkup”
Test Procedure	1. User responds to chatbot messages. 2. Messages are relayed on to the Firebase storage to connect to the IoT hardware.
Expected Result	1. The MediBot indicates to the user that the specific body vital measurement can begin as it was requested. 2. The appropriate Sensor is activated.
Actual Result	The chatbot informs the user that the monitoring process can be started by using the specific sensor(s).
Status	Pass

Test Case ID	CBT005
Test Description	End Checkup
Test Objective	1. To ensure that a checkup is safely completed. 2. To check if sensors are deactivated

Prerequisites	The sensors have recorded the chose body vital or all the vitals in the case of an overall health checkup
Test Data	• Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value
Test Procedure	1. The Monitored Values are set in the according Firebase Realtime tables, fields. 2. Monitored Values from the Firebase storage is retrieved and updated into the chatbot interface and application. 3. If an overall checkup is conducted, analyse, and predict illness with recommendation after sensor(s) is / are deactivated.
Expected Result	1. Chatbot presents Monitored Values. 2. Chatbot shows that the monitoring process is completed with a diagnosis and recommendation if an overall checkup was completed.
Actual Result	The chatbot informs the user that the checkup is completed with the recently monitored values, diagnosis and recommendation popping up on the chatbot messaging interface.
Status	Pass

Test Case ID	CBT006

Test Description	Deliver any Identified Problem and its Recommended Solution
Test Objective	1. To ensure that an overall checkup is completed properly. 2. To ensure that a diagnosis and recommendation is provided after sensors have been used and monitored values are stored in the cloud.
Prerequisites	The sensors have recorded all the body vitals and sensors are deactivated.
Test Data	• Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value • Symptom Input
Test Procedure	1. Each body vital test value is compared to check which diagnoses' range they fall in. 2. The symptom inputs received from the user is analysed by the system to find matches. 3. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Identifies and presents Diagnosis with highest score. 2. Presents Recommendation corresponding to identified Diagnosis.
Actual Result	1. Identifies and presents Diagnosis with highest score.

	2. Presents Recommendation corresponding to identified Diagnosis. The information is displayed by the chatbot at the end of a health checkup
Status	Pass

5.2 Integration Testing

5.2.1 Test Plan

Test Case ID	**Importance Level**	**Test Description**
INT001	High	Conduct Overall Health Checkup
INT002	Medium	Measure Only Heart Rate
INT003	Medium	Measure Only Blood Oxygen Level
INT004	Medium	Measure Only Body Temperature
INT005	Medium	Measure Only Blood Glucose

5.2.2 Test Cases

Test Case ID	INT001
Test Description	Conduct Overall Health Checkup
Test Objective	1. To ensure that the user can use the application to conduct an overall health checkup and monitor each body vital.

	2. To check if the diagnosis and recommendation can be correctly identified.
Prerequisites	The “MediBot” navigation button is selected to access the medical chatbot and the user choose to perform a complete checkup.
Test Data	• User Message • Heart Rate Value • Oxygen Level Value • Body Temperature Value • Blood Glucose Value • Symptom Input
Test Procedure	1. Sensors are activated in order. 2. Monitored Values are passed onto the Firebase storage after user makes use of the sensors. 3. Each body vital test value is compared to check which diagnoses’ range they fall in. 4. The symptom inputs received from the user is analysed by the system to find matches. 5. System calculates each diagnosis final score to find the highest score.
Expected Result	1. Presents Monitored Values 2. Identifies and presents Diagnosis with highest score. 3. Presents Recommendation corresponding to identified Diagnosis.
Actual Result	1. Presents Monitored Values

	2. Identifies and presents Diagnosis with highest score. 3. Presents Recommendation corresponding to identified Diagnosis. The information is displayed by the chatbot at the end of the overall health checkup.
Status	Pass

Test Case ID	INT002
Test Description	Measure Only Heart Rate
Test Objective	1. To ensure that the user can use the application to conduct a checkup that only monitors the heart rate and oxygen level.
Prerequisites	1. The “MediBot” navigation button is selected to access the medical chatbot and 2. The user chooses not to perform a complete checkup and chooses only Heart Rate and O2.
Test Data	• User Message • Heart Rate Value • Symptom Input
Test Procedure	1. MAX30102 Sensor is activated. 2. Monitored Value is passed onto the Firebase storage after user makes use of the sensor.
Expected Result	1. Presents Monitored Value
Actual Result	1. Presents Monitored Value

	The information is displayed by the chatbot at the end of the Heart Rate and O2 checkup.
Status	Pass

Test Case ID	INT003
Test Description	Measure Only Blood Oxygen Level.
Test Objective	1. To ensure that the user can use the application to conduct a checkup that only monitors the heart rate and oxygen level.
Prerequisites	1. The “MediBot” navigation button is selected to access the medical chatbot. 2. The user chooses not to perform a complete checkup and chooses only Heart Rate and O2.
Test Data	• User Message • O2 Value • Symptom Input
Test Procedure	1. MAX30102 Sensor is activated. 2. Monitored Value is passed onto the Firebase storage after user makes use of the sensor.
Expected Result	1. Presents Monitored Value
Actual Result	1. Presents Monitored Value The information is displayed by the chatbot at the end of the Heart Rate and O2 checkup.

Status	Pass

Test Case ID	INT004
Test Description	Measure Only Body Temperature
Test Objective	1. To ensure that the user can use the application to conduct a checkup that only monitors the body temperature.
Prerequisites	1. The “MediBot” navigation button is selected to access the medical chatbot. 2. The user chooses not to perform a complete checkup and chooses only Body Temperature.
Test Data	• User Message • Temperature Value • Symptom Input
Test Procedure	1. MLX90614 Temperature Sensor is activated. 2. Monitored Value is passed onto the Firebase storage after user makes use of the sensor.
Expected Result	1. Presents Monitored Value
Actual Result	1. Presents Monitored Value The information is displayed by the chatbot at the end of the Body Temperature checkup.
Status	Pass

Test Case ID	INT005
Test Description	Measure Only Blood Glucose
Test Objective	1. To ensure that the user can use the application to conduct a checkup that only monitors the blood glucose.
Prerequisites	1. The “MediBot” navigation button is selected to access the medical chatbot. 2. The user chooses not to perform a complete checkup and chooses only Blood Glucose.
Test Data	• User Message • Blood Glucose Value • Symptom Input
Test Procedure	1. Blood Glucose IR Sensor is activated. 2. Monitored Value is passed onto the Firebase storage after user makes use of the sensor.
Expected Result	1. Presents Monitored Value
Actual Result	1. Presents Monitored Value The information is displayed by the chatbot at the end of the Blood Glucose checkup.
Status	Pass

5.3 User Acceptance Testing

In order to ensure that the overall healthcare system meets the requirements and expectations of its users and future patients, User Acceptance Testing (UAT) was conducted. There can be no risks taken with the health of an individual. So, it is critical that this is the final process in the testing phase.

The end users are requested to validate the software's functionality and usability. To gather valuable feedback on whether the system acts as intended and fulfils all their needs. Users can identify any discrepancies between their expectations and the actual behavior of the software and provide input on improvements and possible additional features to implement to ease the use of interface. The testing also is required to identify and uncover hidden defects and issues that may have been missed during earlier testing stages. While other testing phases, such as unit testing or system testing, focus on technical aspects and system integration, UAT focuses on gathering if the healthcare system can be deployed and gathering the satisfaction rates of the overall system.

For the proposed healthcare system, the author gathered a few test users and allowed them to test the overall hardware and software before conducting an interview to apply any necessary changes.

5.3.1 Interview Questions

Question 1: Does the healthcare application satisfy all the requirements / objectives?

Purpose: The question is asked to assess if the healthcare system aligns with all the users' needs and fulfils its intended purposes.

Question 2: Have you found any errors or problems with the overall healthcare system?

Purpose: To understand and identify any unidentified, remaining problems with the application, model, and the integration of the system components.

Question 3: Is the user interface and the IoT model easy to navigate / use?

Purpose: This question focuses on the usability aspect of the software, ensuring that users can interact with the application and model easily and efficiently. This question can also help identify any UI improvements to be made.

Question 4: Are there any functions or features that are missing, that you expect to see in the final system?

Purpose: To identify and gather possible improvements and future enhancements to the healthcare system.

Question 5: Do you think the healthcare system can perform well if implemented in a real-world scenario?

Purpose: The author asks this question to evaluate whether the test-user is satisfied with the healthcare system and believes if improved a little more can be implemented in a useful manner.

