

12-2022

Smartphone as an Edge for Context-Aware Real-Time Processing for Personal e-Health

Muhammad Bangash

Follow this and additional works at: <https://neiudc.neiu.edu/uhp-projects>



Part of the [Computer Sciences Commons](#), and the [Health Information Technology Commons](#)

SMARTPHONE AS AN EDGE FOR CONTEXT-AWARE REAL-TIME
PROCESSING FOR PERSONAL E-HEALTH

A Thesis Project Presented to
the Faculty of the University Honors Program
Northeastern Illinois University

In Partial Fulfillment of the Requirements
of the NEIU Honors Program
for Graduation with Honors

Muhammad Bangash
November 2022




HONORS SENIOR PROJECT
ACCEPTANCE AND APPROVAL FORM


Muhammad Bangash

Smartphone as an Edge for Context-Aware Real-time Processing for Personal e-Health


This senior project has been reviewed by the faculty of the NEIU Honors Program and is found to be in good order in content, style, and mechanical accuracy. It is accepted in partial fulfillment of the requirements of the NEIU Honors Program and graduation with honors.




Dr. Ahmed Khaled, Department of Computer Science
Faculty Advisor
12/1/2022
Date



Dr. Xiwei Wang, Department of Computer Science
Faculty Reader
12/2/2022
Date



Dr. Nadja Insel, Department of Earth Science
Honors Curriculum & Standards Board
12/6/2022
Date



Dr. Jon Hageman, Department of Computer Science
Coordinator, University Honors Program
6 Dec 2022
Date

ABSTRACT

The medical domain is facing an ongoing challenge of how patients can share their health information and timeline with healthcare providers. This involves secure sharing, diverse data types, and formats reported by healthcare-related devices. A multilayer framework can address these challenges in the context of the Internet of Medical Things (IoMT).

This framework utilizes smartphone sensors, external services, and medical devices that measure vital signs and communicate such real-time data with smartphones. The smartphone serves as an “edge device” to visualize, analyze, store, and report context-aware data to the cloud layer. Focusing on medical device connectivity, mobile security, data collection, and interoperability for frictionless data processing allows for building context-aware personal medical records (PMRs). These PMRs are then securely transmitted through a communication protocol, Message Queuing Telemetry Transport (MQTT), to be then utilized by authorized medical staff and healthcare institutions.

MQTT is a lightweight, intuitive, and easy-to-use messaging protocol suitable for IoMT systems. Consequently, these PMRs are to be further processed in a cloud computing platform, Amazon Web Services (AWS). Through AWS and its services, architecting a customized data pipeline from the mobile user to the cloud allows displaying of useful analytics to healthcare stakeholders, secure storage, and SMS notifications. Our results demonstrate that this framework preserves the patient’s health-related timeline and shares this information with professionals. Through a serverless Business intelligence interactive dashboard generated from AWS QuickSight, further querying and data filtering techniques are applied to the PMRs which identify key metrics and trends.

Keywords: IoMT, AWS, Android, Context-aware, Personal Medical Record, Real-time

ACKNOWLEDGEMENTS

My sincerest appreciation to Dr. Ahmed Khaled, for the continued direction throughout the timeline of this Honors Senior Project. For the invaluable feedback and comments that have contributed to a quality piece of work, thank you, Dr. Xiwei Wang and Dr. Nadja Insel.

My sincerest gratitude is extended to Dr. Jon Hageman for helping pave an educational pathway that includes participating in the University Honors Program (UHP) and presenting this work in the 2022 John Sargon Albazi Research & Creative Activities Student Symposium.

I want to acknowledge the financial support I received from UHP and the Honors Council of the Illinois Region (HCIR) through their Margaret Messer Student Research Grant. Additionally, for allowing me to participate in their Spring 2022 symposium and received an award as 2nd place from the overall 60-plus presentations across the Illinois region.

I would like to acknowledge the Society for the Advancement of Chicanos/Hispanics and Native Americans in Science (SACNAS) for accepting this work and inviting me to present at the National Diversity In 2022 STEM Conference, the nation's largest multidisciplinary and multicultural STEM diversity conference.

My deepest appreciation to Northeastern Illinois University (NEIU), a public university in Chicago, that recognizes the potential I have and continues to give students numerous opportunities to succeed both academically and professionally.

TABLE OF CONTENTS

ABSTRACT.....	iii
ACKNOWLEDGEMENTS.....	iv
TABLE OF CONTENTS.....	v
LIST OF TABLES	vi
LIST OF FIGURES	vii
LIST OF ABBREVIATIONS.....	xi
INTRODUCTION	1
LITERATURE REVIEW	7
MAIN STUDY.....	16
FUTURE WORK.....	55
SUMMARY AND CONCLUSIONS	58
BIBLIOGRAPHY.....	59

LIST OF TABLES

TABLE 1. CAPTURING REAL-TIME DATA VIA INTERNAL SENSORS.	22
TABLE 2. CAPTURING REAL-TIME DATA VIA MEDICAL SENSORS.	22
TABLE 3. CAPTURING REAL-TIME DATA VIA EXTERNAL SERVICES.	23
TABLE 4. CAPTURING REAL-TIME DATA VIA WEARABLE DEVICES.	23
TABLE 5. CAPTURING REAL-TIME DATA VIA WEARABLE DEVICES.	24
TABLE 6. WEATHER API SURVEY FOR EXTERNAL SERVICES.	37

LIST OF FIGURES

FIGURE 1. A REPORT ON THE GROWTH OF IOT SECTOR.....	3
FIGURE 2. AN EXAMPLE OF AN IOMT-BASED HEALTHCARE SYSTEM	4
FIGURE 3. PROJECT SCOPE.....	4
FIGURE 4. ELECTRONIC HEALTH RECORD ADOPTION BETWEEN 2008-2017 ..	7
FIGURE 5. GENERAL IMPLEMENTATION OF AN IOMT SYSTEM.....	10
FIGURE 6. AN EXAMPLE OF AN MQTT SYSTEM	12
FIGURE 7. MONGODB DOCUMENT STRUCTURE	14
FIGURE 8. HOW EDGE COMPUTING WORKS.....	18
FIGURE 9. IOMT FRAMEWORK, MOBILE ARCHITECTURE VERSION ONE.....	18
FIGURE 10. IOMT FRAMEWORK, MOBILE ARCHITECTURE VERSION TWO...	19
FIGURE 11. IOMT FRAMEWORK, CLOUD ARCHITECTURE.....	19
FIGURE 12. PROTOTYPE, LANDING PAGE	24
FIGURE 13. PROTOTYPE, USER REGISTRATION PAGE	24
FIGURE 14. PROTOTYPE, HOME PAGE.....	25
FIGURE 15. PROTOTYPE, INTERNAL SENSORS	25
FIGURE 16. PROTOTYPE, SCAN SENSORS.....	25
FIGURE 17. PROTOTYPE, GET USER LOCATION.....	25
FIGURE 18. PROTOTYPE, MEASURE USER PULSE	26
FIGURE 19. PROTOTYPE, EXTERNAL MEDICAL SENSORS	26
FIGURE 20. PROTOTYPE, STORE USER RECORDS.....	26
FIGURE 21. PROTOTYPE, VIEW A USER RECORD	26
FIGURE 22. PROTOTYPE, USER ANALYTICS	27

FIGURE 23. PROTOTYPE, PUSH USER RECORD TO BROKER.....	27
FIGURE 24. ENABLING MOBILE AUTHENTICATION WITH FIREBASE.....	28
FIGURE 25. USER REGISTRATION VERSION 1	28
FIGURE 26. USER REGISTRATION VERSION 2	28
FIGURE 27. USER INPUT OF BIRTHDATE	29
FIGURE 28. USER LOGIN PAGE	29
FIGURE 29. FIREBASE AUTHENTICATION AND REAL TIME DATABASE.....	29
FIGURE 30. OBSERVE AUTHENTICATED USERS OVER FIREBASE CONSOLE	29
FIGURE 31. OBSERVE DAILY USERS OVER FIREBASE CONSOLE.....	29
FIGURE 32. OBSERVE FIREBASE REAL-TIME DATABASE	30
FIGURE 33. VIEW USER PROFILE AND CURRENT RECORD.....	31
FIGURE 34. CLICK HISTORY BUTTON.....	31
FIGURE 35. VIEW HISTORY RECORDS.....	32
FIGURE 36. VIEW HISTORY	32
FIGURE 37. SMARTPHONE ENVIRONMENTAL SENSORS.....	33
FIGURE 38. TYPE TEMPERATURE SENSOR IS DEPRECATED	33
FIGURE 39. SCAN SMARTPHONE ENVIRONMENTAL SENSORS	34
FIGURE 40. RETRIEVE USER LOCATION, GPS DISABLED	35
FIGURE 41. RETRIEVE USER LOCATION, GPS ENABLED	35
FIGURE 42. CAMERA SENSOR HOMEPAGE	36
FIGURE 43. CAMERA SENSOR INTEGRATION RESULTS	36
FIGURE 44. API CLIENT REQUEST AND SERVER RESPONSE	37
FIGURE 45. RETRIEVE WEATHER INFO TEST 1	38

FIGURE 46. RETRIEVE WEATHER INFO TEST 2	38
FIGURE 47. EXTERNAL MEDICAL SENSORS BUNDLE, BITALINO	39
FIGURE 48. BITALINO ORIGINAL API	40
FIGURE 49. BITALINO ITERATION #1	40
FIGURE 50. BITALINO ITERATION #2	41
FIGURE 51. BITALINO ITERATION #3	41
FIGURE 52. BITALINO ITERATION #4	41
FIGURE 53. BITALINO ITERATION #5	41
FIGURE 54. BITALINO ECG SENSOR FINAL ITERATION	42
FIGURE 55. BITALINO PPG SENSOR FINAL ITERATION	42
FIGURE 56. PERSONAL MEDICAL RECORD JSON SCHEMA	42
FIGURE 57. PERSONAL MEDICAL RECORD JSON VALIDATION	44
FIGURE 58. FEATURES OF MODIFIED PAHO MQTT ANDROID SERVICE	45
FIGURE 59. PATIENT CONNECTED TO CLOUD	45
FIGURE 60. PATIENT PUBLISHED PMR TO CLOUD	45
FIGURE 61. AWS EC2 INSTANCE RUNNING FOR HIVEMQ BROKER	47
FIGURE 62. AWS EC2 SECURELY CONNECTS WITH A UNIQUE KEY PAIR	47
FIGURE 63. EDITING EC2 INBOUND RULES FOR INCOMING TRAFFIC	47
FIGURE 64. EDITING HIVEMQ BROKER CONFIG TO BRIDGE TO IOT CORE ...	47
FIGURE 65. AWS EC2 INSTANCE RUNNING FOR MOSQUITO BROKER	48
FIGURE 66. EDITING EC2 INBOUND RULES FOR INCOMING TRAFFIC	48
FIGURE 67. GENERATING AWS SECURITY CERTIFICATES FROM IOT CORE	49
FIGURE 68. RULES TO ROUTE TRAFFIC FROM TOPIC TO IOT ANALYTICS	49

FIGURE 69. ADDING AWS CERTIFICATES TO MOSQUITO BROKER	49
FIGURE 70. CUSTOM MOSQUITO BROKER CONFIGURATION	49
FIGURE 71. THE FOLLOWING GRAPHIC SHOWS AN OVERVIEW OF HOW YOU CAN USE AWS IOT ANALYTICS	50
FIGURE 72. CREATING A HEALTHCARE CHANNEL IN AWS IOT ANALYTICS	51
FIGURE 73. CREATING AN SQL QUERY JOB FOR THE HEALTH DATASET	51
FIGURE 74. CREATING A CRON EXPRESSION FOR THE SQL QUERY JOB	51
FIGURE 75. RESULT OVERVIEW OF RECORDS AFTER QUERY IS RUN	51
FIGURE 76. CONNECTING HEALTHCARE DATASET WITH QUICKSIGHT	52
FIGURE 77. FILTERING VALUES IN QUICKSIGHT	52
FIGURE 78. ORGANIZING DASHBOARD	53
FIGURE 79. DISPLAYING PERSONAL MEDICAL RECORDS	53
FIGURE 80. DISPLAYING PATIENT LOCATION IN A REAL-TIME MAP	53
FIGURE 81. DISPLAYING PATIENT LOCATION IN A REAL-TIME MAP	54
FIGURE 82. REAL-TIME INTERACTIVE DASHBOARD FOR HEALTHCARE STAKEHOLDERS AND PROFESSIONALS	54
FIGURE 83. ONE SINGLE CODE BASE, MULTIPLE PLATFORMS	55
FIGURE 84. EXPANDING OUTREACH BY COMMUNICATING WITH SMART WEARABLE DEVICES	56

LIST OF ABBREVIATIONS

API	Application Programming Interface
AWS	Amazon Web Services
BI	Business Intelligence
EC2	Elastic Compute Cloud
ECG	Electrocardiogram
EHRs	Electronic Health Records
GCP	Google Cloud Platform
GDP	Gross Domestic Product
GPS	Global Positioning System
ICT	Information and Communication Technology
IoMT	Internet of Medical Things
IoT	Internet of Things
MQTT	Message Queuing Telemetry Transport
PMR	Personal Medical Record
PPG	Photoplethysmography
QoS	Quality of Service
SNS	Simple Notification Service
SMS	Short Message Service
SDGs	Sustainable Development Goals
UN	United Nations
UUID	Universally Unique Identifier
WHO	World Health Organization

INTRODUCTION

The United Nations (UN) assembled in 2015 to introduce globally the 17 Sustainable Development Goals (SDGs) and believes by bringing awareness to these inclusive goals, will allow communities across the world to put forward collective effort in mitigating problems such as improving infrastructure, poverty, hunger, and unhealthy lifestyles. Each of these goals has a role to play, significance, and an impact. The UN has set the deadline to reach them by 2030. Under the umbrella of SDGs, the project is in alignment with the vision of these goals by providing the groundwork capabilities to improve infrastructure for personalized healthcare.

Long-term planning is essential for organizations to keep their mission afloat. Strategic decisions by stakeholders ensure employees continue being employed and the organization reaches quarterly initiatives. When organizations create quarterly roadmaps, data-driven decisions can play a critical role in these high-stake meetings. In addition to data, rapport-building with clients and customers is one of the approaches businesses and corporations utilize to increase trustworthiness and create profitable relationships. The Four Ps', Patients, Providers, Payors, and Policymakers are the key stakeholders who shape the direction of healthcare institutions.

One of the strategic goals that can benefit these stakeholders is frameworks for increasing meaningful relationships between caregiver and their patients. Unlike non-essential organizations, the institutions that make up the medical domain are vital for keeping the population healthy, which results in a more productive economy.

As the COVID-19 pandemic ravaged communities, we witnessed the ripple effects it had. For example, medical institutions were heavily strained as COVID-19

cases doubled, tripled, quadrupled, and exponentially increased in a gradual manner. The precedence in checkups was given to patients diagnosed or having pre-COVID-19 symptoms. However, the patients who needed general checkups for their vital information (i.e., pulse and oxygen levels) were not given the same precedence. These issues have called for an acceleration to integrate cloud services, real-time mobile applications, and new frameworks for personalized healthcare.

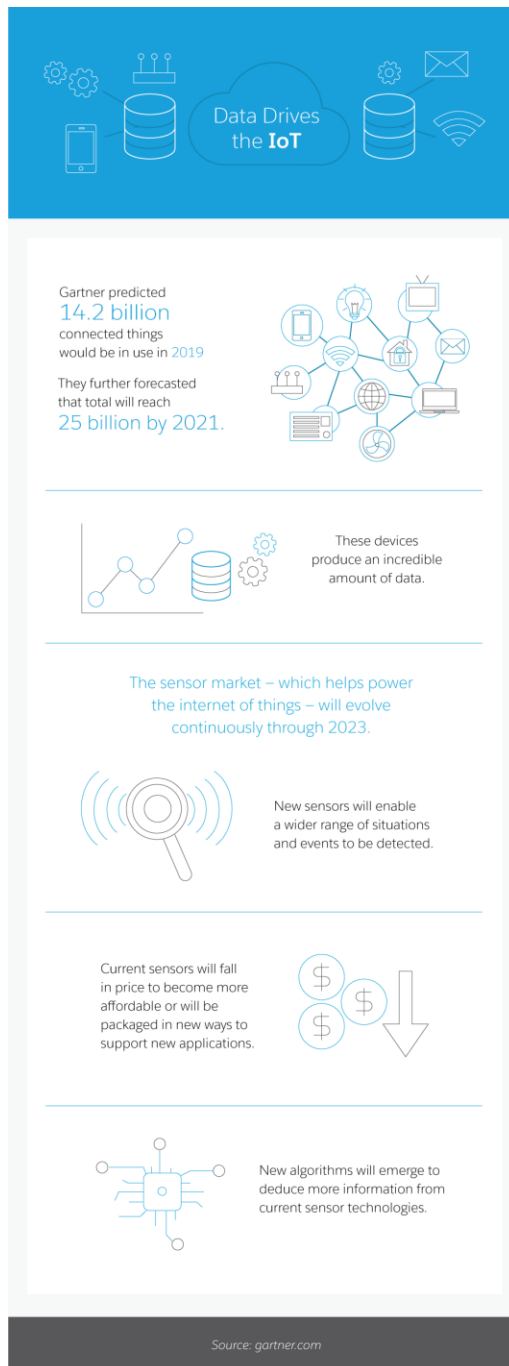
In 2015, according to the Economic Strategy Institute, a nonprofit that tracks the US market, reports industry spending in the cloud services sector continues to increase. At the current time, the economy will add three trillion in the gross domestic product (GDP), which will create eight million new jobs by 2025. Cloud services include data analysis, the Internet of Things (IoT), and cloud computing. The properties of IoT are as follows: they need a unique identity, the ability to capture data, the ability to be accessible and programmable by users, and most importantly, the ability to communicate with other devices. A “Thing” can be any device that provides service, offers functionality, and forms the network of IoT. A “Thing” can offer more than one service!