List of Interviewees

Name	**Age**	**Current Illness (If Any)**	**Description**
Mr. Dilon Fernando (baniaz56@hotmail.com)	24	Hyperthyroidism and Gout	Mr. Dilon Fernando is a student that is completing his bachelor's degree in computing. The individual was interviewed as part of the analysis and therefore was asked to test the proposed system.
Mrs. Janaki Fernando (Did not consent to give email)	55	Thyroid, High Blood Pressure	Mrs. Janaki Fernando is a stay-at-home wife that was also interviewed as part of the analysis. Thus, it seemed right for her to test the developed system.
Ms. Veena Jose (veena.jose@gmail.com)	29	Asthma	Mrs. Veena Jose is currently working as a sales manager. She agreed to test the system after communication through a common friend.
Mr. Dane Francis Dasalla (dasallafrancis@gmail.com)	21	-	Mr. Dane Francis Dasalla is a third-year student that was interviewed as part of the analysis.
Mr. Taha Mohamed Taha (taha.19@gmail.com)	20	Nasal Sinuses	Mr. Taha Mohamed Taha is a final-year student that is completing his bachelor's degree in computer science. The individual agreed to test the software when asked.

5.3.2 Answers Provided by Patients / Regular People

Question 1: Does the healthcare application satisfy all the requirements / objectives?	
Answer	
Mr. Dilon Fernando	The application satisfies all the needs and objectives described.
Mrs. Janaki Fernando	The application satisfies all the needs and objectives described.
Ms. Veena Jose	The application can monitor previous monitored data, conduct checkups, and receive predictions so it seems like it fulfils all requirements.
Mr. Dane Francis Dasalla	I think the application has almost everything you want from a monitoring and recommendation system.
Mr. Taha Mohamed Taha	All the main functions are there in the app so yes, all the objectives are satisfied.
Result Summary	The healthcare system ticks all the desired objectives discussed in the initial chapters of this project.

Question 2: Have you found any errors or problems with the overall healthcare system?	
Answer	
Mr. Dilon Fernando	The application takes can lag and break down at times but not much more.

Mrs. Janaki Fernando	When checking my body temperature, the value came quite low. It may have been because of the difficulty of placing the sensor on your head.
Ms. Veena Jose	The checkup, even though the chatbot does work it seems like incorrect inputs are not understood.
Mr. Dane Francis Dasalla	I don't see an issue in all honesty.
Mr. Taha Mohamed Taha	The application interface is simple which is good but requires some fitting to fit the dimension of every phone.
Result Summary	The testing of the application and the model have brought up some necessary faults that need to checked and corrected, before the system is finalized

Question 3: Is the user interface and the IoT model easy to navigate / use?	
Answer	
Mr. Dilon Fernando	The physical model is a bit tough to use because of the multiple wires.
Mrs. Janaki Fernando	The model is a bit congested but still works fine. The application interface layout had some oversized or undersized options, but nothing that critical.
Ms. Veena Jose	The model is very rough. It is difficult to actually check the phone app when standing in the checkup.

Mr. Dane Francis Dasalla	Yes, the user interface and IoT model were pretty easy to use. I didn't have much trouble navigating through the app. Maybe they could make it even more user-friendly, but it's decent as it is.
Mr. Taha Mohamed Taha	The interface is simple and easy to use and the model too. But when conducting the checkup, the sensors might need physical labeling.
Result Summary	The interface is accepted by almost all the testers only requiring a few minor tweaks. However, the model needs to be built in a more accessible, easier ways for the user.

Question 4: Are there any functions or features that are missing, that you expect to see in the final system?	
Answer	
Mr. Dilon Fernando	The system completes the checkup as intended but I would have more trust in the system if I could see the accuracy percentage of the results.
Mrs. Janaki Fernando	There is no way of accessing the month / year old statistical results. The application only displays the results of the past week.
Ms. Veena Jose	Since some diseases do not have recommended medications. So, maybe the addition of doctor consultations / appointment booking can improve the system.
Mr. Dane Francis Dasalla	I have an iPhone so I would like to eventually test and use the app on IOS devices.

Mr. Taha Mohamed Taha	As I have mentioned before, the interface screens need resizing to display all the buttons and components in a clear way.
Result Summary	The author recognizes the different improvements that can be made to the system and can proceed with implementing changes as well as considers the points for future enhancements.

Question 5: Do you think the healthcare system can perform well if implemented in a real-world scenario?	
Answer	
Mr. Dilon Fernando	Yes. The system can help a lot of people after the slight modifications.
Mrs. Janaki Fernando	Of course, the system works well and with a more solid version can help improve patient's lives.
Ms. Veena Jose	People nowadays can monitor their body vitals using smart watches and other devices similar. But the fact that this system has the ability to identify the issues and predict the solutions will make it unique, Yes.
Mr. Dane Francis Dasalla	A lot of people will find it very useful to have a system like this at their homes.
Mr. Taha Mohamed Taha	I believe the healthcare system has potential for real-world use. With a bit more improvement, it could be quite helpful in practical situations.
Result Summary	Even though the healthcare system has different flaws that need to be corrected, the proposed system was majorly accepted.

CHAPTER 6: SYSTEM IMPROVEMENTS & FUTURE ENHANCEMENTS

6.0 Overview

In this chapter, the author discusses the improvements that can be made to the system in the future to enhance the overall healthcare system for deployment in a real-world scenario. After taking into consideration all the responses from User Acceptability Testing (UAT), the author recognizes that even though the system fulfils most of the proposed objectives, the system can continue to be enhanced in multiple ways. Thus, this chapter will highlight some of the most significant improvements that can be implemented in the future.

6.1 Improvement to the Physical Model

One of the points that was highlighted in the UAT was how the IoT sensors were a bit uncomfortable to use. The model is very basic, with the sensors exposed in a congested manner without proper casing. The wiring is quite short and the overall model, at the moment, extremely fragile. The MLX90614 Temperature sensor in this model is also rather awkward to use. The sensor works best when it measures the body temperature at the forehead or mouth. But due to the length of the wiring, it can be difficult to get the proper value and the user might find the placement uncomfortable. A simple shift in wiring can cause the system to malfunction or produce faulty results, which is absolutely unacceptable when it comes to health.

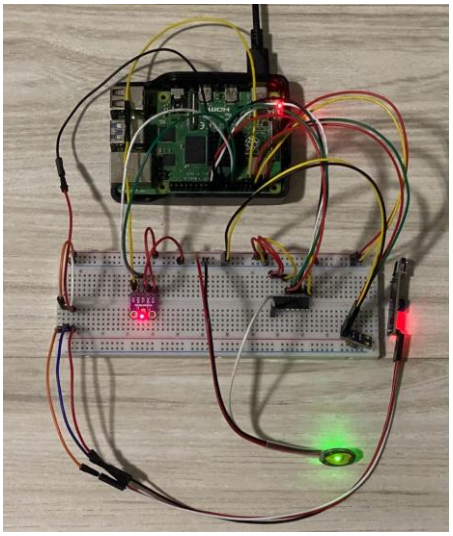


Figure 6.1 IoT Model before UAT Testing

The model needs to be developed in a better way in the future with clear casing and longer wiring. After the tester's comments, the author attempted to improve the model as much as possible such that the sensors can easily be accessible for monitoring. Figure 6.1 and Figure 6.2 shows the difference before and after the User Acceptability Testing.

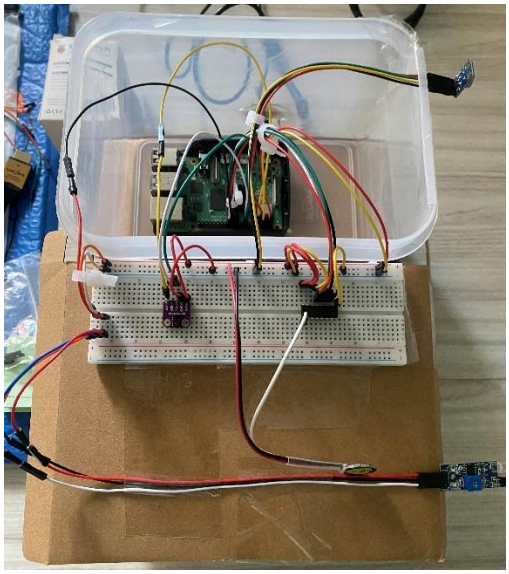


Figure 6.2 IoT Model After UAT Testing

Labels around the sensors or an instruction manual would also help massively since users do not need to know the technical side but should know when and how to use the sensors. Since we live in an ever-growing technological world, it would not be impossible to build a medical robot with an interface display for input and output to go alongside the application. The robot would have the integrated sensors and would come in a much more advanced structure with it being able to perform the same monitoring functions but in this case being able to display the monitored values, prediction, and recommendations through the display screen as displayed in Figure 6.1.

Figure 6.3 Example of a Medical Robot

Medical robots are becoming more prominent; thus, this would fit the criteria perfectly. The robots could also have integrated treatment mechanisms to treat small cuts and wounds, so the patient gets a final product that has maximum capabilities.

6.2 Cross Platform Development

The mobile application also needs continuous improvement. The interface design should be properly scaled for all different devices. As mentioned in the previous chapters, the healthcare application was proposed to be developed only on Android smartphones. However, a large majority of people do not use Android phones and instead prefer IOS devices. People that do not have Android devices would not be able to download and install the application, making it impossible to monitor, track and even conduct checkups even if they had the physical model. Therefore, it only makes sense that the application in the future is developed for IOS smartphones. In the future, it would be preferable that the application UI is built on a cross platform software such as Flutter instead of Android Studio. Using Flutter as shown in Figure 6.2., the app developed could be compatible with almost all smartphones today, allowing even more users to access this healthcare system.

Figure 6.4 Flutter App Development

6.3 Implementation of Machine Learning

The current healthcare system can identify illnesses and diseases based in a smart manner by going through a rule based expert system. The system does accurately and autonomously perform the prediction and recommendation however, they are only scored based on the monitored values and symptoms identified. To enhance the accuracy of these predictions and recommendations, it would be highly recommended that the monitored data, symptoms along with

the prediction and recommendation are pushed into a dataset that is fed into intelligent Machine Learning Models.

Models like KNN, SVM, Synthetic Networks can be implemented to classify the data, which could then be used with clustering models to identify the diagnoses and solutions more accurately and automatically. The accuracy score can also be presented to the user in the future once this is implemented, so the user can have improved confidence in the healthcare system with its predictions and recommendations.

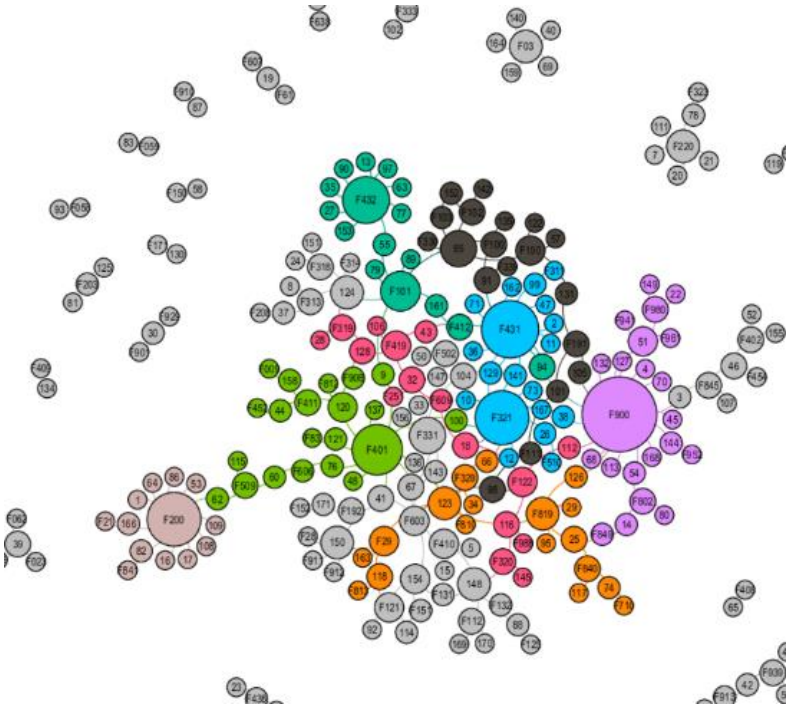


Figure 6.5 Clustering of Health Statistics

Figure 6.3 illustrates that if Machine Learning is implemented, the healthcare system can also collect and identify more diseases, illnesses, and recommendations. The current system is limited to only identify sixteen specific diagnoses. But in the future, the number of diagnoses can be upgraded by identifying patterns and communicating with professionals to produce a better system.

6.4 Implementation of Additional Smart Systems

Currently, the healthcare system only applies AI in the Chatbot of the healthcare system. The intelligent system reads the user's input and responds back to them in according to what they have written. The chatbot also utilizes AI when it retrieves the monitored body vitals and the input of symptoms experienced when an overall health checkup is completed. The chatbot feeds the data into a rule-based algorithm to calculate the diagnosis based on the highest probability score. However, as discussed in the first chapters, the implementation of AI in healthcare can be seen in various systems.

6.4.1 Medication Reminder / Acquiring System

For example, various medication reminder applications utilize their user's medical information to automatically send an email, message, notification etc. to the respective user. The systems analyze the collected data to intelligently remind the end users to take their medication at the correct times. Another form of smart system that deals with medication are the acquiring systems. There are a lot more applications that are developed that are mapped with real-time map data to know the quickest route to get medication or visit a clinic.

These features can be easily incorporated with the current healthcare application. Users will be able to conduct their overall health checkup and receive a recommended solution for if a disease is recognized. But instead of simply stopping the process, the medical chatbot can access and locate the nearest medical clinic to gather the recommended solution as soon as possible. The pricing, delivery can be automated with updates to the user profile. Thus, having a system that is more complete and get the medication delivered to them as soon as possible.

6.4.2 Doctor Consultations

In a similar manner, a second recommendation system can be coded in to recommend appropriate doctors to treat the identified illness. Not all diagnoses can be resolved by medication and require contact with medical professionals, thus mapped hospital information can be used to locate the nearest doctors for quick contact. The smart system can categorize the doctors according to their speciality and based on the identified disease, recommend a specialist for the user to contact and quickly book an appointment.

CHAPTER 7: CONCLUSION

To conclude, health plays a pivotal role in ensuring a prolonged and gratifying life. There is an escalating global focus on developing advanced healthcare systems to bolster countries' life expectancies, emphasizing early identification of diseases and their remedies, rather than solely concentrating on treatment methods. The integration of cutting-edge technologies such as Artificial Intelligence (AI) and Internet of Things (IoT) in medical applications has emerged as a key strategy to deliver efficient healthcare. This project combines both these technologies for this purpose and builds an IoT Based Personal Healthcare Monitoring System which can identify illnesses along with its recommended solutions via an AI Chatbot.

IoT facilitates the precise collection of medical data, while AI aids in comprehending and interpreting this data to devise optimal approaches without the necessity of an actual doctor. The current study proposes an innovative system through an Android healthcare application, empowering individuals to detect and identify health issues based on their real-time body vitals. The mobile app efficiently gathers and graphs monitored body parameters, and a smart AI chatbot collaborates with the IoT framework to record and transmit users' health information promptly. This information undergoes analysis by rule-based algorithms, enabling accurate and swift decisions regarding the identification and prediction of health problems along with their corresponding solutions, all in an autonomous and efficient manner.

REFERENCES

1. Abdulmalek, S. et al. (2022) “IOT-based healthcare-monitoring system towards improving quality of life: A Review,” Healthcare, 10(10), p. 1993. Available at: https://doi.org/10.3390/healthcare10101993.

2. Alfandi, O. (2022) “An intelligent IOT monitoring and prediction system for health critical conditions,” Mobile Networks and Applications, 27(3), pp. 1299–1310. Available at: https://doi.org/10.1007/s11036-021-01892-5.

3. Amin, R. et al. (2020) “IOT based medical assistant for efficient monitoring of patients in response to COVID-19,” 2020 2nd International Conference on Advanced Information and Communication Technology (ICAICT) [Preprint]. Available at: https://doi.org/10.1109/icaict51780.2020.9333448.

4. Anushree, N.R., Lakshmi, R. and Niranjan, K.R. (2021) IOT-Based Contactless Body Temperature Monitoring Using Raspberry Pi with Camera and Email Alert, 8(4), pp. 371–376. Available at: https://ijirt.org/master/publishedpaper/IJIRT152778_PAPER.pdf.

5. Basu, J. (2022) “Research on disparities in primary health care in rural versus urban areas: Select perspectives,” International Journal of Environmental Research and Public Health, 19(12), p. 7110. Available at: https://doi.org/10.3390/ijerph19127110.

6. Das, T. et al. (2021) “Sustainable development goal 3: Good health and well-being,” South-East Asia Eye Health, pp. 61–78. Available at: https://doi.org/10.1007/978-981-16-3787-2_4.