“The internet will disappear. There will be so many...devices, sensors, things that you are wearing, things that you are interacting with, that you won’t even sense it. It will be part of your presence all the time.”

-Eric Schmidt, Google Chairman

A Gartner report explains that the number of connected Things will reach 14.2 billion in 2019 and top 25 billion by 2021 (Fig. 1). In the healthcare industry, these internet-connected devices and applications play a vital role for medical personnel such as remote health monitoring and smart alert systems.

Figure 1. A report on the growth of IoT sector



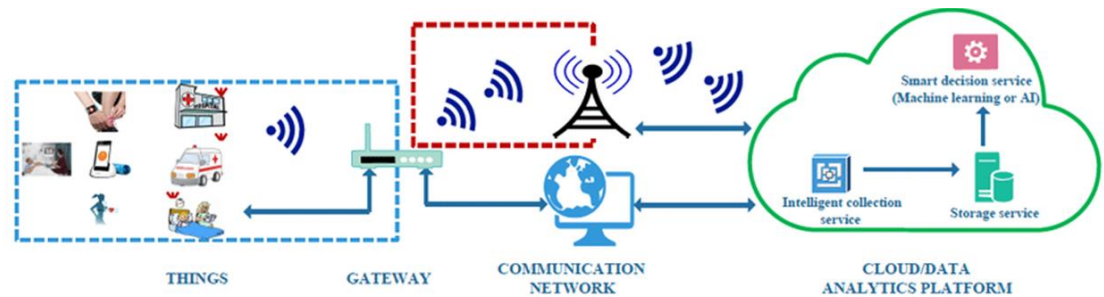
Note. Retrieved from Salesforce. *The Internet of Things Connects Customers to Their World*. Accessed 10 Nov. 2021.

In the healthcare industry, the collection and storage of personal medical data are becoming increasingly dependent on the use of cloud computing and secure communication. Information and Communication Technology (ICT) allows the

healthcare industry to access, store, modify, and relay information. For example, “identifying health symptoms, locating health-care providers, and storing personal health information is now (*sic*) more common among the general population and is critical for improving quality of life for many individuals” (Harrington et al., 2020). These healthcare services are also called mHealth or eHealth.

Under the umbrella of IoT is the Internet of Medical Things (IoMT). In the healthcare industry connected devices play a big role for medical personnel such as tracking patient blood sugar levels through remote health monitoring and smart alert systems for notifying professionals. Figure 2 illustrates an example of the typical IoMT architecture of a healthcare system.

Figure 2. An example of an IoMT-based healthcare system

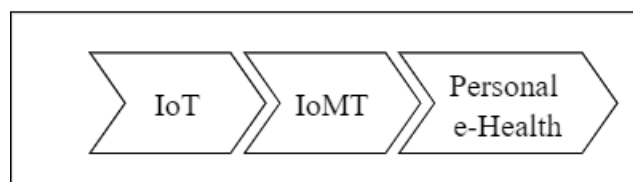


Note. Retrieved from (Alam et al., 2018). *A Survey on the Roles of Communication Technologies in IoT Based Personalized Healthcare Applications*. Accessed 15 Nov. 2021.

Project Scope and Proposal

The scope of this project falls under the umbrella of the Internet of Things as the Internet of Medical Things with a focus on *personal e-Health*, as illustrated in Figure 3.

Figure 3. Project scope



The medical domain is facing ongoing challenges such as retrieving a clear picture of the patient, essential health timeline is lost, real-time capabilities, and implementing security. The traditional medical record includes information such as the patient's name, heart rate, blood pressure, and electrocardiogram (ECG). For a doctor, such a record does not give a full picture of the medical status and the patient's living environment. Therefore, in this project, we will create context-aware personal medical records (PMRs) that capture the user environment such as the current date, city, state, and weather.

A multilayer framework can address these healthcare challenges through a physical layer, an edge layer, and a cloud layer. The physical layer consists of the hardware utilized such as medical sensors. The edge layer consists of a mobile device being used to communicate with the physical layer. The cloud layer consists of utilizing multiple cloud platforms to give the framework user security, real-time cloud capabilities, an MQTT broker, a gateway to cloud services, and an interactive dashboard for healthcare stakeholders and professionals.

In this project, the focus is on: 1) device connectivity, where we connect smartphone sensors and medical Bluetooth sensors to smartphones; 2) data collection, where medical sensors and services collect and report to smartphones; 3) interoperability, where the smartphone performs local real-time processing and protocol-independent data for friction-less data processing and analysis (through a communication broker and the internal database); 4) Building context-aware personal medical records or PMRs, to be securely stored and further processed by a cloud platform and utilized by authorized users, medical staff, applications, and medical institutions.

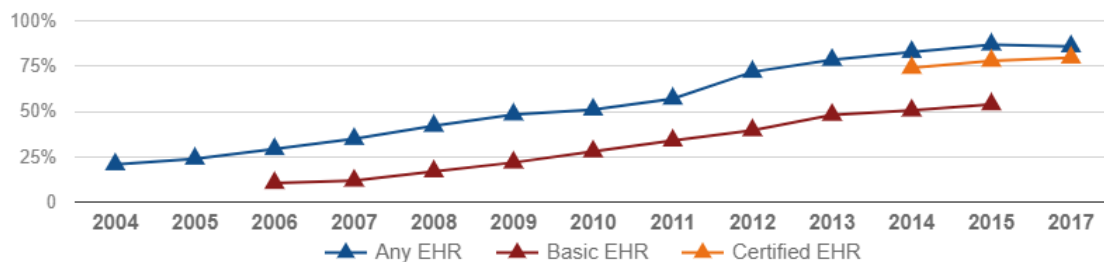
In this project, we are building real-time context-aware PMRs through mobile security services from Google Cloud, internal sensors of a smartphone, two external Bluetooth medical sensors, external services (weather, humidity), and a local database that reports such collected data to an Android-based smartphone. We are utilizing the MQTT protocol and a customized bridge between the broker and gateway that will allow the patient to share their PMRs securely with the second cloud provider. These PMRs will then be processed by a cloud platform (AWS) to be further queried, filtered, and visualized in an interactive real-time dashboard for authorized medical staff and institutions. Furthermore, the cloud providers have built-in firewall protections and follow popular compliances which will offer this IoMT framework, a multi-layer security protection.

The thesis is structured as follows. Chapter 1 (Literature Review) will be on healthcare challenges and related IoMT frameworks. Chapter 2 (Main Study) will address the multilayer framework more deeply, implementation challenges, findings, and results. Chapter 3 (Future Work) will address a direction to expand and improve the implemented framework. Chapter 4 (Summary and Conclusions) will be a summary of the thesis.

LITERATURE REVIEW

To create a context-aware medical record, we need to first understand the history and challenges in the field of healthcare. The translations of Egyptian medical papyri writings (1,600-3,000 BC) demonstrate the usage of writing a patient's medical history, or as we call it today a medical record (Evans, 2016). Millennia later, in February 2009, President Barack Obama signed the American Recovery and Reinvestment Act, which included shifting medical records from the traditional file-based system to a more modern digitized system of electronic health records (EHRs). A decade later, in 2019, the Office of the National Coordinator for Health Information, which oversees the development of national health technology infrastructure and is part of the U.S. Department of Health & Human Services, reported “EHRs (*sic*) adoption has more than doubled since 2008 from 42% to 86%” (Fig. 4).

Figure 4. Electronic health record adoption between 2008-2017



Note. Retrieved from Office of the National Coordinator for Health Information. Accessed 20 Nov. 2021.

However, new issues regarding EHRs have emerged. As the COVID-19 pandemic emerged worldwide, the traffic of patients increased, and a report by STAT, a trusted and authoritative journal about health, medicine, and the life sciences.

“Because of the way most modern electronic health record systems are built, it can take a clinician a long time to get a clear picture of the patient in front of him

or her. That's because a patient's electronic health record is split into many tabs. Some information is under the problem list, some under medications, some under imaging, and so on. The essential timeline of health data is lost. This may mask underlying vulnerabilities because it is difficult to reassemble a patient's data into a cohesive narrative, causing an incorrect view of the patient's risk for Covid-19" (Perakslis & Huang, 2020).

A recent study published in the Journal of Medical Internet Research, states "The EHR's function will be more optimal if patients can share their health data with healthcare providers. Personal health records can help patients share their data with healthcare providers and provide useful information during health emergencies" (Harrington et al., 2020). In this project we will focus on addressing these problems and issues by creating an IoMT framework such as an implementation of a cloud dashboard which can be accessible through multiple healthcare providers. Additionally, a set of features that demonstrate[s] real-time processing and a complex data flow among an interface, cloud platforms, Bluetooth medical sensors and a smartphone device.

Related work

This section will address related studies that help support the proposed IoMT framework. In "Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned," a conference paper explains a sensor framework that shows different ways of mobile business application development with the integration of external sensors (Schobel et al., 2013). A part of the paper consists of developing a mobile fitness application "XFitXtreme" and using external sensors to provide relevant data such as vital signs for athletes during their workout sessions. The

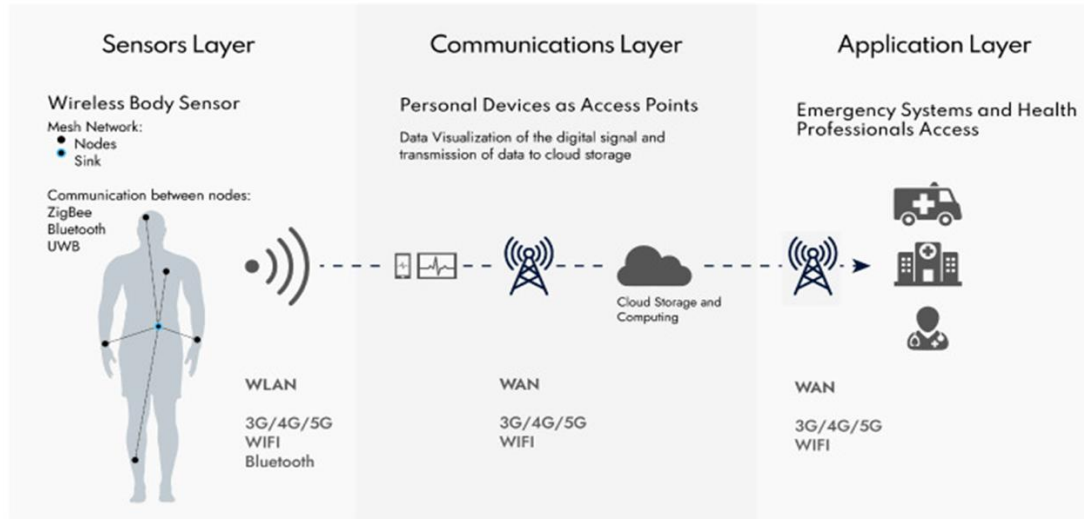
authors mentioned there is an increasing trend of external sensors being used for data collection in the field of healthcare. In the field of healthcare, these external sensors can be utilized to measure a patient's data such as their vital signs (pulse and oxygen levels). The study explains mobile applications on smartphones and tablets are continuing to play a role in the medical domain. Additionally, research on personalized healthcare and IoMT frameworks which utilize smartphones as an edge layer continues to hold value and popularity in the medical domain.

The XFitExtreme is based on an Android framework. The application establishes and communicates with external sensors using Bluetooth (2.4GHz frequency). The application uses the SQLite database to store user data locally such as recorded vital signs. The first external sensor tested was the Polar WearLink+ Heart Rate Monitor. The second external sensor tested was the MedChoice Oximeter MD300C318T, providing both heart rate and oxygen saturation values. In this study, a limitation was found that there can be an interruption in the connection between the external sensors and the application (Schobel et al., 2013). It is important to note that in the medical domain stable connectivity holds significance and importance as professionals are dependent on accurate data for a proper diagnosis.

In an article published in the Journal of Medical Internet Research, which includes a section on an IoMT system for patient monitoring, the authors state, "Over the last years, different types of IoMT architectures were proposed and investigated. These are generally multilayer systems that intend to assure safe data transmission and communication. Recent implementations generally propose three main layers, a wearable sensing layer, an intermediary data acquisition and transmission layer and a cloud

computing layer” (Silva & Tavakoli, 2020). Similarly, this project framework is based on a multi-layer system (Fig. 5).

Figure 5. General implementation of an IoMT system



Note. Retrieved from Journal of Medical Internet Research. *Domiciliary Hospitalization through Wearable Biomonitoring Patches: Recent Advances, Technical Challenges, and the Relation to Covid-19*. Accessed 20 Dec. 2021.

Previously implemented middleware telemonitoring healthcare system monitors, such as Raspberry Pi boards, have been used to collect body sensors' data for patients' healthcare parameters. However, the Raspberry Pi board displays patient information on a webpage supported by HTTP, where patients and doctors can communicate without a physical presence (Verma et al., 2022). So far, physicians and family members can access patients' health parameters using the body wireless sensor network (BWSN). Other developments, such as the Ciphertext-policy-attribute-based encryption, have been utilized with the webserver to alarm ambulances when the patient's health is at risk (Tsao et al., 2022). A significant challenge with these developments is that it is difficult to equip the elderly with the growing new technologies since they are limited from reality with smartphones and computing. Smart healthcare based on IoT reduces the complexity

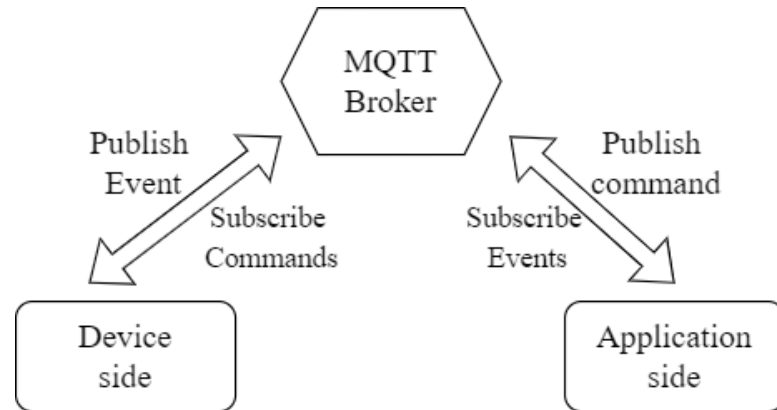
and complications by ensuring intelligent data integration and maintaining such information through cloud service. The MQTT protocol has its significance in monitoring people at risk. It ensures patients and physicians to act in time for disease prevention and early professional intervention in case of a medical emergency.

Real-Time Health Monitoring with MQTT

This section will address MQTT systems serving as healthcare middleware. Healthcare challenges such as speed, price, and complexity will ease over time with the implementation of systems supported by the IoT. There is numerous open and standard middleware, such as MQTT based on IoT, which is easily deployable in healthcare systems. MQTT protocol services are open and standard middleware for outer connections supported by IoT technology. For computer science professionals, other popular middleware outer connections that are easily deployable include representational state transfer (REST), API, data distribution service (DDS), advanced message queuing protocol (AMQP), extensible messaging and presence protocol (XMPP), java message service (JMS), and simple object access protocol (SOAP) (IJCSMC Journal & Hasan, 2018). MQTT protocols have high ability and low power consumption, making them the best choice for IoT networks (Masykur et al., 2020). It is a lightweight messaging protocol that enhances the transformation and distribution of data. It is distinct from the HTTP report/request model since it transmits data from one machine to another (as illustrated below in Figure 6). It can also reduce network bandwidth issues. Thus, making it highly reliable even when using a reliable connection with the assurance of message delivery even with disconnection. MQTT has an information-flow approach that simplifies data sensing, analysis, visualization, and user profile. IBM created the MQTT

protocol in the 1990s, which is an advanced and convenient technology for healthcare systems. With limited technology in the 1990s, IBM implemented the asynchronous messaging protocol connecting message senders and receivers in space and time (Tsao et al., 2022).

Figure 6. An example of an MQTT system



Note. Retrieved from (Saminathan & Geetha, 2018). *Real Time Health Monitoring System Using IoT*.

Accessed 20 May, 2022.

A significant aspect of MQTT is the scalable and reliable network environments with the global guarantee of sending and receiving simple commands in different quality of service (QoS). Technology advancement requires MQTT to undergo modification for its future ability.

Relational Database

This section will address the historical advancement of the database. In previous decades, when databases did not exist, all the information was documented on paper and stored in archives. Organizations would use physical files for a product, employee, or client. This was a manual file system that performed well if the content was small but not applicable for scaled applications. Commercial organizations faced the issue of easy accessibility and other limitations such as loss of records which continued until the birth

of the relational database. In 1970, a scientist from the International Business Machines Research Laboratory, Edgar Frank "Ted", released a paper “A Relational Model of Data for Large Shared Banks” (2001). This paper “introduced a new way to model data. It elaborated a way of building a bunch of cross-linked tables that would allow you to store any piece of data just once” (Kelly, 2020). Nine years later, in 1979, Oracle, a technology corporation, pushed to the market a commercial relational database (Kelly, 2020). Further advancements were made to the relational database.

NoSQL and Cloud Computing

This section will address the historical advancement of relational databases and the era of cloud computing platforms. In the late 2000s, as the prices of computer storage declined, NoSQL databases were introduced. Some examples of NoSQL databases were Google BigTable and Cassandra. The earlier relational databases were designed to compute and store onto a single system (server). As internet usage increased, there was a need for a more scalable database across multiple servers. Multiple servers connected to the internet are called the cloud.

The concept of cloud computing means a group of thousands of computers connected by a network, providing users with cloud applications and storage. The current popular cloud providers are Amazon Web Services (AWS), Azure from Microsoft, and Google Cloud Platform (GCP). The goal of these cloud providers is to allow small businesses or big organizations to grow and succeed in real-world usages such as your local shop or Netflix streaming. Further, if your local shop has forty customers daily, only that amount of cloud storage will be used. However, if the local shop grows to two hundred customers daily, the owner does not need to worry about a storage deficiency,

the cloud will scale as appropriate. NoSQL databases over the cloud are now used across the healthcare and financial industries to store sensitive data.