7. Ganesh, E.N. (2019) “Health monitoring system using Raspberry Pi and IOT,” Oriental journal of computer science and technology, 12(1), pp. 08–13. Available at: https://doi.org/10.13005/ojcst12.01.03.

8. Glantz, A., Örmon, K. and Sandström, B. (2019) “‘how do we use the time?’ – an observational study measuring the task time distribution of nurses in Psychiatric Care,” BMC Nursing, 18(1). Available at: https://doi.org/10.1186/s12912-019-0386-3.

9. Gsangaya, K.R. et al. (2022) “Portable IOT body temperature screening system to combat the adverse effects of COVID-19,” Journal of Sensor and Actuator Networks, 11(2), p. 22. Available at: https://doi.org/10.3390/jsan11020022.

10. Hossain, M.A. et al. (2020) “Design and implementation of an IOT based Medical Assistant Robot (AIDO-bot),” 2020 IEEE International Women in Engineering (WIE) Conference on Electrical and Computer Engineering (WIECON-ECE) [Preprint]. Available at: https://doi.org/10.1109/wiecon-ece52138.2020.9397958.

11. Humayun, M. et al. (2022) “Agent-based Medical Health Monitoring System,” Sensors, 22(8), p. 2820. Available at: https://doi.org/10.3390/s22082820.

12. Jegatheesh, C. et al. (2019) PATIENT HEALTH MONITORING WITH HEALTH ASSISTANT, 06(10), pp. 891–898. Available at: https://www.irjet.net/archives/V6/i10/IRJET-V6I10157.pdf.

13. Khan, M.M. et al. (2022) “IOT-based Health Monitoring System Development and analysis,” Security and Communication Networks, 2022, pp. 1–11. Available at: https://doi.org/10.1155/2022/9639195.

14. Küfeoğlu, S. (2022) “SDG-3 good health and well-being,” Emerging Technologies, pp. 229–253. Available at: https://doi.org/10.1007/978-3-031-07127-0_5.

15. Mahveen, A. and Patil, P.C. (2022) IoT Virtual Doctor Robot, 10(8), pp. 931–936. Available at: https://ijcrt.org/papers/IJCRT2208357.pdf.

16. Manickam, P. et al. (2022) “Artificial Intelligence (AI) and internet of medical things (IOMT) assisted biomedical systems for Intelligent Healthcare,” Biosensors, 12(8), p. 562. Available at: https://doi.org/10.3390/bios12080562.

17. Nethrasri, P., Surthi, R. and Yadav, A. (2021) Artificial Intelligence Based Healthcare Assistant and Consulting Android Application, 9(6). Available from : https://ijcrt.org/papers/IJCRT2106467.pdf

18. OECD (2021), Health at a Glance 2021: OECD Indicators, OECD Publishing, Paris, https://doi.org/10.1787/ae3016b9-en.

19. Omisola, I. (2021) What is Google Firebase and why should you use it?, MUO. Available at: https://www.makeuseof.com/what-is-google-firebase-why-use-it/ (Accessed: April 1, 2023).

20. Rahimi, A., Zakri, H. and Khalil, A. (2021) “Development of automatic reminder system for Geriatric Medicine Intake,” Malaysian Journal of Science Health & Technology, 7(1), pp. 8–14. Available at: https://doi.org/10.33102/mjosht.v7i1.152.

21. Rakshit, P., Srivastava, P.K. and Chavan, O. (2022) “IOT-based personalized health and Fitness Monitoring System,” Reinvention of Health Applications with IoT, pp. 19–30. Available at: https://doi.org/10.1201/9781003166511-2.

22. Reddy, A.B., Sai, A.K. and Sivasangari, A. (2020) “Non-invasive blood glucose level monitoring using IOT,” 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184) [Preprint]. Available at: https://doi.org/10.1109/icoei48184.2020.9142887.

23. Shinde, N.V. et al. (2021) “Healthcare chatbot system using Artificial Intelligence,” 2021 5th International Conference on Trends in Electronics and Informatics (ICOEI) [Preprint]. Available at: https://doi.org/10.1109/icoei51242.2021.9452902.

24. S Khamitkar, S. (2020) “IOT based system for Heart Rate Monitoring,” International Journal of Engineering Research and, V9(07). Available at: https://doi.org/10.17577/ijertv9is070673.

25. The Scientific World. (2019) Importance of good health in our life - how can we achieve good health and well being?, The Scientific World - Let's have a moment of science. Blogger. Available at: https://www.scientificworldinfo.com/2019/12/importance-of-good-health-in-our-life.html (Accessed: March 27, 2023).

26. Thale, S. et al. (2020) Design of Smart Medical Assistant Robot for Contactless Preliminary Health Check Up of Patients, 07(08), pp. 1741–1745. Available at: https://www.irjet.net/archives/V7/i8/IRJET-V7I8289.pdf.

27. United Nations, United Nations Sustainable Development Goals [Online] (2021), Available at: https://www.un.org/sustainabledevelopment/wp-content/uploads/2017/03/3_Why- It- Matters- 2020.pdf. Accessed 31 July 2021

28. UN Office for Outer Space Affairs, Sustainable Development Goal 3: Good Health and Well Being [Online] (2021), Available at: https://www.unoosa.org/oosa/en/ourwork/space4sdgs/sdg3.html.

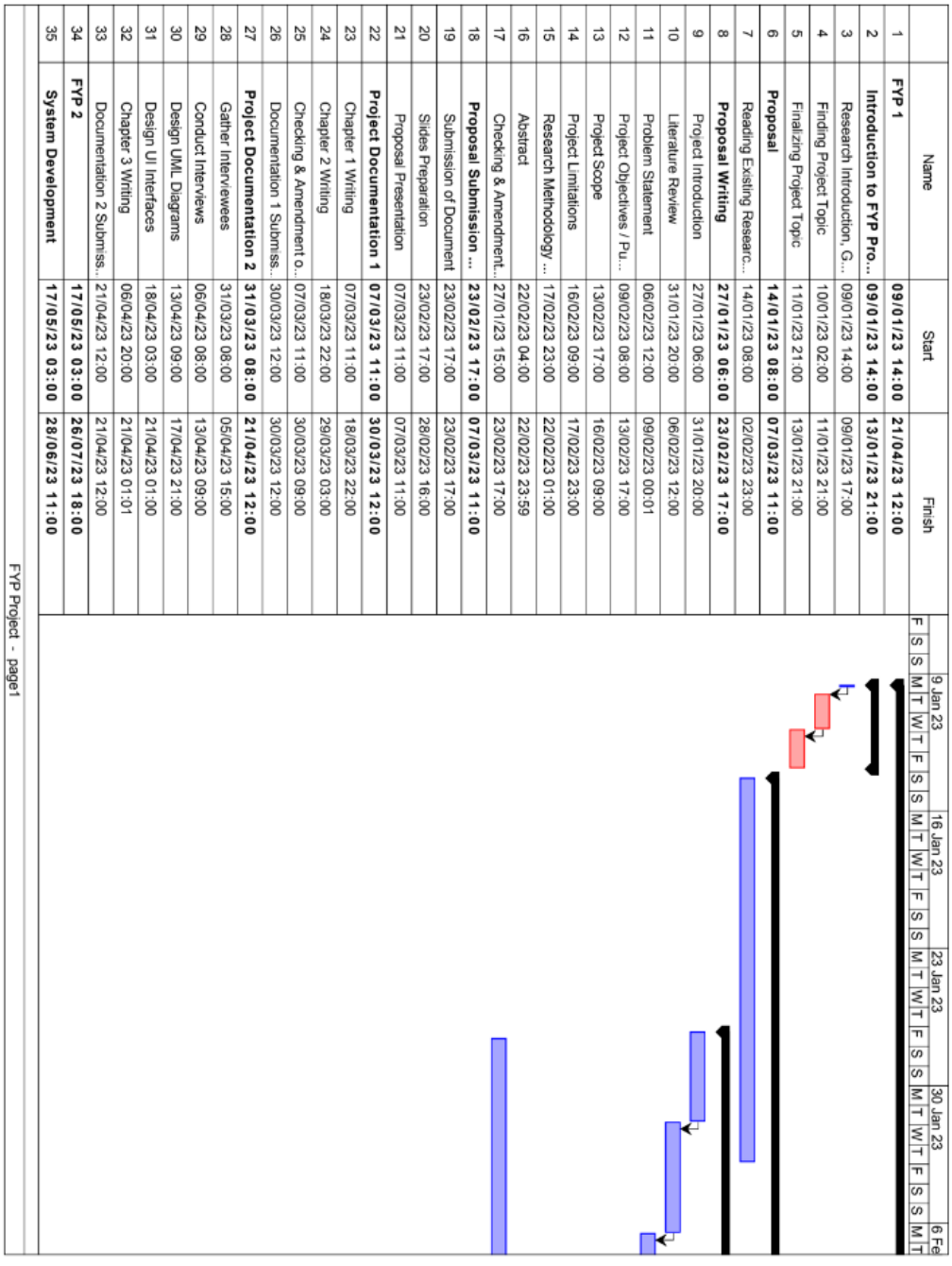
29. Uzayr, S.bin (2022) “Android Studio Basics,” Mastering Android Studio, pp. 85–132. Available at: https://doi.org/10.1201/9781003229070-3.