MongoDB and DynamoDB

This section will address two current popular NoSQL databases. NoSQL databases have garnered attention and have become popular. There are two market leading NoSQL databases; the first being MongoDB, a document-oriented database program, in which data is distributed effectively and scaled using either of the cloud providers. In MongoDB, you also do not need to join the data as a table as compared to the traditional relational database. MongoDB stores data in documents which results in faster queries. “No, Clippy, I’m not talking about Microsoft Word Documents. I’m talking about BSON documents. Figure 7 illustrates that “BSON is a binary representation of JSON (JavaScript Object Notation) documents” (Schaefer, 2020). In contrast, JSON depicts organized data in a way that is easily compatible with the conceptual world that most developers inhabit. The JSON format is used for serializing and transmitting structured data over a network connection. It is primarily used to transmit data between a server and web applications. Web services and APIs use JSON format to provide public data. JSON can be used with modern programming languages.

Figure 7. MongoDB document structure

```
{  
  name: "sue",  
  age: 26,  
  status: "A",  
  groups: [ "news", "sports" ]  
}
```



The diagram shows four horizontal arrows pointing from the right towards the corresponding lines of the JSON document. Each arrow is labeled 'field: value' in blue text. The arrows point to the lines: 'name: "sue"', 'age: 26,', 'status: "A"', and 'groups: ["news", "sports"]'.

Note. Retrieved from MongoDB. *Documents*. Accessed 10 Nov. 2021.

The second NoSQL database is DynamoDB, which is limited to AWS. One of the key differences between these two is that DynamoDB's infrastructure scalability is fully managed by AWS and can be available with a few clicks while MongoDB requires manual configuration (Wickramasinghe, 2021). As the choice of cloud provider is AWS, we choose DynamoDB for the proposed framework.

MAIN STUDY

Context-aware Healthcare System

This section addresses the proposed IoMT framework and its components. The goal of the project is to design, implement, and test a framework consisting of three layers for an edge-based IoMT context-aware healthcare system that demonstrates a set of mobile features with real-time processing capabilities. Figures 8-11 illustrates the project's tools, components, and three layers:

The first layer, the physical layer, captures vital signs and information about the user's environment before transmitting such collected data to the second layer. The second layer, the edge layer, provides real-time processing with edge devices. The second layer consists of a smartphone being utilized as an "edge computing device" to visualize, analyze, and store medical records in a local database and then report these stored medical records by first going through a secure middleware - an MQTT broker. The mobile application for personal healthcare is developed for an Android ecosystem.

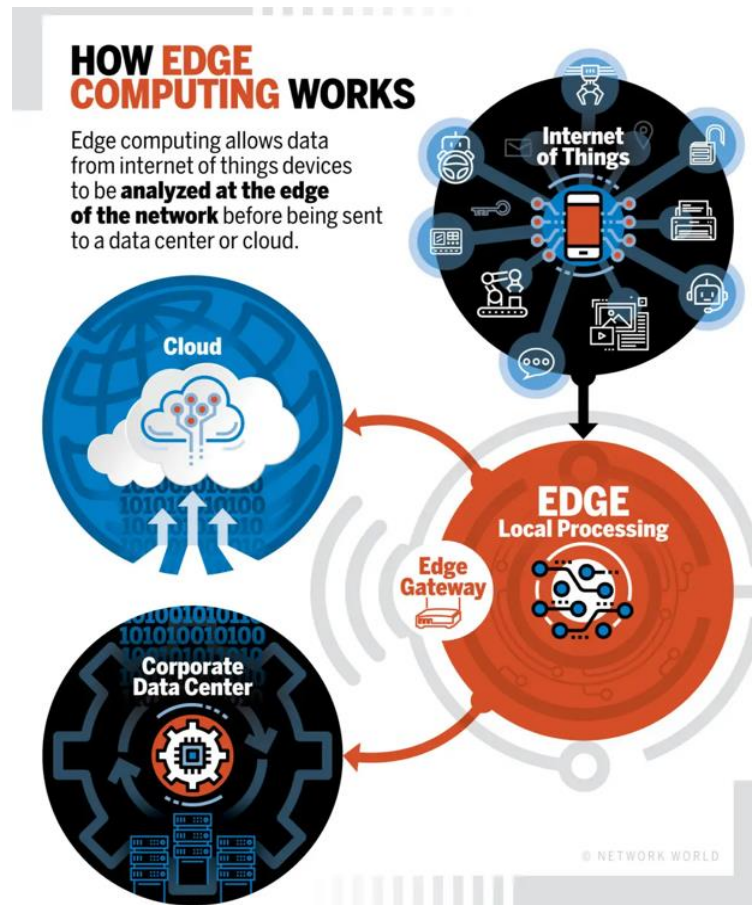
For Android developers, the popular way of programming applications is with languages such as Flutter, Kotlin, and Java. This project is written in Java. For mobile security, user registration, and data synchronization, Firebase Authentication and Real Time Database from Google Cloud were utilized. For retrieving the current location of the mobile user, we utilized Geocoder, an API service that takes advantage of internal GPS sensors. For retrieving the weather, there is a survey done in the External Weather Services section for the different APIs (Bush, 2019). We utilized Openweathermap API. There were two external medical sensors utilized, ECG and PPG, purchased from a company called BITalino. We utilized their provided API to communicate with these

sensors and further optimized it by visualizing the retrieved data frames through a library called Androidplot. For the local database in the Android application, there were two options either Room or SQLite. We utilized SQLite.

The MQTT broker allows for further communication with the third layer that hosts cloud services for further data processing. The third layer, the cloud layer, utilizes a gateway to bridge the medical information that is published to the broker to the on-cloud microservices. For the MQTT broker library, the middleware, we utilized and optimized the open-source Mosquito library from Eclipse to bridge communication with AWS IoT Core. For configuring the Mosquito library and hosting it as the MQTT broker, AWS Elastic Compute Cloud (EC2) was utilized. With EC2 a cloud developer can setup up an instance or server. For communicating from the Android device to the MQTT broker, a library called Paho MQTT Android Service was utilized. AWS IoT Core was utilized to generate security certificates and create IoT rules which serve as the gateway from the broker to the cloud services.

AWS IoT Analytics, a cloud service, was utilized to create a healthcare dataset from the retrieved records and to further process them by querying. A daily scheduler (cron) is applied over AWS to query records every minute. AWS QuickSight, a cloud service, was utilized to interact with the healthcare dataset and to publish a real-time interactive dashboard. Chapter 4 (Future Work) will also address SMS notifications for important medical alerts through AWS SNS. Additionally, a secure cloud database through AWS DynamoDB can be utilized to develop a medical portal.

Figure 8. How edge computing works



Note. Retrieved from (Gold & Shaw, 2022). *What is edge computing and why does it matter?* Accessed 10 Nov. 2021.

Figure 9. IoMT framework, mobile architecture version one

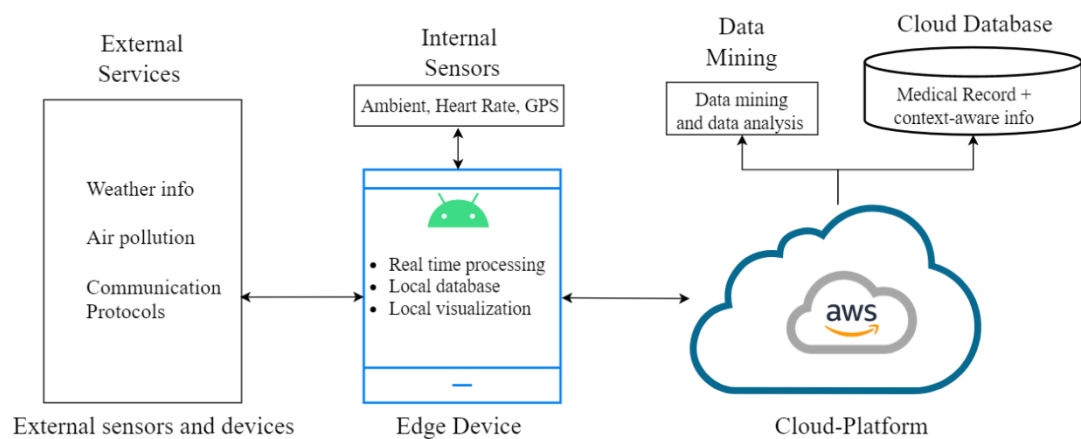


Figure 10. IoMT framework, mobile architecture version two

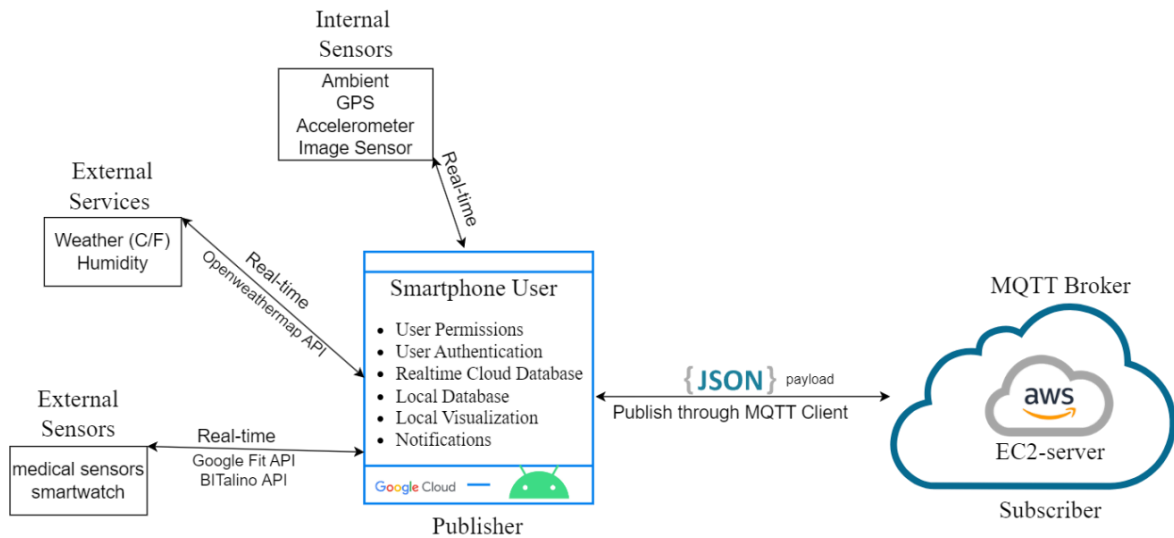
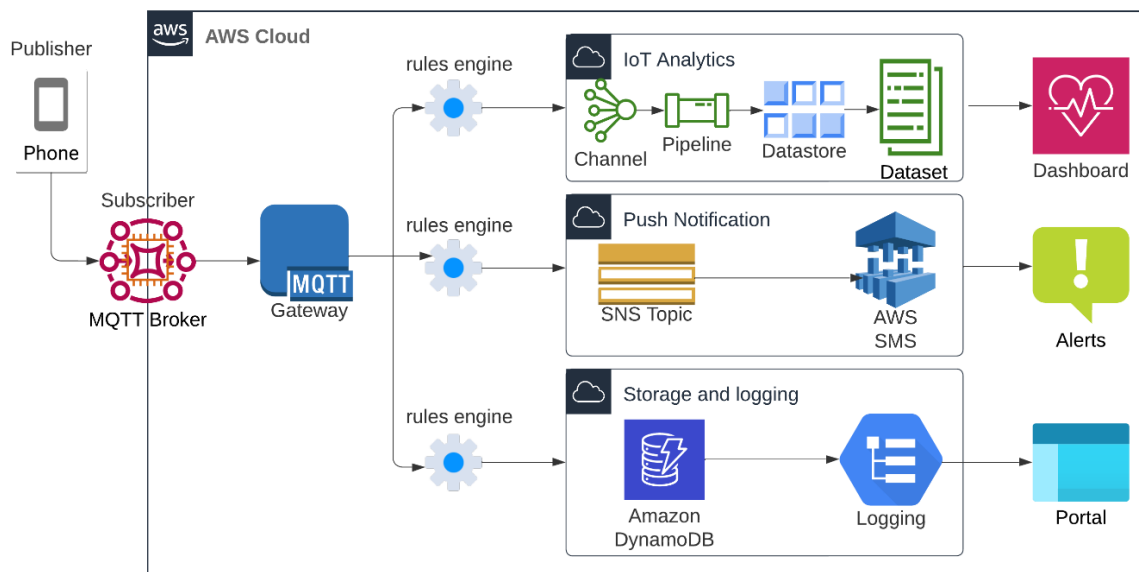


Figure 11. IoMT framework, cloud architecture



Project Lifecycle

This section addresses the project lifecycle. Before discussing the project lifecycle, this project follows a hybrid management approach by combining waterfall and agile practices. In an agile practice, the project phases overlap, and tasks are completed in iterations, which are called sprints. The timeline of sprints can range from two to four weeks and the goal is to focus on delivering value quickly to stakeholders and clients. In

the waterfall practice, is a sequential order of project phases linked to clearly defined expectations and goals, which is a linear approach.

In phase one, initiating the project: this is the launchpad for the entire process of the project. In this phase, we define project goals and deliverables, identify the budget and resources, and any other details that can impact the successful completion of the project (*Google Project Management*;, 2021a). Consequently, I will write a detailed set of instructions, consisting of defining the smartphone application features and each component of the project. By documenting all this information in one place such as a proposal will showcase the project's value and earn an approval from faculty members to move forward with it.

In phase two, creating a plan, this is a breakdown of all the tasks that will need to be completed. For example, a schedule, resources, and a plan for risk and change such as what to do in case my project encounter unexpected problems (software changes, quality control, sabbatical leave).

In phase three, executing and completing tasks: the development of the smartphone application and implementation of the cloud architecture. I will then attempt to develop each component and look for alternatives when needed. For those components that cannot be implemented due to complexity, show some possible further steps.

Subsequently, as part of the designing process, designers follow major conventions, and affordances, understand context such as the user's emotional state and implement innovative user experience (UX) through a friendly interface (Kim, 2015). In the current era, an everyday computational device or interactive software revolves around the principles defined in Human-Computer Interaction (HCI). The reason is, that human

beings are complex. Hence, HCI uses a variety of disciplines such as engineering, psychology, ergonomics, and design, to create a more compelling product than competitors, and an efficient product for users (Kim, 2015). A good user experience includes simplicity and creativity such as removing unnecessary user steps, buttons, and pages. Moreover, at NEIU, Java is the common programming language that is taught throughout computer science courses. If this project is extended horizontally or vertically, another student will be able to pick up and continue within a short period of time.

Consequently, I will ensure the code base is modular and remove the redundancy of extra lines of code by making use of packages, classes, methods, and parameters. Inside each package we have classes and inside each class, we have its methods. It is important to call the methods and pass the parameters whenever needed. In addition, I will ensure the code is commented on and simple naming conventions are used. So, if the project expands in the future, it will be easier for developers for transitioning and refactoring as necessary. Accordingly, after the implementation is over, there will be user testing for the application use cases and features. This ensures when presenting a live demo such as to stakeholders, they are happy with the results. Some examples of bugs testers should be on the lookout for include functional errors, algorithmic errors, data duplication, error handling, and hardware defects.

In phase four, closing the project, all the resources have been accounted for, have crossed the timeline, and it is time to close the project. Why is it important to close? Closing the project is a chance to evaluate how the project went. You can make note of what worked and what did not so you can plan better for next time. Even if the project is a massive success, it is helpful to take time to reflect. Closing the project is also a great

way to connect with anyone from within or outside the organization who may have had an interest in the project's goals and outcomes (*Google Project Management*., 2021).

Project Use Cases

This section addresses the use cases defined for the IoMT framework. The below use cases for the patient are meant to give a direction (see Tables 1-5). The plan is to implement as many use cases as possible during the project timeline. Defining use cases of the users before moving to the prototype phase will help reduce future refactoring and gives a clear picture of the application features that will need to be developed. For the next paragraphs, I will be narrating in the user story template commonly used in agile methodology. This “particular template, often referred to as ‘As a... I want to... So That...’ (*User Story Templates in Agile*, 2015).

Table 1. Capturing real-time data via internal sensors.

Sensor	Measurements	Medical Relevance
Ambient	Indoor	What is the temperature around the patient?
GPS	Location	Where is the patient located?
Accelerometer	Physical activity	How active is the patient?
Camera image	Pulse	What is the pulse of the patient?

As a patient, I want to capture the indoor temperature so that my doctor can learn about my environment. As a patient, I want to capture the current location so that my doctor can know where I am located. As a patient, I want to capture my physical activity so that my doctor can know how active I am. As a patient, I want to capture my pulse so that my doctor can notify me of any irregularities.

Table 2. Capturing real-time data via medical sensors.

Sensor	Measurements	Medical Relevance
ECG	Electrocardiography (ECG)	What is the heart rate of the patient?
PPG	Photoplethysmography (PPG)	What is the heart variation of the patient?

As a patient, I want to capture my ECG so that my doctor can learn about my heart condition. As a patient, I want to capture my PPG so that my doctor can learn my heart variation.

Table 3. Capturing real-time data via external services.

Service	Measurements	Medical Relevance
Weather	Temperature	What is the temperature around the patient?
Weather	Humidity	What is the humidity around the patient?
Weather	Pressure	What is the pressure around the patient?
Weather	Weather Pattern	What are the cloud conditions around the patient?

As a patient I want to capture multiple weather parameters such as the temperature, humidity, pressure, and cloud pattern so that my doctor can learn about my environmental surroundings.

Table 4. Capturing real-time data via wearable devices.

Sensor	Measurements	Medical Relevance
Altimeter	Air pressure	What is the altitude of the patient?
Heart rate	Pulse	What is the heart rate of the patient?
Pulse oximeter	SpO2	What is the oxygen level of the patient?

As a patient, I want to capture my air pressure so that my doctor can learn about my altitude. As a patient, I want to capture my pulse and SpO2 so that my doctor can notify me of any irregularities.

Table 5. Capturing real-time data via wearable devices.