30. WHO (2022) World Health Statistics, World Health Organization. World Health Organization. Available at: https://www.who.int/data/gho/publications/world-health-statistics (Accessed: March 28, 2023).

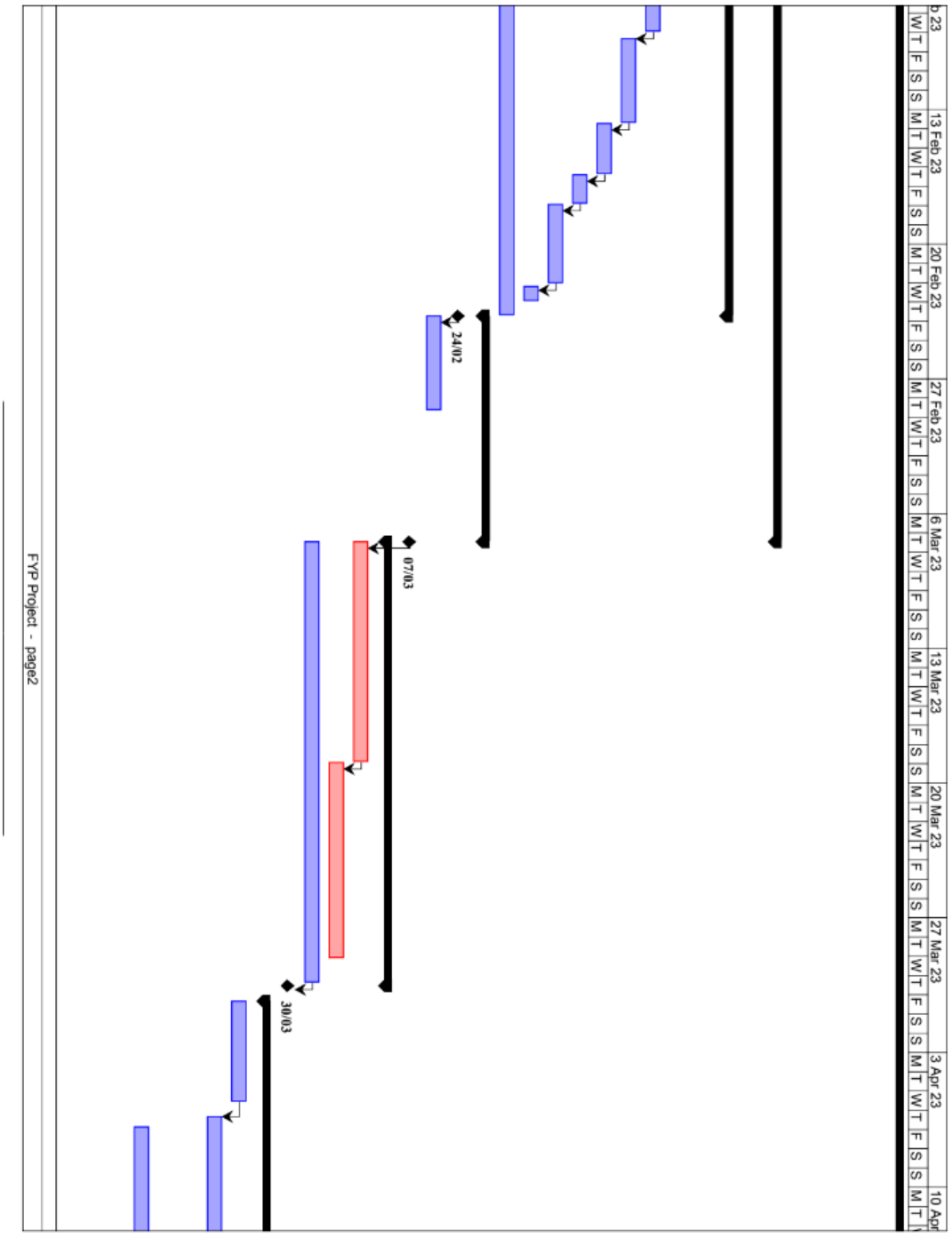
31. Xingyu, B (2023) Coronavirus (COVID-19) Visualization; Prediction, Kaggle. Available at:https://www.kaggle.com/code/therealcyberlord/coronavirus-covid-19-visualization-prediction (Accessed: April 24, 2023).

APPENDIX

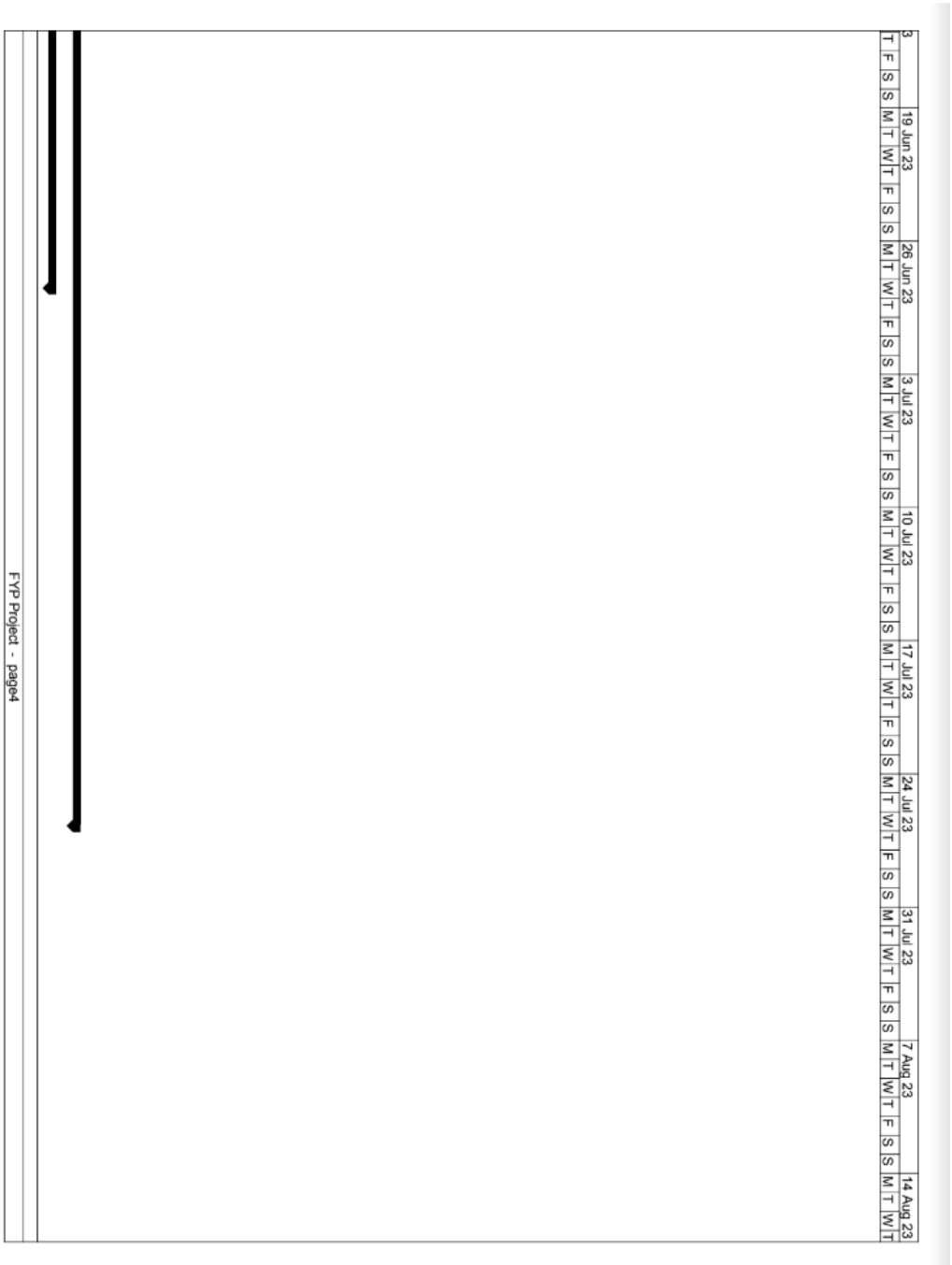
A. Project Gantt Chart



	Name	Start	Finish
1	**FYP 1**	**09/01/23 14:00**	**21/04/23 12:00**
2	**Introduction to FYP Pro...**	**09/01/23 14:00**	**13/01/23 21:00**
3	Research Introduction, G...	09/01/23 14:00	09/01/23 17:00
4	Finding Project Topic	10/01/23 02:00	11/01/23 21:00
5	Finalizing Project Topic	11/01/23 21:00	13/01/23 21:00
6	**Proposal**	**14/01/23 08:00**	**07/03/23 11:00**
7	Reading Existing Researc...	14/01/23 08:00	02/02/23 23:00
8	**Proposal Writing**	**27/01/23 06:00**	**23/02/23 17:00**
9	Project Introduction	27/01/23 06:00	31/01/23 20:00
10	Literature Review	31/01/23 20:00	06/02/23 12:00
11	Problem Statement	06/02/23 12:00	09/02/23 00:01
12	Project Objectives / Pu...	09/02/23 08:00	13/02/23 17:00
13	Project Scope	13/02/23 17:00	16/02/23 09:00
14	Project Limitations	16/02/23 09:00	17/02/23 23:00
15	Research Methodology ...	17/02/23 23:00	22/02/23 01:00
16	Abstract	22/02/23 04:00	22/02/23 23:59
17	Checking & Amendment...	27/01/23 15:00	23/02/23 17:00
18	**Proposal Submission ...**	**23/02/23 17:00**	**07/03/23 11:00**
19	Submission of Document	23/02/23 17:00	23/02/23 17:00
20	Slides Preparation	23/02/23 17:00	28/02/23 16:00
21	Proposal Presentation	07/03/23 11:00	07/03/23 11:00
22	**Project Documentation 1**	**07/03/23 11:00**	**30/03/23 12:00**
23	Chapter 1 Writing	07/03/23 11:00	18/03/23 22:00
24	Chapter 2 Writing	18/03/23 22:00	29/03/23 03:00
25	Checking & Amendment o...	07/03/23 11:00	30/03/23 09:00
26	Documentation 1 Submss...	30/03/23 12:00	30/03/23 12:00
27	**Project Documentation 2**	**31/03/23 08:00**	**21/04/23 12:00**
28	Gather Interviewees	31/03/23 08:00	05/04/23 15:00
29	Conduct Interviews	06/04/23 08:00	13/04/23 09:00
30	Design UML Diagrams	13/04/23 09:00	17/04/23 21:00
31	Design UI Interfaces	18/04/23 03:00	21/04/23 01:00
32	Chapter 3 Writing	06/04/23 20:00	21/04/23 01:01
33	Documentation 2 Submss...	21/04/23 12:00	21/04/23 12:00
34	**FYP 2**	**17/05/23 03:00**	**26/07/23 18:00**
35	**System Development**	**17/05/23 03:00**	**28/06/23 11:00**

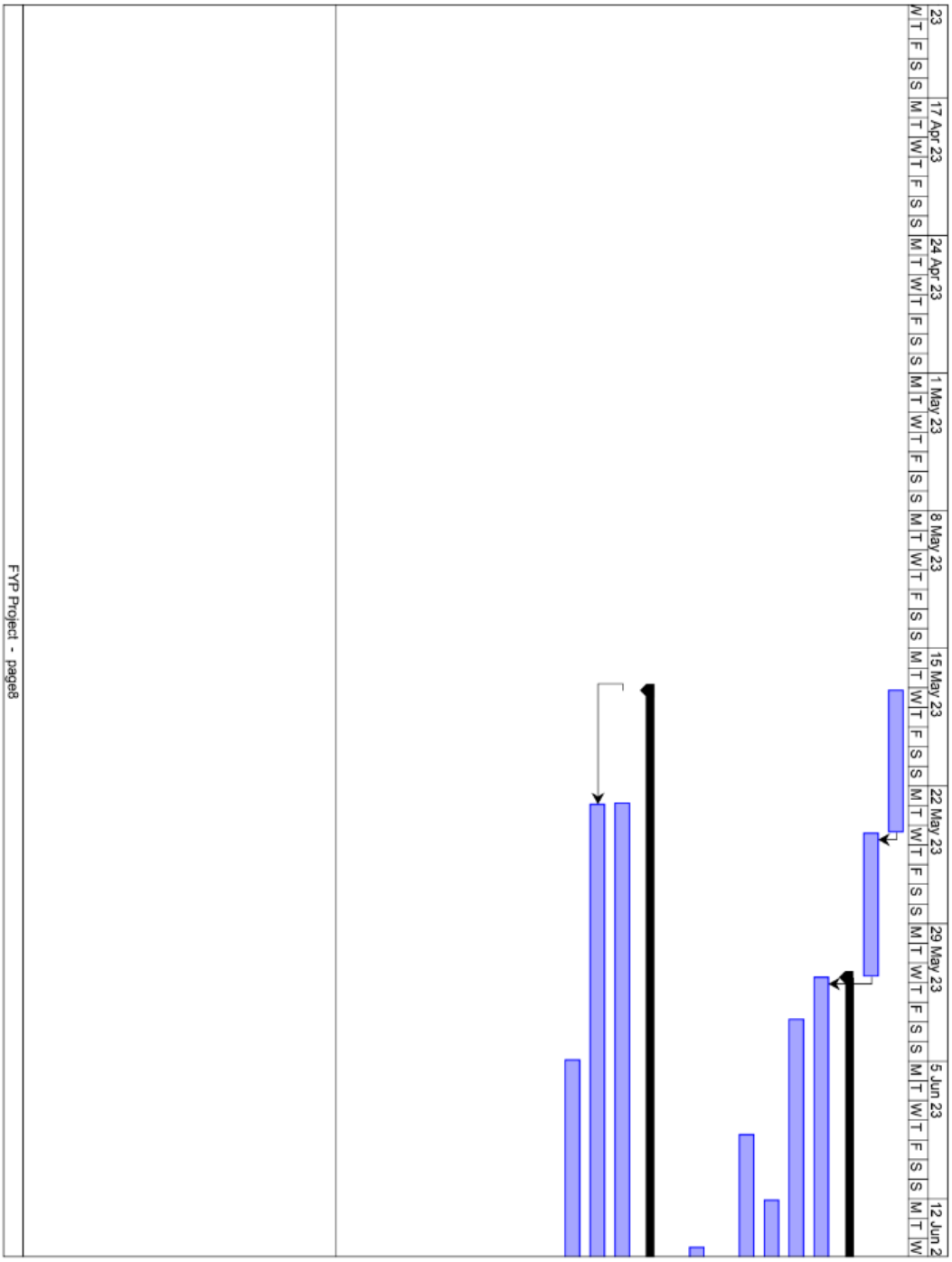

FYP Project - page2
24/02
07/03
30/03
13 Feb 23
20 Feb 23
27 Feb 23
6 Mar 23
13 Mar 23
20 Mar 23
27 Mar 23
3 Apr 23
10 Apr