Metrics Measurements	Medical Relevance
Duration of exercise	What is the exercise routine of the patient?
Hours slept	What is the sleep pattern of the patient?

As a patient, I want to capture the duration of exercise so that my doctor can learn about my routine. As a patient, I want to capture the hours I slept so that my doctor can learn about my sleep pattern.

Prototyping

When prototyping, the best practice is to include stakeholders or clients' input so they can approve the design before the development phase. As illustrated below, the prototype designed for the mobile personal healthcare application (see Fig. 12-23).

Figure 12. Prototype, landing page

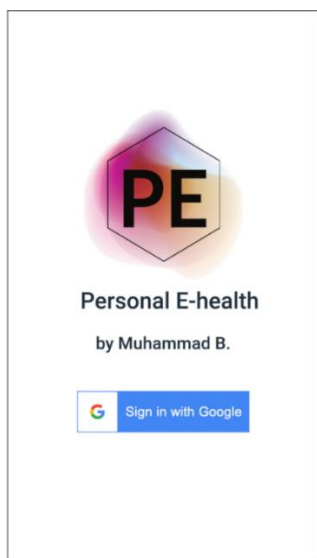


Figure 13. Prototype, user registration page

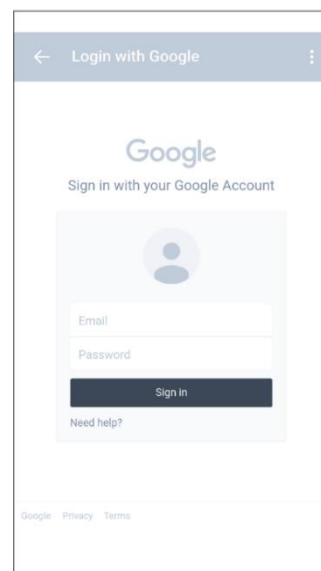


Figure 14. Prototype, home page

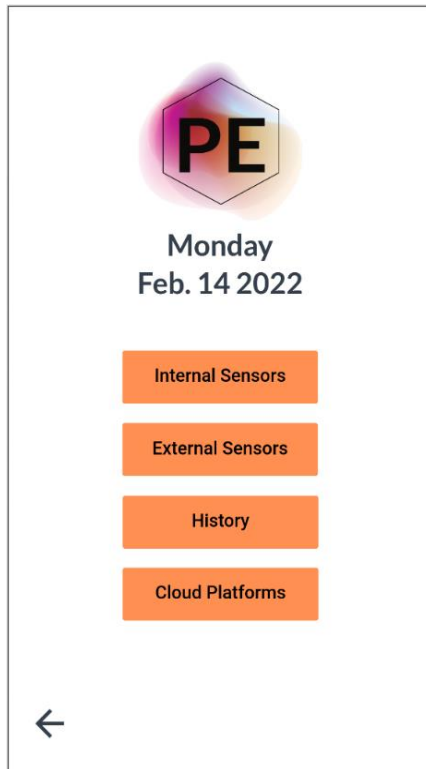


Figure 15. Prototype, internal sensors

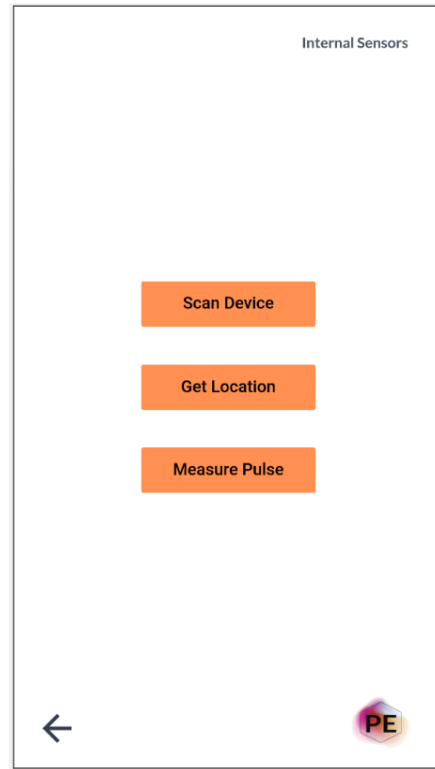


Figure 16. Prototype, scan sensors

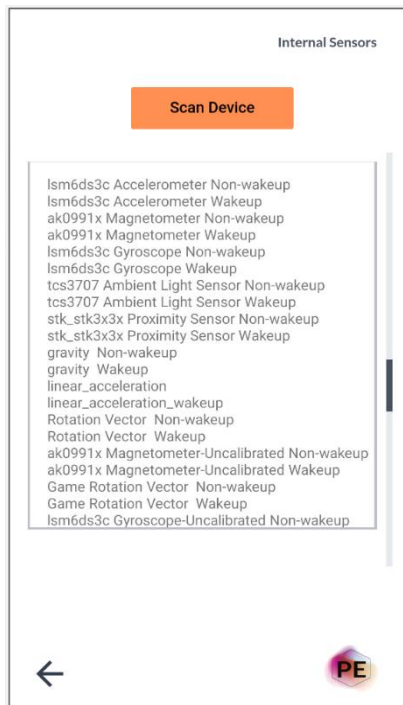


Figure 17. Prototype, get user location



Figure 18. Prototype, measure user pulse

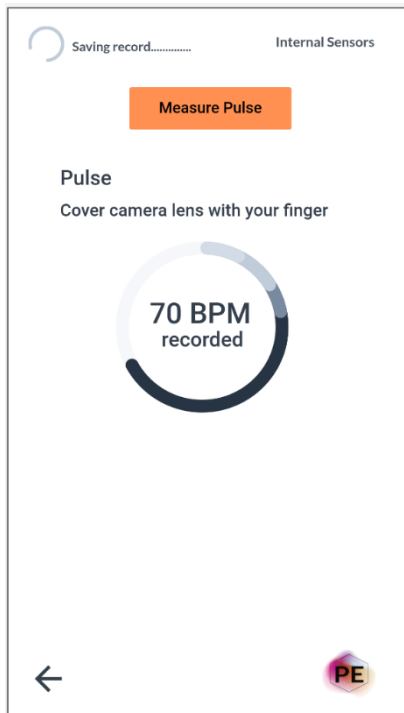


Figure 19. Prototype, external medical sensors

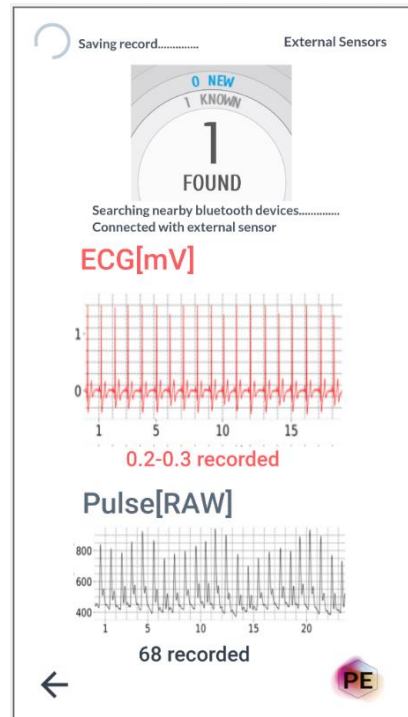


Figure 20. Prototype, store user records

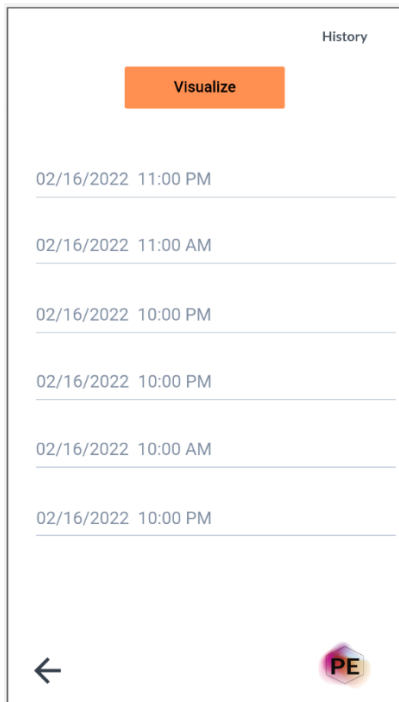


Figure 21. Prototype, view a user record



Figure 22. Prototype, user analytics

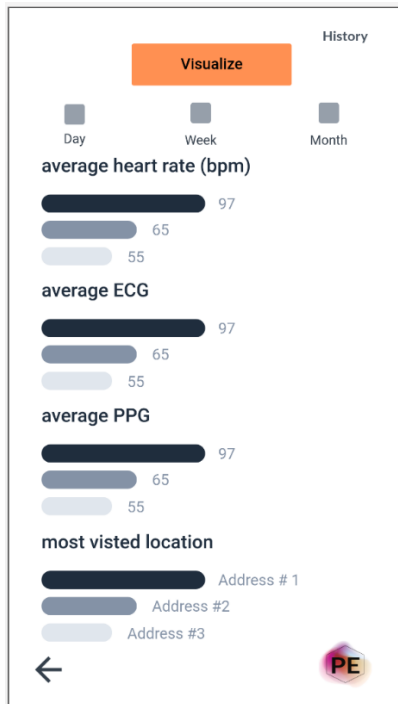
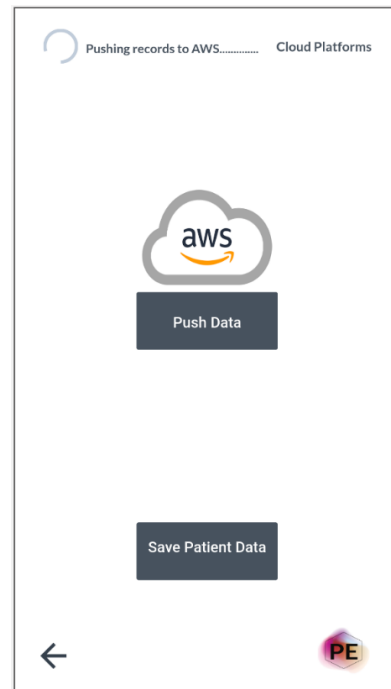


Figure 23. Prototype, push user record to broker



Mobile Authentication

The Google Cloud Adoption Framework (GCAF) assists with determining the essential tasks and goals that will speed up an organization’s cloud journey. In GCAF, there are four themes defined to develop a cloud-first organization. That is introducing stakeholders and their teams to four adoption themes “continuously learn, effectively lead, efficiently scale, and comprehensively secure in the cloud” (Google Cloud, 2022).

Google Cloud has many services for accomplishing a variety of technical and business objectives. A service called Firebase helps adapt the fourth GCAF theme, “Security”. Leveraging authentication with a Gmail account will help the patient feel peace of mind. One of the benefits of cloud adoption is that in-house security does not need to be developed as the provider have comprehensive security models deployed.

As illustrated in the figures below, when a new user is created, Firebase employs a randomly generated unique identifier or UUID. This identifier has a low probability of

being repeated. Its importance takes effect when processing the record over the cloud. The UUID serves as a key when categorizing retrieved records from different users. The first step to authentication is utilizing the Firebase console. I will ensure the account has billing enabled and then add the application package name. The Firebase console will then generate google-services.json which will be specific to the project files. Below figures showcase the security and user registration implementation (see Figs. 24-31).

Figure 24. Enabling mobile authentication with Firebase

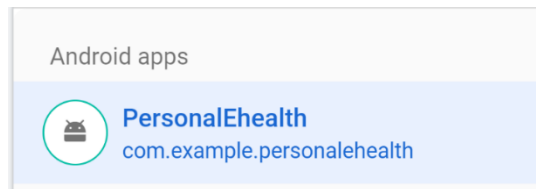


Figure 25. User registration version 1

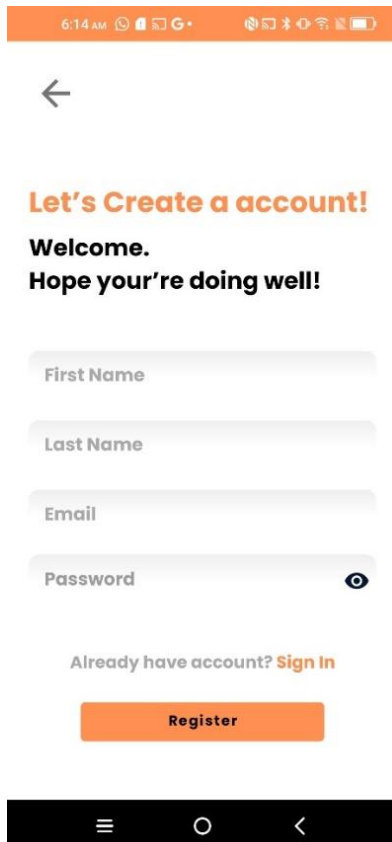
A screenshot of a mobile app's user registration screen. At the top, there's a back arrow and a title 'Let's Create a account!'. Below the title, a welcome message says 'Welcome. Hope your're doing well!'. The form consists of four input fields: 'First Name', 'Last Name', 'Email', and 'Password'. The 'Password' field has an eye icon to toggle visibility. At the bottom, there's a link 'Already have account? Sign In' and an orange 'Register' button. The bottom navigation bar shows a hamburger menu, a home icon, and a back arrow.

Figure 26. User registration version 2

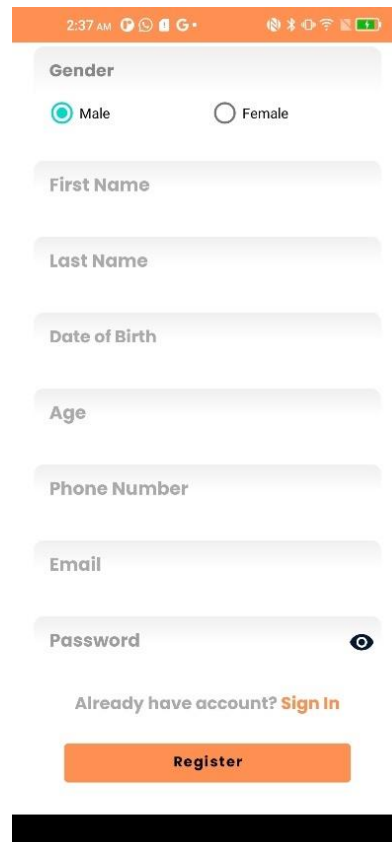
A screenshot of a mobile app's user registration screen, version 2. It includes a back arrow at the top left. The form starts with a 'Gender' section with radio buttons for 'Male' (selected) and 'Female'. This is followed by input fields for 'First Name', 'Last Name', 'Date of Birth', 'Age', 'Phone Number', 'Email', and 'Password'. The 'Password' field has an eye icon. At the bottom, there's a link 'Already have account? Sign In' and an orange 'Register' button. The bottom navigation bar is partially visible, showing a hamburger menu and a home icon.

Figure 27. User input of birthdate

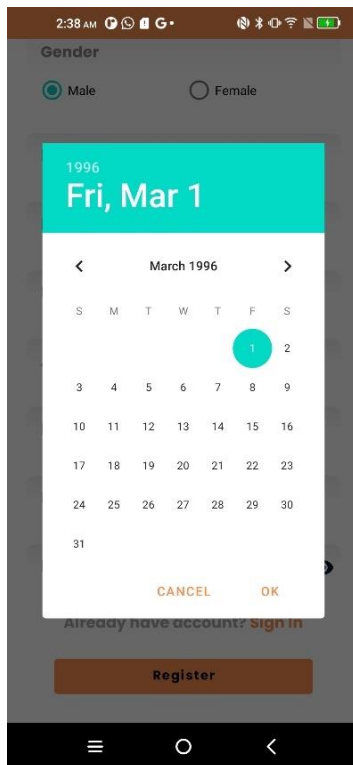


Figure 28. User login page

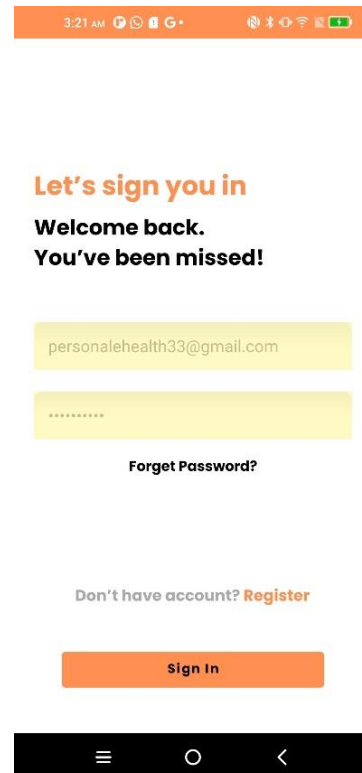


Figure 29. Firebase Authentication and Real Time Database

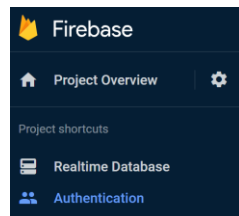


Figure 30. Observe authenticated users over Firebase console

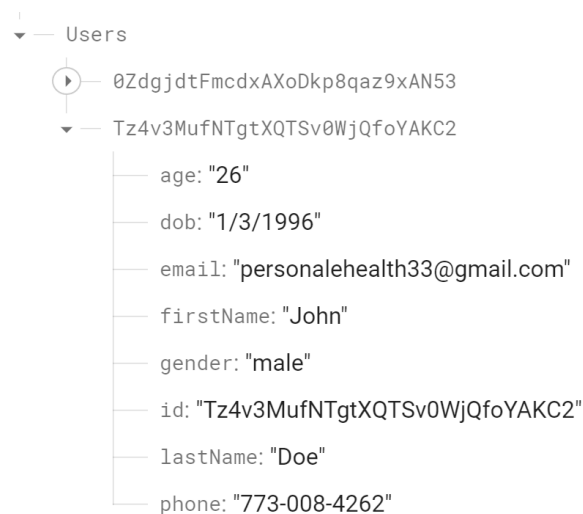
Identifier	Providers	Created ↓	Signed In	User UID
personalehealth66@gmail...	📧	Oct 18, 2022	Oct 23, 2022	0ZdgjdtFmcdxAXoDkp8qaz9xAN53
personalehealth33@gmail...	📧	Oct 18, 2022	Oct 23, 2022	Tz4v3MufNTgtXQTSv0WJQfoYAKC2

Figure 31. Observe daily users over Firebase console



There is an importance to retain certain user attributes such as the UUID, first name, last name, date of birth, age, etc. For instance, if a user deletes the mobile application, the attributes will also be deleted as they are synced locally. Having them synced with a real-time cloud database fixes the issue (see Fig. 32). Upon registration, when such attributes are generated and populated, in parallel, the application will sync with a NoSQL document database, Firebase Realtime Database. Data is available even when your app is not running because it is synced in real-time across all clients (Google Developers, 2022a). Data is synchronized in real-time to every connected client and stored in a JSON format. With Android and Apple devices you can create cross-platform apps that automatically update with the most recent data across all of your clients while using a single Realtime Database instance (Google Developers, 2022a).

Figure 32. Observe Firebase Real-Time Database



Local Mobile Database

In our framework, the information collected in real-time from the user environment builds the personal medical record. The database utilized for the mobile application is called SQLite Android. As it is lightweight, and embedded with the

operating system, Cleverroad, a mobile app development company, states “the main idea of SQLite is to get rid of server-client architecture and store all the app information directly on a mobile device. In such a manner, most Android developers have been using it for more than 20 years for the projects where they need to store the data on a device, not on a server” (Roshina, 2022). SQLite is now implemented and the PMRs are being successfully stored (see Fig. 33-36). Each attribute populates after its data has been collected. The user history is sorted by date and time. Every time a user pushes a record to the cloud, that record will be moved to the database. After, a new empty record will be created to then be further populated. One function to further improve the user experience is giving the ability s to manually edit, update, and delete the existing record.

Figure 33. View user profile and current record

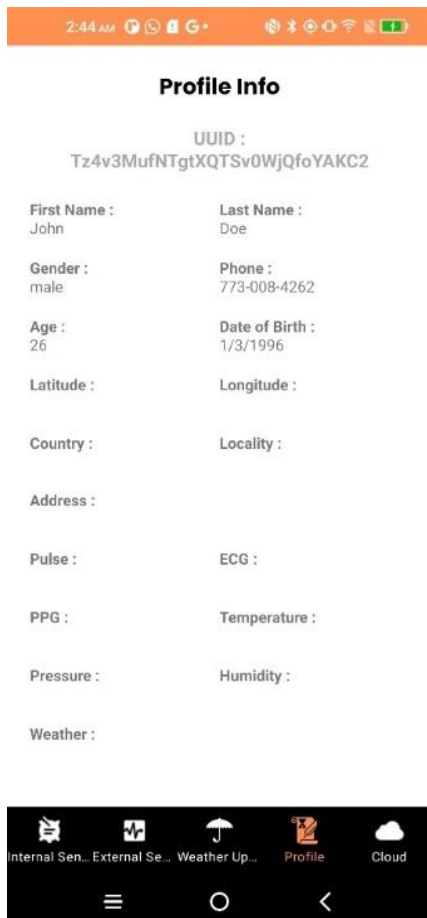


Figure 34. Click history button

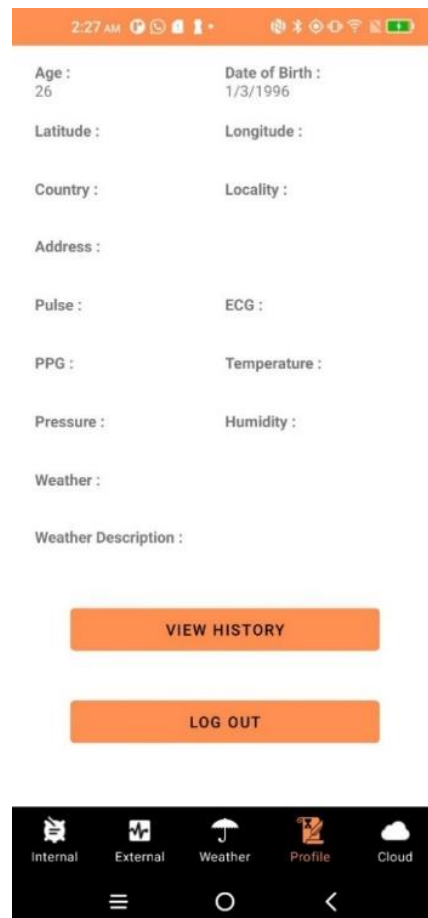


Figure 35. View history records



Figure 36. View history



Internal Sensors

Smartphones have embedded sensors or internal sensors. These sensors can be utilized to create a context aware PMR. Context-awareness is a systematic approach to gather information from the user surroundings. Some old smartphone sensors used to include environmental sensors: an ambient temperature-- provides the current ambient (room) temperature, heartbeat-- reports the heart rate of a user, pressure-- reports the pressure around the user, humidity-- reports the humidity around the user (Fig. 37).

The internal temperature sensor, “TYPE_TEMPERATURE”, cannot be used to accurately read the environment (Fig. 38). In contrast most mainstream smartphone devices have internal temperature sensors to prevent the CPU, battery, and components

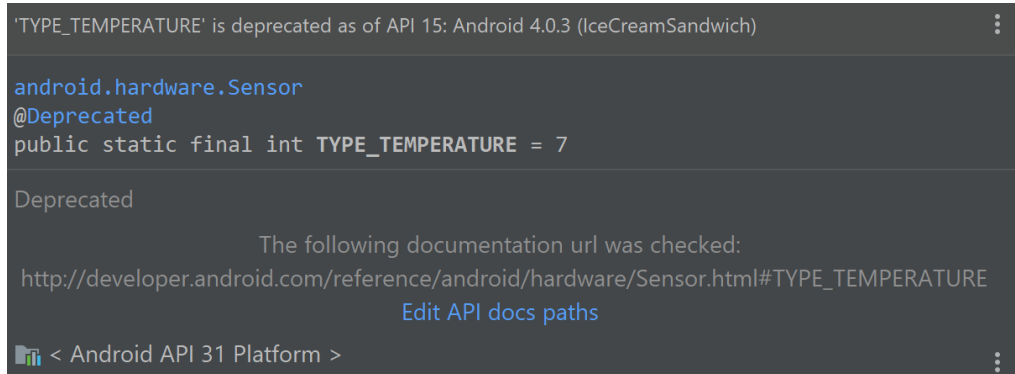
from overheating. These sensors inform the device to shut down when the temperature is either extremely hot or cold. There are applications in the Google Play Store that utilize this specific sensor and customized algorithms to predict an ambient reading.

Figure 37. Smartphone environmental sensors

Sensor	Sensor event data	Units of measure	Data description
<code>TYPE_AMBIENT_TEMPERATURE</code>	<code>event.values[0]</code>	°C	Ambient air temperature.
<code>TYPE_LIGHT</code>	<code>event.values[0]</code>	lx	Illuminance.
<code>TYPE_PRESSURE</code>	<code>event.values[0]</code>	hPa or mbar	Ambient air pressure.
<code>TYPE_RELATIVE_HUMIDITY</code>	<code>event.values[0]</code>	%	Ambient relative humidity.
<code>TYPE_TEMPERATURE</code>	<code>event.values[0]</code>	°C	Device temperature. ¹

Note. Retrieved from Google LLC. *Environment sensors*. Accessed 08 Aug. 2022.

Figure 38. Type Temperature sensor is deprecated

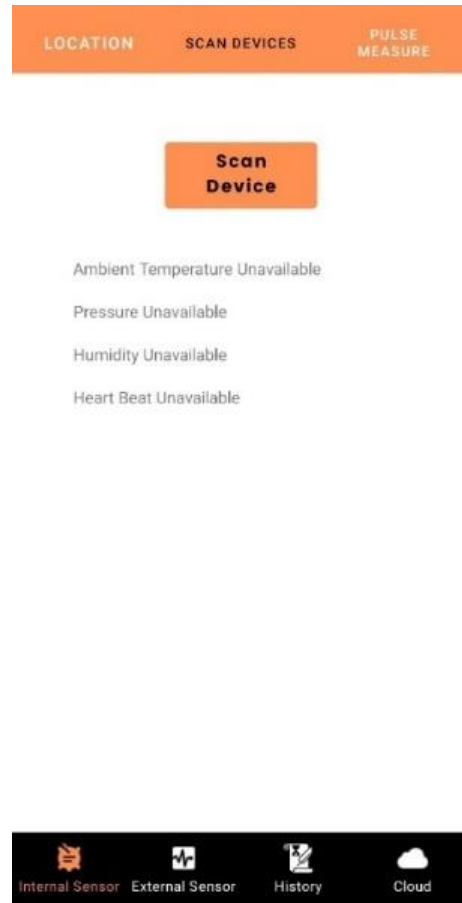


Note. Retrieved from Google LLC. *Type Temperature Sensor*. Accessed 08 Aug. 2022.

Consequently, it is a rarity now to find a sensor in modern mainstream smartphones that is dedicated to accurately report a user's ambient room temperature. Additionally, Samsung and Motorola, on the early stages of the smartphone era, used to equip their Android generations such as Galaxy S4 and Note 3, Moto X smartphones with physical temperature sensors (Shams, 2016). However, these manufacturers did not trust the accuracy due to the internal body heat generated from different device components. Consequently, on the mobile application I have created a quick function to scan the

device's environmental sensors and will return true if any of them are found. I have tested this function with an Android smartphone called TCL 10L and it returned false (Fig. 39).

Figure 39. Scan smartphone environmental sensors



For retrieving the current location, the internal sensor Global Positioning System (GPS) will be utilized. The grid system that enables us to locate absolute or precise places on the surface of the Earth is made up of latitude and longitude (National Geographic Society, 2010). Latitude and longitude can be used to pinpoint certain locations and for identifying landmarks. For pharmacies who only wish to reach a customer base in the vicinity of each of their establishments, latitude and longitude targeting can be an option (Healthy Ads, 2021). The app utilized an Android class called Geocoder. According to the documentation, Geocoder is a “class for handling geocoding and reverse geocoding.

Geocoding is the process of transforming a street address or other description of a location into a (latitude, longitude) coordinate” (Google Developers, 2022b). Geocoder has been implemented and is functioning (as illustrated in Fig. 40-41). Programmatically, I will first check if the GPS status of the user is enabled. If enabled, the application will then try to retrieve the location. The location is pinpointed using satellites.

Figure 40. Retrieve user location, GPS disabled

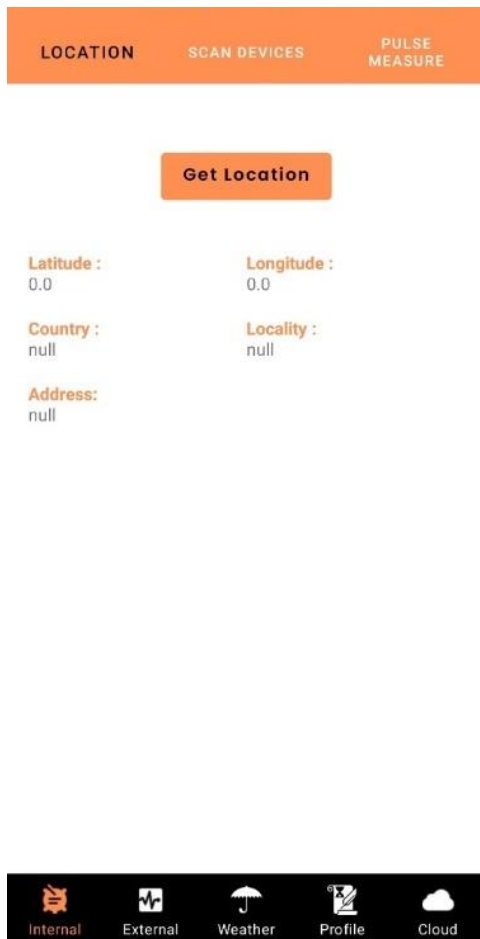
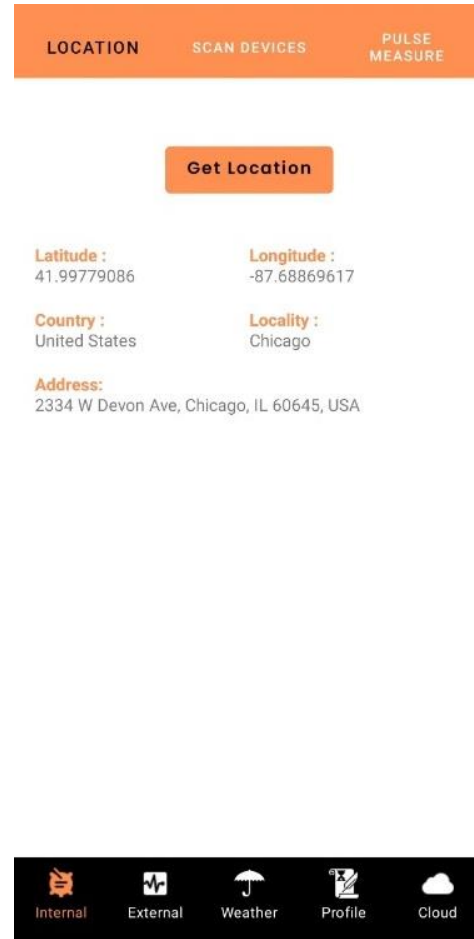


Figure 41. Retrieve user location, GPS enabled



Measuring the pulse was achieved through the camera image sensor. Simply the user needs to place finger on the rear-facing camera lens and the reading will be recorded via a customized algorithm (as illustrated in Fig. 42-43).

Figure 42. Camera sensor homepage

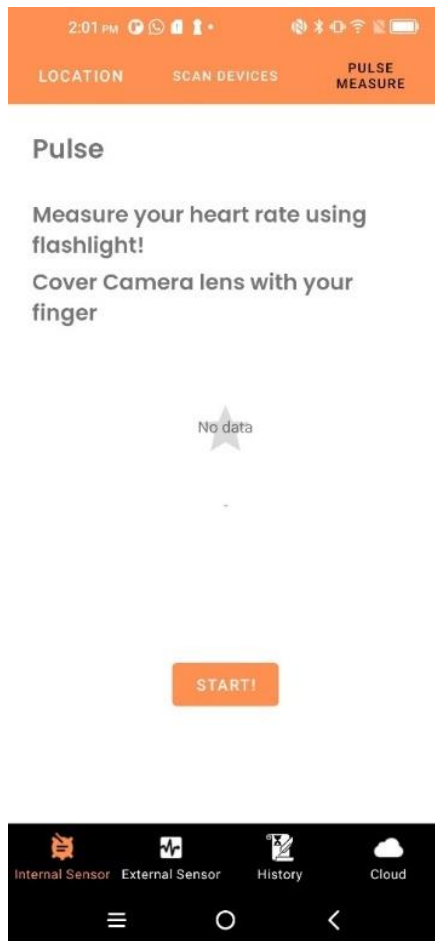


Figure 43. Camera sensor integration results



Another sensor that can help further expand the record is the accelerometer, which is a motion sensor. Although, I was not able to implement it, according to the documentation:

Most Android-powered devices have an accelerometer, and many now include a gyroscope. The availability of the software-based sensors is more variable because they often rely on one or more hardware sensors to derive their data. (Google Developers, 2022c)

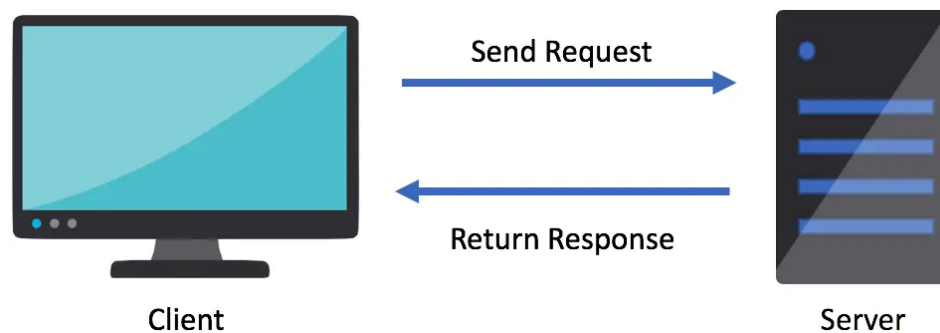
External Weather Services

The weather information surrounding the patient can be obtained through an Application Programming Interface (API). There are several benefits of utilizing APIs.

For instance, some benefits include outsourcing, increased mobility, moving from one service by changing data endpoints, increased developer productivity, and connectivity.

For example, when a client (user) requests information such as weather, the healthcare application will send a request to the server. The request is handled over a server, doing all the heavy lifting as their computing power is more powerful than personal computers or mobile phones. Finally, returning the needed information as a response back to the user (Fig. 44).

Figure 44. API client request and server response



Note. Retrieved from Salesforce Trailhead. *API Basics*. Accessed 20 Aug. 2022.

There are several weather APIs that developers can use for the integration. Tomorrow.io, is a paid service, and is enterprise-grade (i.e., used by Uber and Google Cloud). It provides worldwide assistance and returns meteorological information on weather patterns, moon phases, pollen counts, and air quality, as well as fire danger (Bush, 2019). Since this is a paid service, I researched and opted for a free one instead, called OpenWeatherMap API (see Table 6).

Table 6. Weather API Survey for External Services.

Weather API	Pricing	Free Tier
OpenWeatherMap	free, paid plans	1,000 calls per day

Stormglass	free, paid plans	50 requests per day
Yahoo Weather	free, paid plans	2,000 calls per day
AccuWeather	limited trial, paid plans	50 calls per day

OpenWeatherMap through one API call can give essential weather data in one response. As shown in the below, the API has been implemented (see Fig. 45-46).

Figure 45. Retrieve weather info test 1

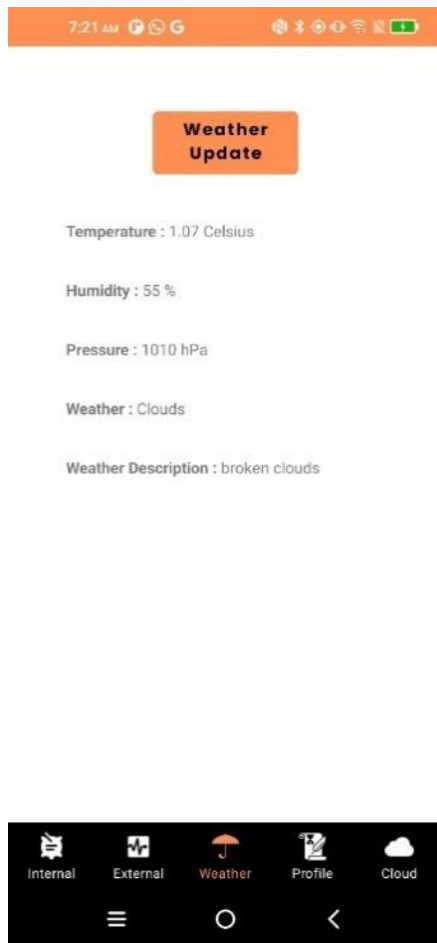
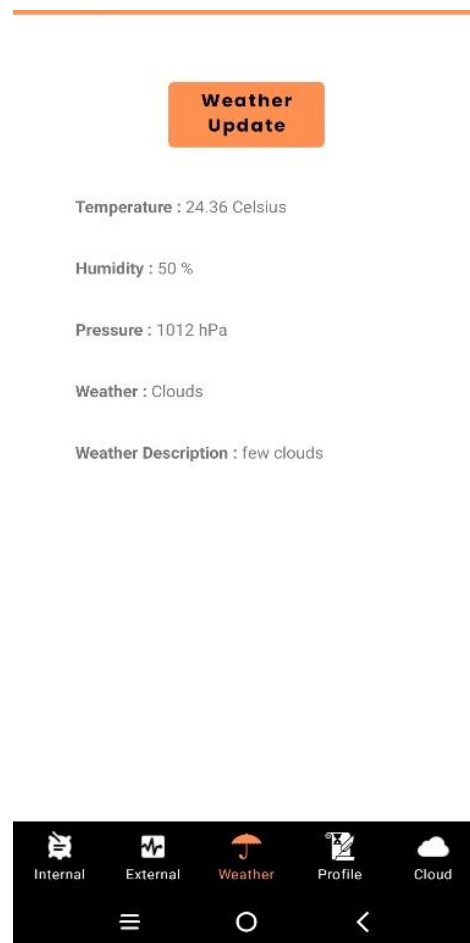


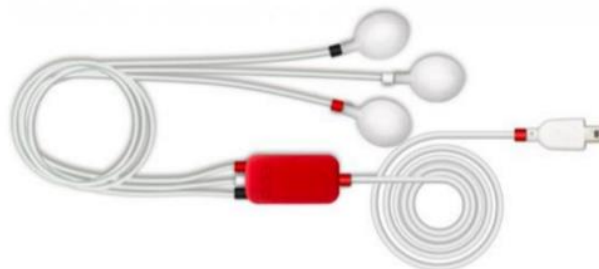
Figure 46. Retrieve weather info test 2



External Medical Sensors

The external medical sensors purchased is a bundle called BITalino HeartBIT (see Fig. 47). According to their website, “BITalino is an affordable & open-source biosignals platform designed for Education & Prototyping. This is the ideal toolkit to be used in a

laboratory & classroom environments or to create prototypes and applications using physiological sensors” (pluxbiosignals, 2022). They also added “The BITalino HeartBIT bundle, which is designed for everyone who wants to measure cardiac activity measurements (e.g., heart rate & heart rate variability) by evaluating electrocardiography (ECG) and Photoplethysmography (PPG) signals. This bundle is completely assembled with our 3D Printed Casing for BITalino (r)evolution Plugged making it more convenient to use, wear, share, and transport. The core of our HeartBIT is our BITalino Core BT allowing the user to connect more sensors in the future if convenient” (*HeartBIT*, 2022).



Note. Retrieved from (HeartBIT, 2022). *Plux Wireless Biosignals*. Accessed 20 Jan. 2022.

having to develop from scratch. Since they have made this API open source, further improvements were made to their baseline codebase (see Fig. 48-55). The samples read were 1000 frames. Inside each frame is the real-time data of the sensor. By utilizing Androidplot, a library for visualization, I was able to optimize the library and showcase the frames in real time. By developing a save and stop button, the average of all frames was taken from the value at index zero before storing it in the local database.

Figure 48. BITalino original API

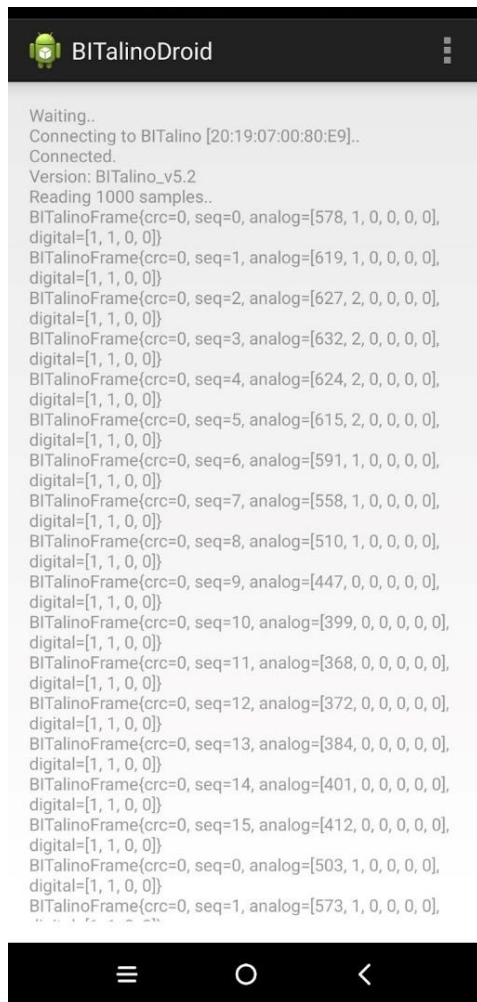


Figure 49. BITalino iteration #1

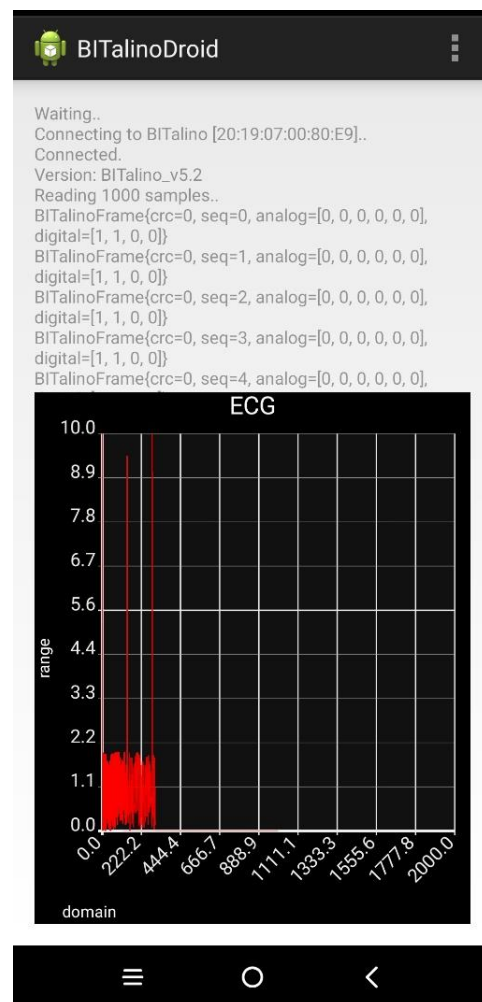


Figure 50. BITalino iteration #2

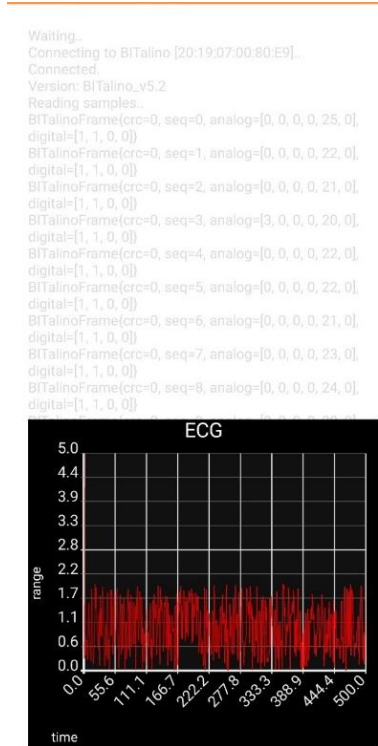


Figure 51. BITalino iteration #3

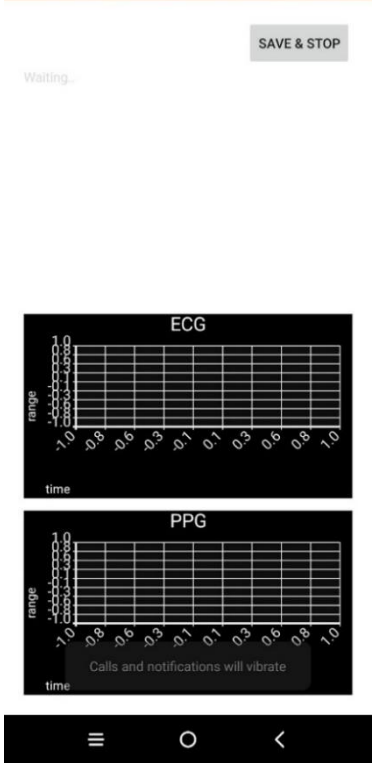


Figure 52. BITalino iteration #4

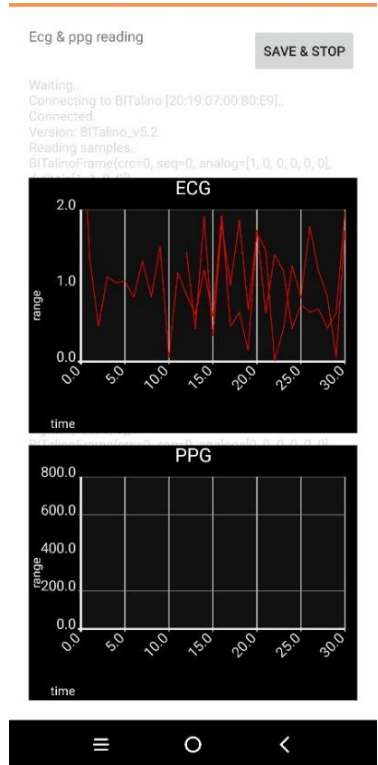


Figure 53. BITalino iteration #5

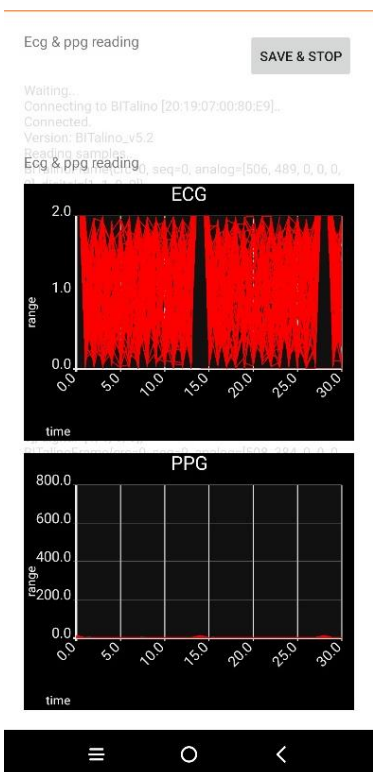


Figure 54. BITalino ECG sensor final iteration

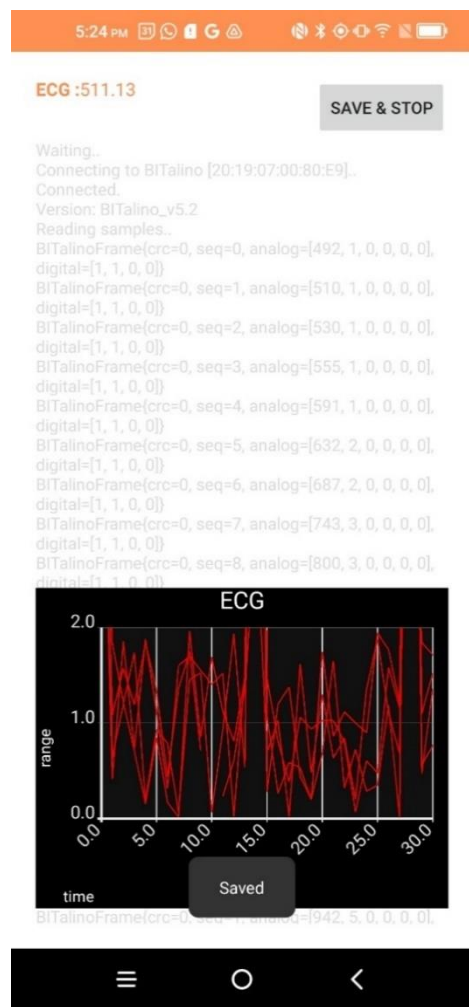
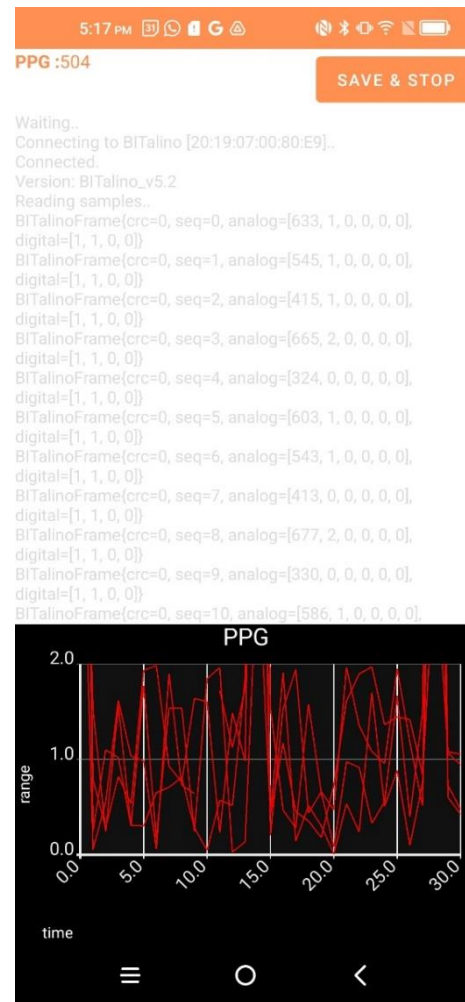


Figure 55. BITalino PPG sensor final iteration



Personal Medical Record Schema

In the below table is an entire snippet of one personal medical record (see Fig. 56). This is how the record is displayed when sent over to the customized data pipeline. The purpose of using the JSON schema is that it will be easier to visualize how the data is structured. In the future, if required, restructuring the schema can be done quickly.

Figure 56. Personal Medical Record JSON Schema

```
{
  "User": {
    "Identification_Information": {
      "id": "Tz4v3MufNTgtXQTSv0WjQfoYAKC2",
```

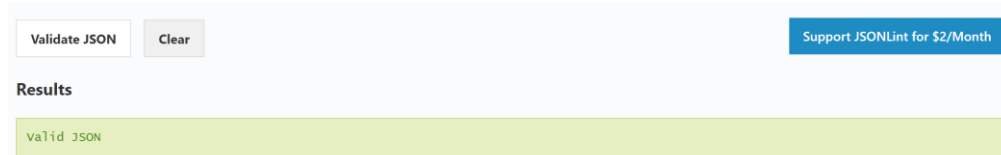
```

        "first_name": "John",
        "last_name": "Doe",
        "age": "26",
        "DOB": "1/3/1996",
        "phone_no": "773-008-4262",
        "email": "personalehealth33@gmail.com",
        "gender": "male",
        "address": "2045 W Devon Ave, Chicago, IL 60659, USA",
        "country": "United States",
        "city": "Chicago",
        "longitude": "-87.68185752",
        "latitude": "41.99770812"
    },
    "VitalSigns": {
        "ECG": {
            "sensor_name": "BITLANO ECG",
            "value": "510.52",
            "date": "null",
            "location": {
                "longitude": "-87.68185752",
                "latitude": "41.99770812"
            }
        },
        "PPG": {
            "sensor_name": "BITLANO PPG",
            "value": "506",
            "date": "null",
            "location": {
                "longitude": "-87.68185752",
                "latitude": "41.99770812"
            }
        },
        "pulse": {
            "sensor_name": "Mobile Camera Sensor",
            "value": "131",
            "date": "null",
            "location": {
                "longitude": "-87.68185752",
                "latitude": "41.99770812"
            }
        }
    },
    "Weather_Information": {
        "temperature": "24.36",
        "humidity": "50",
        "pressure": "1012",
        "weather": "Clouds",
        "weatherDescription": "few clouds"
    }
}

```

After the record is finalized, the next step is validating the record by using a free online service called JSONLint. Other services are also available. Validating the record will make further processing easier over the cloud. An invalid record can lead to extra data filtering steps over the cloud to normalize the JSON payload. After observing the results, the record is found valid (see Fig. 57).

Figure 57. Personal medical record JSON validation



Publishing a Record

MQTT is a protocol offering a publish and subscribe system. It allows you to publish from one device to another. MQTT allows a user to subscribe to a topic. That means a user can receive messages if subscribed. A user can also send a message through publishing. And a user does not have to both publish and subscribe at the same time. A user can wish to either only subscribe or only publish. Through MQTT messages, a user can send files as long as they are in an acceptable payload format. For publishing the user record, I will first utilize the Paho Android Service, an MQTT client library written in Java (Eclipse Foundation, 2022). The Foundation states that “The Paho project has been created to provide reliable open-source implementations of open and standard messaging protocols aimed at new, existing, and emerging applications for Machine-to-Machine (M2M) and IoT” (Eclipse Foundation, 2022).

After implementing the library, it was found through testing that the Java client was not compatible with Android 12, the twelfth release of the mobile operating system. However, through surveying repositories, I found a working modified Paho MQTT

library written in Kotlin. This library holds the same features as the Java one (see Fig. 58-60). Kotlin is also an object-orientated programming language like Java.

Figure 58. Features of modified Paho MQTT Android Service

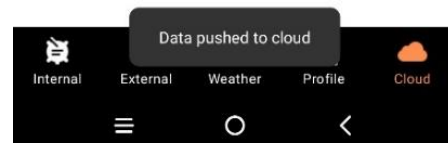
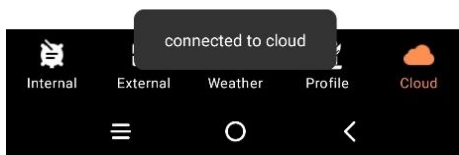
MQTT 3.1	✓	Automatic Reconnect	✓
MQTT 3.1.1	✓	Offline Buffering	✓
LWT	✓	WebSocket Support	✓
SSL / TLS	✓	Standard TCP Support	✓
Message Persistence	✓		

Note. Retrieved from a GitHub repository (hannesa2, 2022). *MQTT Android Service*. Accessed 23 Oct. 2022. <https://github.com/hannesa2/paho.mqtt.android>

Figure 59. Patient connected to cloud



Figure 60. Patient published PMR to cloud



Hosting MQTT HiveMQ Broker and Bridging to AWS IoT Core

In an MQTT system, a broker is responsible to receive all the messages and acknowledgments. The broker decides who is interested in receiving and publishing the message. It does this to all the subscribed users. There are several broker libraries available on the internet and are free to use. The library first tested for the MQTT broker is HiveMQ Community Edition, a Java-based open-source library (hivemq, 2022). This library will be set up on a server or instance of the Amazon Elastic Compute Cloud (EC2). As the instances link, EC2 offers secure, resizable capacity in the cloud.

A developer can develop and deploy apps more quickly using EC2 because there is no longer a need to make an upfront hardware investment (Amazon, 2022). A developer can launch as many or as few virtual servers as required, set up networking and security settings, and control storage using EC2 (Amazon, 2022). In addition, in the future if the user traffic of patients is high, EC2 will handle changes in demand or popularity spikes, eliminating the need for medical professionals to predict traffic.

Once the broker is set up and listening to incoming traffic from port 8080 (web) and 1883 (smartphone devices), the next step is to implement a manual configuration. So, that the broker can recognize the generated security certificates from the authenticated AWS console. With these certificates, the broker can then be authorized to communicate with AWS IoT Core. This bridge between the broker to the cloud services is through a middleware also called an MQTT Gateway. As mentioned before, IoT Core is the gateway we need to connect to. If this bridge fails, another broker library will need to be used and tested. It is crucial for this step to succeed as this part of the cloud architecture then allows the utilization of microservices to further process the PMR for healthcare

professionals. The testing result for the HiveMQ library is that it was listening to payloads (messages) but failed to accept the generated security certificates (see Fig. 61-64). Although, I reviewed HiveMQ's documentation, I was not able to pinpoint the issue.

Figure 61. AWS EC2 instance running for HiveMQ broker

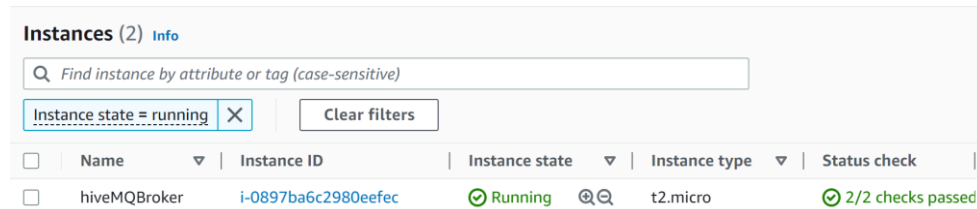


Figure 62. AWS EC2 securely connects with a unique key pair

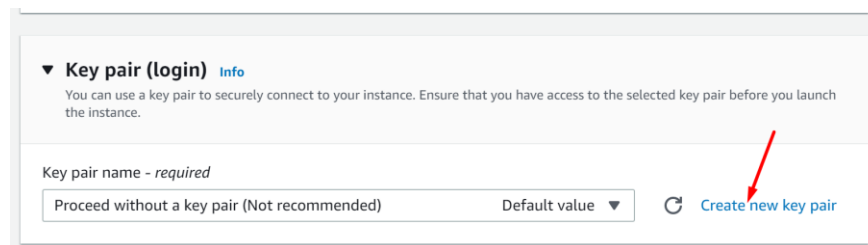


Figure 63. Editing EC2 inbound rules for incoming traffic

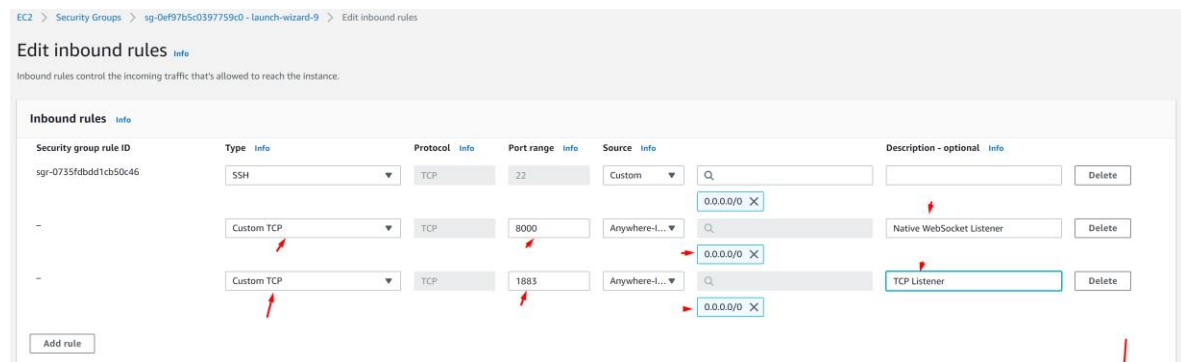
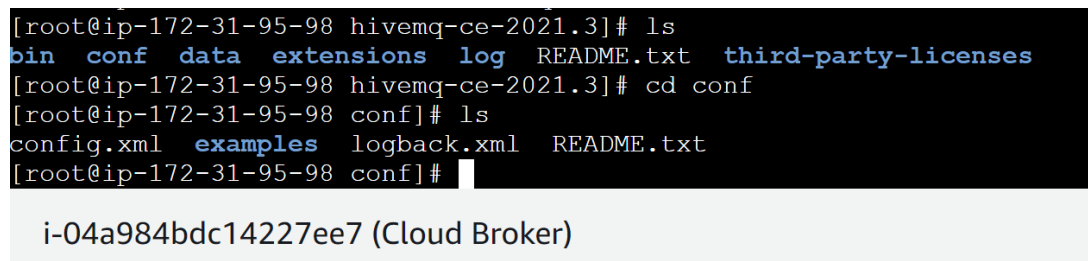


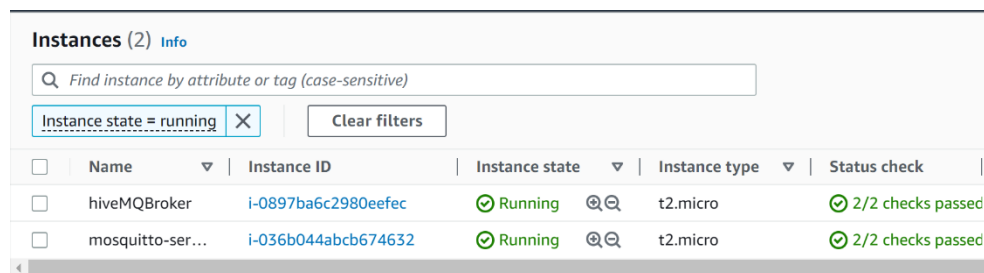
Figure 64. Editing HiveMQ broker config to bridge to IoT Core



Hosting MQTT Mosquito Broker and Bridging to AWS IoT Core

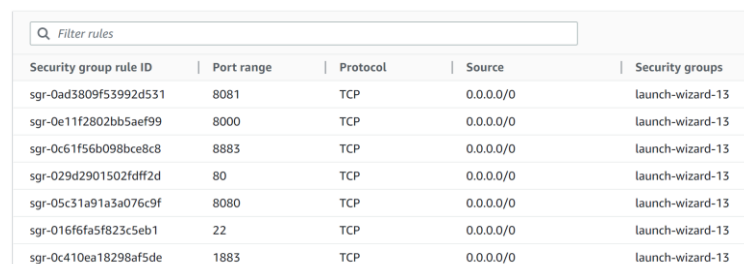
In this section, we address how the Mosquito library integrates with our framework. The second MQTT broker library to test the middleware, we were able to utilize and optimize the open-source Mosquito library from Eclipse to successfully bridge communication with AWS IoT Core. As mentioned before AWS IoT Core serves as the gateway between the broker and cloud services such as IoT Analytics and QuickSight. This library is not written in Java but in C. This library is also open source. The Mosquito library also features a username and password. Without these credentials a user will not be able to communicate with the broker. In addition, without the AWS credentials generated from IoT Core being recognized, the cloud services of an organization will not be able to communicate with the broker. As illustrated in the Figures 65-70 below, after creating an EC2 instance, opening the traffic, adding the credentials, creating rules, and a customized configuration allowed a successful bridge between the broker and IoT Core (our gateway to cloud microservices)!

Figure 65. AWS EC2 instance running for Mosquito broker



Instances (2) Info					
<input type="text" value="Find instance by attribute or tag (case-sensitive)"/>					
Instance state = running Clear filters					
<input type="checkbox"/>	Name	Instance ID	Instance state	Instance type	Status check
<input type="checkbox"/>	hiveMQBroker	i-0897ba6c2980eefec	Running	t2.micro	2/2 checks passed
<input type="checkbox"/>	mosquito-ser...	i-036b044abcb674632	Running	t2.micro	2/2 checks passed

Figure 66. Editing EC2 inbound rules for incoming traffic



<input type="text" value="Filter rules"/>				
Security group rule ID	Port range	Protocol	Source	Security groups
sgr-0ad3809f53992d531	8081	TCP	0.0.0.0/0	launch-wizard-13
sgr-0e11f2802bb5aef99	8000	TCP	0.0.0.0/0	launch-wizard-13
sgr-0c61f56b098bce8c8	8883	TCP	0.0.0.0/0	launch-wizard-13
sgr-029d2901502fdff2d	80	TCP	0.0.0.0/0	launch-wizard-13
sgr-05c31a91a3a076c9f	8080	TCP	0.0.0.0/0	launch-wizard-13
sgr-016f6fa5f823c5eb1	22	TCP	0.0.0.0/0	launch-wizard-13
sgr-0c410ea18298af5de	1883	TCP	0.0.0.0/0	launch-wizard-13

Figure 67. Generating AWS security certificates from IoT Core

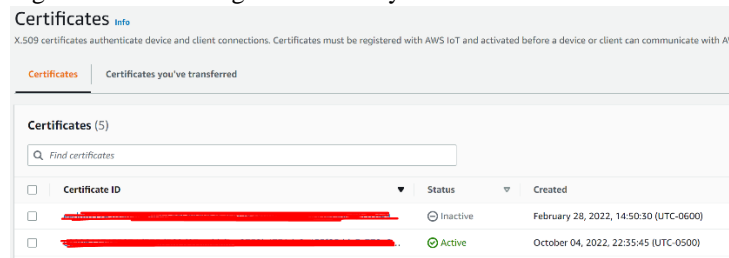


Figure 68. Rules to route traffic from topic to IoT Analytics

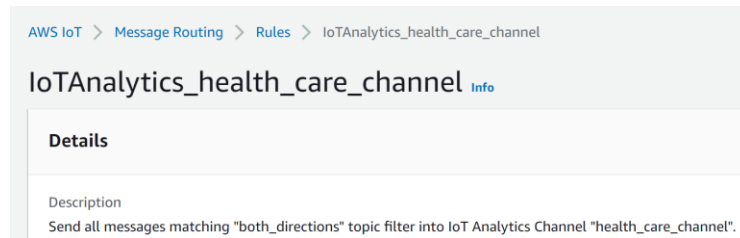


Figure 69. Adding AWS certificates to Mosquito broker

```
ubuntu@ip-172-31-88-23:/etc/mosquitto$ ls
aclfile.example certs mosquitto.conf mosquitto.conf.example passwd pskfile.example pwfile.example
ubuntu@ip-172-31-88-23:/etc/mosquitto$ cd certs
ubuntu@ip-172-31-88-23:/etc/mosquitto/certs$ ls
cert.crt private.key public.key rootCA.pem
ubuntu@ip-172-31-88-23:/etc/mosquitto/certs$
```

Figure 70. Custom Mosquito broker configuration

```
# Specifying which topics are bridged and in what fashion
topic awsiot_to_localgateway in 1
topic localgateway_to_awsiot out 1
topic both_directions both 1

# Setting protocol version explicitly
bridge_protocol_version mqttv311
bridge_insecure false

# Bridge connection name and MQTT client Id, enabling the connection automatically when the broker starts.
cleansession true
clientid bridgeawsiot
start_type automatic
notifications false
log_type all

#
# Certificate based SSL/TLS support
#

# Path to the rootCA
bridge_cafile /etc/mosquitto/certs/rootCA.pem

# Path to the PEM encoded client certificate
bridge_certfile /etc/mosquitto/certs/cert.crt

# Path to the PEM encoded client private key
bridge_keyfile /etc/mosquitto/certs/private.key

#END of bridge.conf

listener 1883
protocol mqtt

listener 8080
protocol websockets

allow_anonymous false
password_file /etc/mosquitto/passwd
ubuntu@ip-172-31-88-23:/etc/mosquitto$
```

AWS IoT Analytics, Cron, AWS QuickSight

In this section, we address how AWS IoT Analytics, Cron, QuickSight integrates with our framework (Fig. 71-82). As illustrated in Figure 71, the first step is to collect the patient data. IoT Analytics is completely linked with AWS IoT Core, allowing it to receive messages from connected devices as they come in (Amazon LLC, n.d.). The second step is to process the received data. IoT Analytics can transport data to the IoT Analytics data storage after enriching it with information from outside sources, such as a weather forecast or in our case the patient personal medical records. The third step is to store the patient data. AWS IoT Analytics stores both processed and unprocessed data automatically so that you can process it later (Amazon LLC, n.d.). The fourth step is to analyze the data. Execute Ad-Hoc SQL Queries: AWS IoT Analytics offers a SQL query engine that enables you to execute SQL Queries and receive results quickly (Amazon LLC, n.d.). With the help of the service, you can extract information from the data storage using conventional SQL queries and schedule them using Cron. I have scheduled a Cron to run queries every other minute. The fifth step is to build and visualize the data through QuickSight. AWS IoT Analytics offers a connector to QuickSight which allows professionals to view health datasets in an interactive dashboard.

Figure 71. The following graphic shows an overview of how you can use AWS IoT Analytics



Note. Retrieved from Amazon. *AWS IoT Analytics: User Guide*. Accessed 25 Oct. 2022.

Figure 72. Creating a healthcare channel in AWS IoT Analytics

<input type="checkbox"/>	Name	Storage type	Status	Last message arrival time	Created	Last updated
<input type="checkbox"/>	health_care_channel	Service managed	Active	Oct 25, 2022 11:45:16 AM -0500	Oct 22, 2022 7:17:25 AM -0500	Oct 22, 2022 7:17:25 AM -0500

Figure 73. Creating an SQL query job for the health dataset

SQL query

```
SELECT
  User.Identification_Information.id AS user_id,
  User.Identification_Information.first_name AS FIRST_NAME,
  User.Identification_Information.last_name AS LAST_NAME,
  User.VitalSigns.ecg_sensor_name AS ECG_SENSOR_NAME,
  User.VitalSigns.ecg_value AS ECG_VALUE,
  User.VitalSigns.ecg_date AS DATE,
  User.VitalSigns.ppg_sensor_name AS PPG_SENSOR_NAME,
  User.VitalSigns.ppg_value AS PPG_VALUE,
  User.VitalSigns.ppg_location.longitude AS LONGITUDE,
  User.VitalSigns.ppg_location.latitude AS LATITUDE,
  User.VitalSigns.pulse_sensor_name AS PULSE_SENSOR_NAME,
  User.VitalSigns.pulse_value AS PULSE_VALUE,
  User.Weather_Information.temperature AS TEMPERATURE,
  User.Weather_Information.pressure AS PRESSURE,
  User.Weather_Information.weather AS WEATHER,
  User.Weather_Information.weatherDescription AS WEATHER_DESCRIPTION
FROM health_care_datastore LIMIT 1000
```

Figure 74. Creating a Cron expression for the SQL query job

Schedule

Cron expression

cron(0/1 * * * ? *)

Figure 75. Result overview of records after query is run

Result preview

6de04a4a-3680-4614-9ebd-2dbbe4710479.csv

user_id	FIRST_NAME	LAST_NAME	ECG_SENSOR_NAME	ECG_VALUE	DATE	PPG_SENSOR_NAME	PPG_VALUE
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	511.13	null	BITLANO PPG	504
Tz4v3MufNTgtXQTSvOWJQfoYAKC2	John	Doe	BITLANO ECG	511.13	null	BITLANO PPG	504
Tz4v3MufNTgtXQTSvOWJQfoYAKC2	John	Doe	BITLANO ECG	511.13	22/10/2022	BITLANO PPG	504
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	0.7	19/10/2022	BITLANO PPG	120
nINZCHukdcMeEAUnpaCivFx9Lu2	Muhammad	Bangash	BITLANO ECG	511.13	22/10/2022	BITLANO PPG	504
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	0.7	19/10/2022	BITLANO PPG	120
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	0.7	19/10/2022	BITLANO PPG	120
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	0.7	19/10/2022	BITLANO PPG	120
02dgdtdfmcdxAXoDkp8qaz9xAN53	John	Doe	BITLANO ECG	0.7	19/10/2022	BITLANO PPG	120

Figure 76. Connecting healthcare dataset with QuickSight

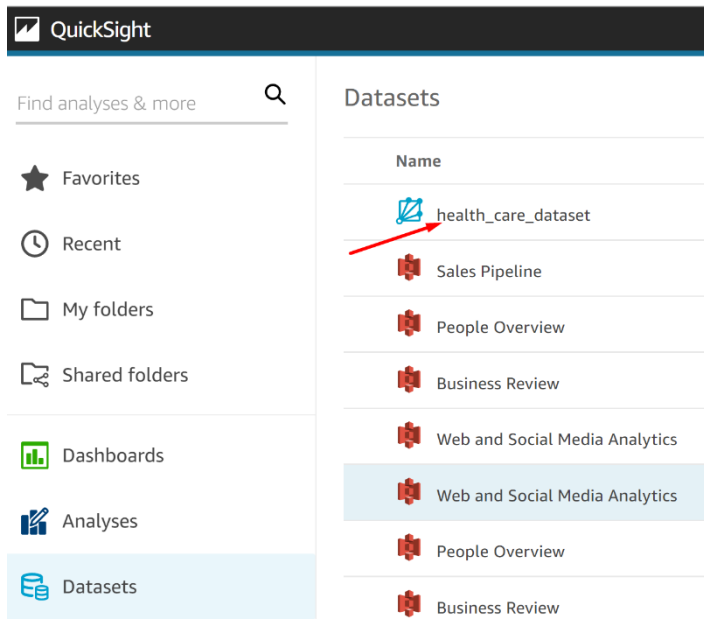


Figure 77. Filtering values in QuickSight

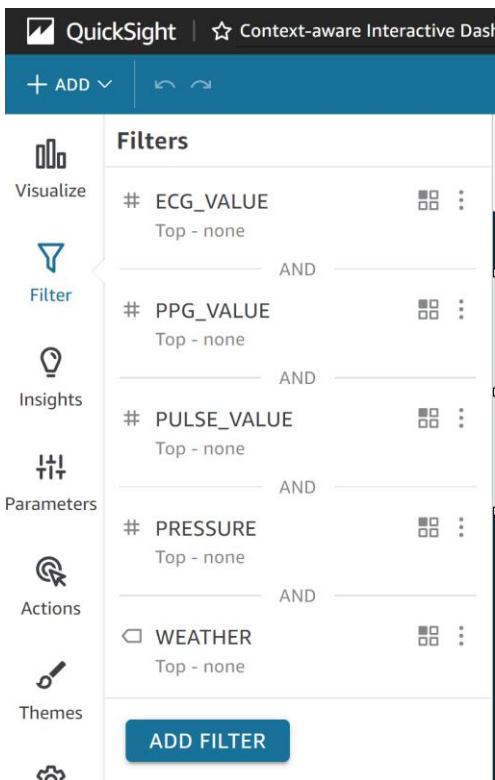


Figure 78. Organizing dashboard

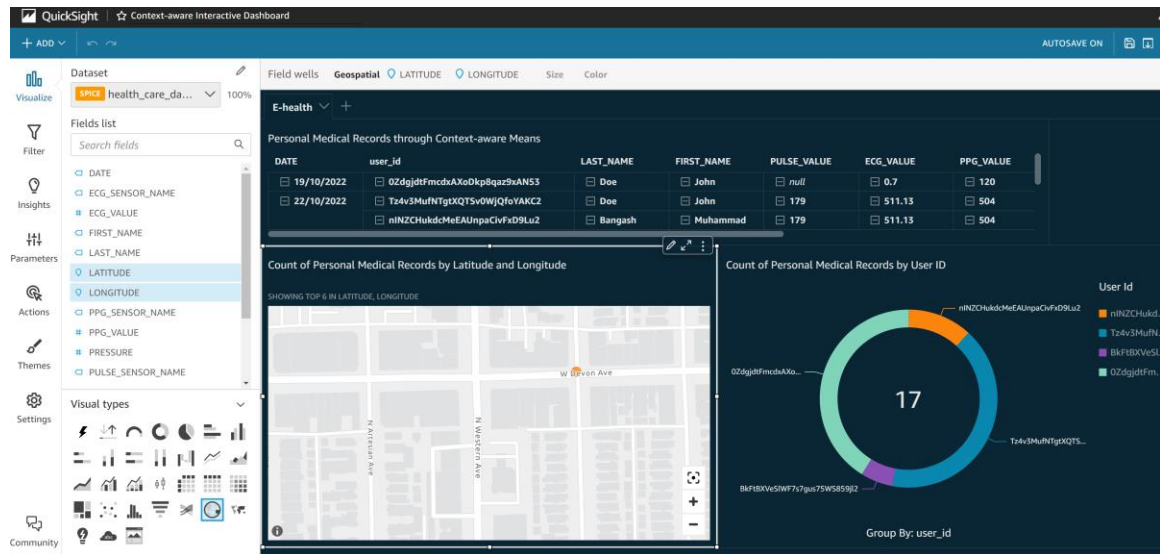


Figure 79. Displaying personal medical records

DATE	user_id	LAST_NAME	FIRST_NAME	PULSE_VALUE	ECG_VALUE	PPG_VALUE	PRESSURE	TEMPERATURE	WEATHER_DESC...	Count
19/10/2022	0ZdgjdtFmcdaAXoDkp8qaz9xAN53	Doe	John	null	0.7	120	1,014	2.45	few clouds	6
22/10/2022	Tz4v3MuftNtptXQTSv0WjQfoYAKC2	Doe	John	179	511.13	504	1,008	14.68	scattered clouds	2
	nINZCHukdcMeEAUupaCivFxD9Lu2	Bangash	Muhammad	179	511.13	504	1,008	14.68	scattered clouds	1
	0ZdgjdtFmcdaAXoDkp8qaz9xAN53	Doe	John	179	511.13	504	1,008	14.68	scattered clouds	1
	BKfBxVeSIWf7s7gus7SW5859j2	Bangash	Muhammad	48	515.06	493	1,003	14.14	heavy intensity ...	1
	Tz4v3MuftNtptXQTSv0WjQfoYAKC2	Doe	John	131	null	null	1,011	25.53	scattered clouds	1
					510.52	506	1,012	24.36	few clouds	3
				179	511.13	504	1,008	14.68	scattered clouds	1
	nINZCHukdcMeEAUupaCivFxD9Lu2	Bangash	Muhammad	179	511.13	504	1,008	14.68	scattered clouds	1

Figure 80. Displaying patient location in a real-time map

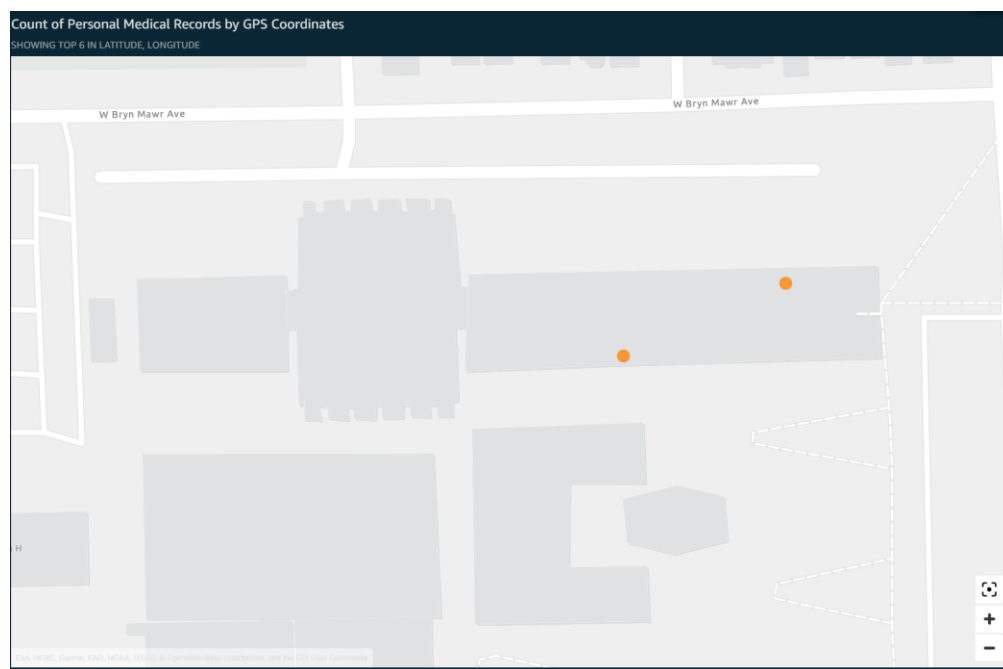


Figure 81. Displaying patient location in a real-time map

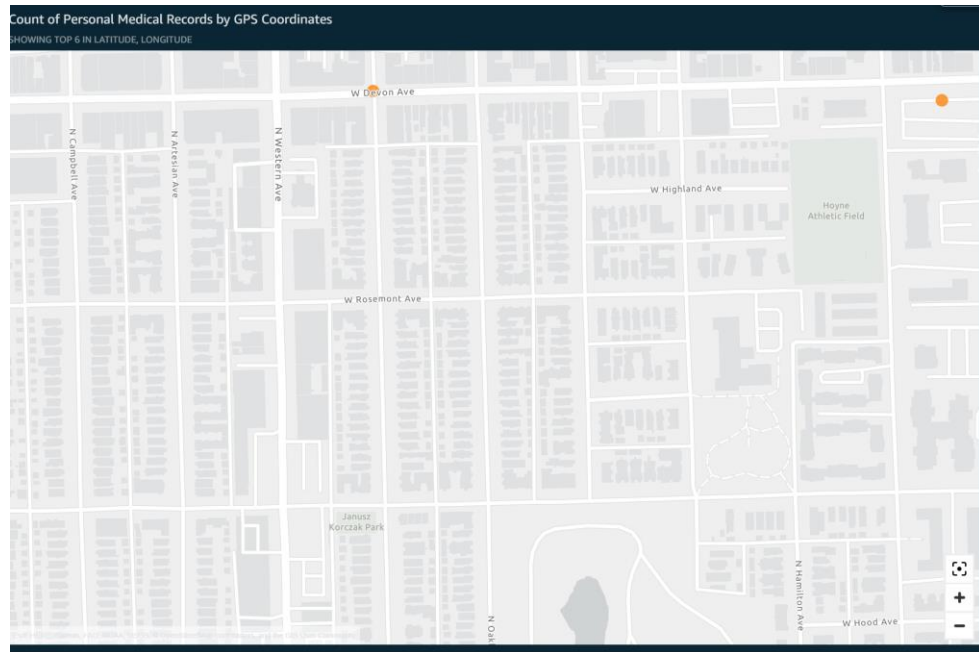
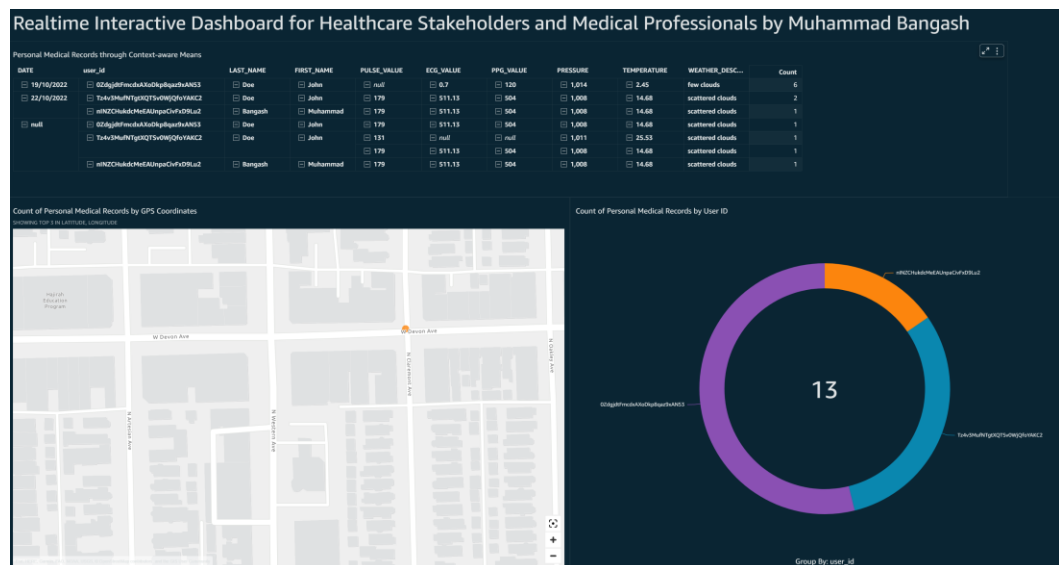


Figure 82. Real-time interactive dashboard for healthcare stakeholders and professionals

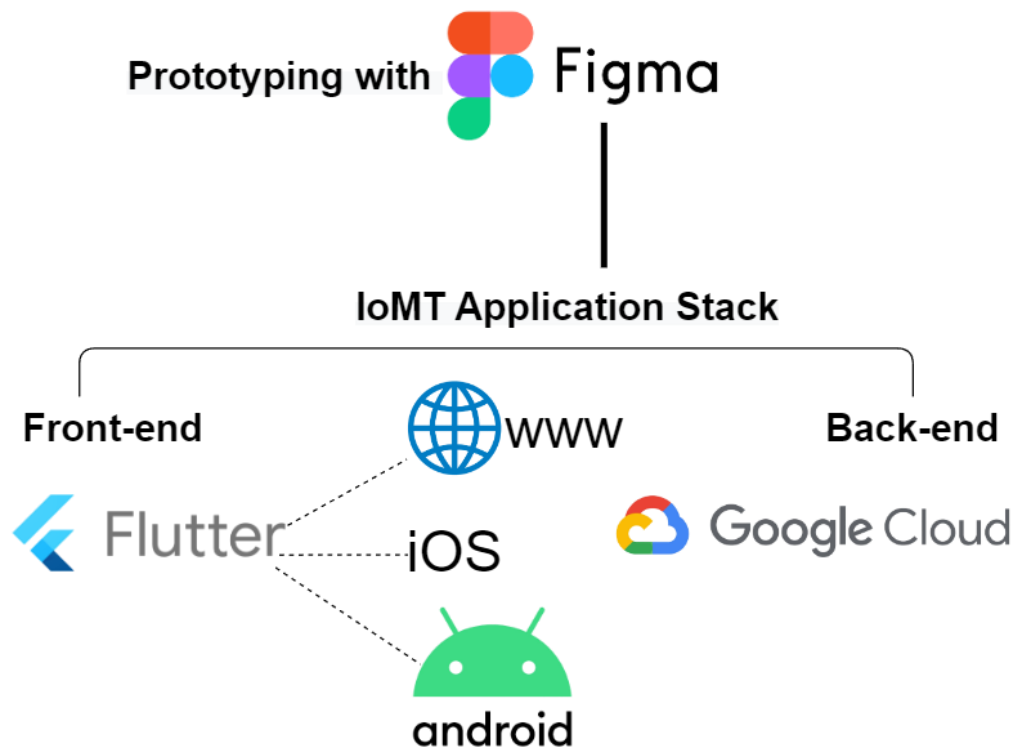


FUTURE WORK

One Single Code Base, Multiple Platforms

Currently, the IoMT application is developed for Android smartphones. In the future, as there is a need to expand with Apple iPhone users, a second application will need to be developed. That will be two code bases for developers to maintain which can be costly for projects and the organization. Through utilizing Flutter, a Google framework with one single code base, multiple platforms can be targeted (Fig. 83). For large-scale projects, enterprises should utilize Figma, a more popular tool for interactive designing. This tool recently was acquired from Adobe, at \$20 billion in cash and stock.

Figure 83. One Single Code Base, Multiple Platforms

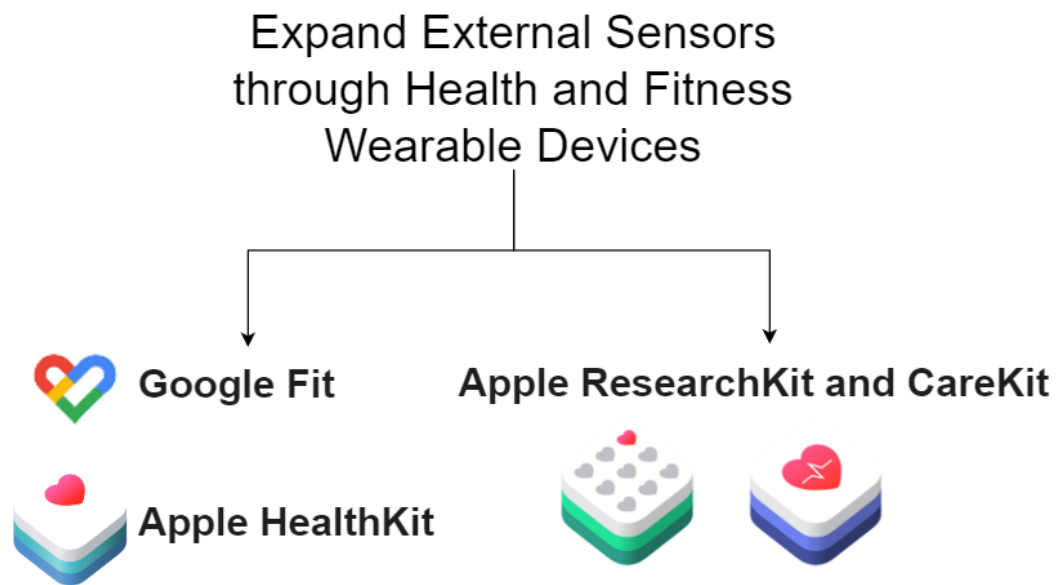


Note. Partial architecture retrieved from Flutter