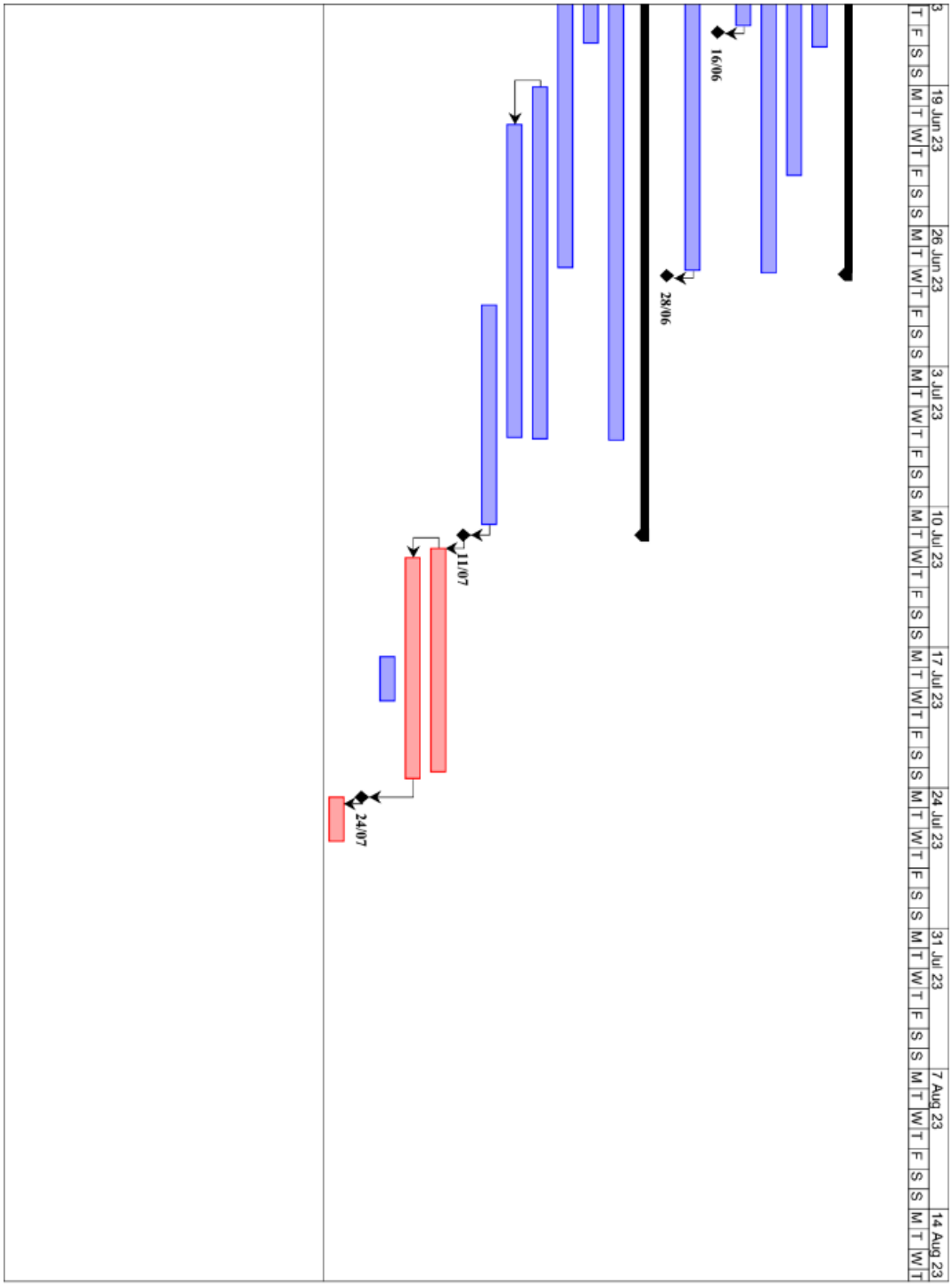

23
17 Apr 23
24 Apr 23
1 May 23
8 May 23
15 May 23
22 May 23
29 May 23
5 Jun 23
12 Jun 2
21/04
FYP Project - page3


19 Jun 23
26 Jun 23
3 Jul 23
10 Jul 23
17 Jul 23
24 Jul 23
31 Jul 23
7 Aug 23
14 Aug 23
FYP Project - page4

	Name	Start	Finish
36	Collecting Hardware Devi...	17/05/23 03:00	24/05/23 09:00
37	Construct Physical Model	24/05/23 09:00	31/05/23 18:00
38	**Programming & Coding**	**31/05/23 18:00**	**28/06/23 09:00**
39	IoT-Related Coding	31/05/23 18:00	17/06/23 03:01
40	Application Development	02/06/23 20:00	23/06/23 13:00
41	Chatbot Development	12/06/23 01:00	28/06/23 09:00
42	Edit Project Proposal	08/06/23 17:00	16/06/23 02:00
43	Edited Proposal Submission	16/06/23 08:00	16/06/23 08:00
44	Chapter 4 Writing	14/06/23 11:00	28/06/23 06:00
45	Chapter 4 Submission	28/06/23 11:00	28/06/23 11:00
46	**Testing**	**17/05/23 03:00**	**11/07/23 10:00**
47	Development Corrections	17/05/23 03:00	06/07/23 18:00
48	Unit Testing	22/05/23 23:00	16/06/23 23:00
49	Module Testing	04/06/23 23:00	28/06/23 03:00
50	Gather Testers	19/06/23 02:00	06/07/23 16:00
51	Conduct User Testing	20/06/23 23:00	06/07/23 14:00
52	Chapter 5 Writing	29/06/23 22:00	10/07/23 23:00
53	Chapter 5 Submission	11/07/23 10:00	11/07/23 10:00
54	Finalizing Last Chapters	12/07/23 01:00	23/07/23 07:00
55	Compile All Chapters	12/07/23 13:00	23/07/23 15:00
56	Pre-VIVA	17/07/23 11:00	19/07/23 17:00
57	Submission of Compiled Re...	24/07/23 11:00	24/07/23 11:00
58	VIVA	24/07/23 11:00	26/07/23 18:00

6 Feb 23
13 Feb 23
20 Feb 23
27 Feb 23
6 Mar 23
13 Mar 23
20 Mar 23
27 Mar 23
3 Apr 23
10 Apr
FYP Project - page7


FYP Project - page8
17 Apr 23
24 Apr 23
1 May 23
8 May 23
15 May 23
22 May 23
29 May 23
5 Jun 23
12 Jun 2


19 Jun 23
26 Jun 23
3 Jul 23
10 Jul 23
17 Jul 23
24 Jul 23
31 Jul 23
7 Aug 23
14 Aug 23
16/06
28/06
11/07
24/07

B. FYP Poster

Faculty of Data Science & Information Technology (FDSIT)

INTI International University

IoT Based Personal Healthcare Monitoring and Recommendation System using AI Chatbot

Problem Statement

- Patients' lack of knowledge.
- Unavailability of the doctors: precautionary measures based solely on assumptions.
- Few healthcare systems that can predict health problem and then recommend its solution autonomously.

Secure Personal Medical Data
Straightforward Chatbot
Simple Interface w/ Light Model

Internet Dependent (Time Lag)
Sensor Accuracy & Maintenance
Can Only Predict and Identify Specific Illnesses & Solutions

Acknowledgement

I would first like to thank my supervisor, Ms. Saras for guiding me through this project. I would also like to acknowledge Dr. Rajermani who kindly provided me with a laptop to complete this project as well as my friend Taha, who provided his phone for testing.

Student's Details

Name: Saurav Hazra ID: I21019956
BCSI - FYP4202
Contact Num: +60 0102453097
i21019956@student.newinti.edu.my

Project Objectives

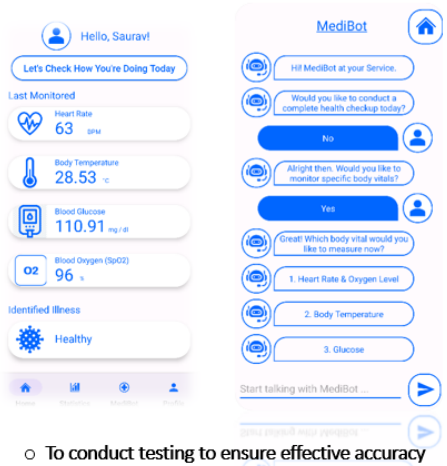
- Detailed study on the important factors and body vitals. Body vitals are the first that indicate health problems, so they are the most important factors to consider in this healthcare system.
- Research different sensors to design an IoT based monitoring system

MAX30102 Sensor
- Measures Heart Rate & Oxygen Level

MLX90614 Sensor
- Measures Body Temperature

NIR Sensor
- Measures Blood Glucose Level

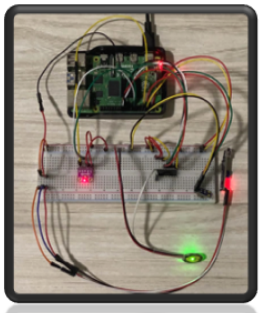
- Develop a mobile application with an AI chatbot recommendation system, that graphs recorded information, identifies health issues and provides solutions.



- To conduct testing to ensure effective accuracy
  - Testing of the Sensors
  - Application Functionality
  - Overall Usability

Project Features

Heart Rate Temp. Glucose O2



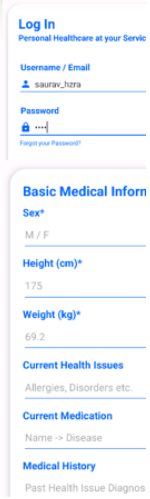
Register & Login for Personal Healthcare

Conduct Checkup of Body Vitals & Symptoms

Use IoT Model Sensors

Identify Diagnosis and Recommend Solution by referencing checkup information and medical info.

Graph & Report Each Body Vital



C. Supervisory Meeting Reports

SUPERVISORY MEETING REPORT

Meeting No. : 1 **Date:** 12/1/2023

Start Time: 2:00 PM **End Time:** 2:30 PM

Review of actions from the last supervisor meeting:-

N/A

Identification of any issues:-

- Breakdown & Explanation of Project (FYP I & II)
- Explanation of ~~Gantt Chart~~ Topic and Time Schedule

Actions set for the next meeting:-

- Evaluate Research on Topic
- Identify Changes needed
- Start Documentation of Proposal

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 2 **Date:** 28/11/2023

Start Time: 10:00AM **End Time:** 10:45 AM

Review of actions from the last supervisor meeting:-

- Proposal Research Feedback
- update on Gantt Chart
- Discussion on Proposal Writing

Identification of any issues:-

- Identify important detail to involve ~~in proposal~~
- Discussion of Proposal & Gantt Chart Submission

Actions set for the next meeting:-

- Proposal & Gantt Chart need to be submitted
- Start ~~preperation~~ preparation of proposal slides.

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 3 **Date:** 04/02/2023

Start Time: 11:00 AM **End Time:** 11:30 AM

Review of actions from the last supervisor meeting:-

- Review of submitted Gantt Chart
- Review of Proposal PPT

Identification of any issues:-

- Simply illustrate FYP I in Gantt chart, FYP II comes in later
- Have more diagrams to represent overall system.

Actions set for the next meeting:-

- Instructions for Proposal Defence
- ~~Start~~ Continue research for Chapter I documentation

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 4 **Date:** 12/02/2023

Start Time: 10:30 AM **End Time:** 10:45 AM

Review of actions from the last supervisor meeting:-

- Proposal ~~Defence~~ Defence is completed, so documentation of Chapter I begins. ~~with~~

Identification of any issues:-

- Re-phrasing of Proposal is needed for Chapter I
- Research and identify all IOT sensors needed.

Actions set for the next meeting:-

- Start Literature Review in depth and identify all software and hardware needed.

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 5 **Date:** 26/02/2023

Start Time: 12:00 PM **End Time:** 12:35 PM

Review of actions from the last supervisor meeting:-

- Evaluation of research
- Sensors like MAX30102, MLX90614, IR sensors are identified
- Raspberry Pi is agreed as IOT ~~ba~~board.

Identification of any issues:-

- Identify how system will work in terms of the software IDEs.
- How the hardware ~~will~~ and software can recommend and identify issues.

Actions set for the next meeting:-

- Continue and complete Literature Review
- Finish Chapter I

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 6 **Date:** 02/03/2023

Start Time: 12:00 PM **End Time:** 12:30 PM

Review of actions from the last supervisor meeting:-

- Complete the summary of literature review
- Finalize and submit ~~the~~ Chapter I

Identification of any issues:-

- Rule based system evaluation
- Discussion of application development over web-development.

Actions set for the next meeting:-

- Discuss the diagrams needed for Chapter 2 ~~-~~ Documentation
- System methodology & ~~Analysis~~ techniques

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 7 **Date:** 15/03/2023

Start Time: 03:00 PM **End Time:** 03:45 PM

Review of actions from the last supervisor meeting:-

- Discuss pros and cons of interviews, questionnaire, dataset analysis.
- Review of use case tables, Rich Picture, Object Oriented, Use-case Diagram.

Identification of any issues:-

- Need to find different types of patients, medical professionals etc for analysis
- Identify questions to ask.
- Data sets typically have privated information

Actions set for the next meeting:-

- Start creating application UI designs
- Document chapter 3.

Student's Name & Signature: Saurav Hazra **Date:**	Saurav Hazra
Supervisor's Name & Signature: **Date:**	

SUPERVISORY MEETING REPORT

Meeting No. : 8 **Date:** 29/03/2023

Start Time: 11:00 AM **End Time:** 11:45 AM

Review of actions from the last supervisor meeting:-

- Review list of interviewees and their answers
- Start analyzing the answers to ~~get~~ create ~~the~~ the expert system.

Identification of any issues:-

- Check credentials of the interviewees and detail their answers
- ~~Start~~ Improvement and simplification of UI designs

Actions set for the next meeting:-

- Start working on the collection of IoT sensors and equipment.
- Finish documentation of Chapter 3.

Student's Name & Signature: Saurav Hazra **Date:**	Saurav Hazra
Supervisor's Name & Signature: **Date:**	

SUPERVISORY MEETING REPORT

Meeting No.: 9 **Date:** 15/04/2023

Start Time: 02:25 PM **End Time:** 02:55 PM

Review of actions from the last supervisor meeting:-

- Check the written implementation of medical rules
- Submission of chapter 3.

Identification of any issues:-

- Development of expert system, chatbot implementation is on Python but app dev. on Android Studio
- Lack of knowledge in IOT coding

Actions set for the next meeting:-

- Code the expert system
- Begin the process of development and implementation.

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 10 **Date:** 10 ~~28~~ ~~15~~/05/2023

Start Time: 02:00 PM **End Time:** 02:30 PM

Review of actions from the last supervisor meeting:-

- Check progress of expert system dev. is complete
- Check how IOT implementation is in progress
- Identify if all sensors work.

Identification of any issues:-

- ~~Need to~~ IOT ~~and~~ system and application needs to be connected to the same network
- Have to access internet at the university and code. LAN
- Sensors need to be configured on separate buses.

Actions set for the next meeting:-

- Develop Gantt chart for complete project (FYP I & FYP II)
- Continue development

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. :	11	**Date:**	15/06/2023
Start Time:	04:00 PM	**End Time:**	05:00 PM

Review of actions from the last supervisor meeting:-

- Submit Final Gantt Chart
- Show overall progress of system

Identification of any issues:-

- Difficulty linking 2 different programming languages.
- Internet connectivity issues
- Statistical graphing needs to be finished.
- Chatbot library

Actions set for the next meeting:

- Continue Chapter 4 Documentation
- Start testing and Chapter 5 Documentation.
- Continue improving application
- Ensure everything works together

Student's Name & Signature: **Date:**	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: **Date:**	

SUPERVISORY MEETING REPORT

Meeting No. : 12 **Date:** 28/06/2023

Start Time: 02:00 PM **End Time:** 02:30 PM

Review of actions from the last supervisor meeting:-

- Submit Chapter 4
- ~~Essu~~ Test individual components of system

Identification of any issues:-

- ~~Ideti~~ Explore network data passing using Firebase
- ~~Test~~ Start User Acceptability Testing
- Heart Rate Pulse sensor is faulty
- Glucose sensor needs new equation.

Actions set for the next meeting:-

- Document test plans
- Identify problems with the model.
- Find and fix ways to repair functionality problems.

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 13 **Date:** 07 / 07 / 2023

Start Time: 01 : 30 PM **End Time:** 02 : 15 PM

Review of actions from the last supervisor meeting:-

- Development problems are repaired
- Chapter 5 Testing plans are documented for submission

Identification of any issues:-

- The names of interviewees are not yet documented
- Identify requirements of poster.

Actions set for the next meeting:-

- ~~Chapter 6 & 7 completion~~
- Start compiling complete document
- Complete Testing process

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 14 **Date:** 13/07/2023

Start Time: 12:00 PM **End Time:** 12:26 PM

Review of actions from the last supervisor meeting:-

- Testing process is completed.
- Chapter 5 submitted

Identification of any issues:-

- Explanation of requirements in Pre-Viva
- Final Poster Template Discussion
- Identify Future Enhancements & Tweaks for current system.

Actions set for the next meeting:-

- Finalize Chapter 6 & 7
- Get ready for Pre-Viva
- ~~St~~ complete the final report.

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

SUPERVISORY MEETING REPORT

Meeting No. : 15 **Date:** 20/07/2023

Start Time: 10:30 AM **End Time:** 11:15 AM

Review of actions from the last supervisor meeting:-

- Pre-Viva is complete
- Get Ready for Viva

Identification of any issues:-

- Finalize the overall system
- Build the final improved model.
- to enhance testing
- Improve Poster.

Actions set for the next meeting:-

-

Student's Name & Signature: Date:	Saurav Hazra Saurav Hazra
Supervisor's Name & Signature: Date:	

D. Turn It In Report %


I21019956_Saurav Hazra_Final Report.pdf

ORIGINALITY REPORT

11%	1%	1%	10%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS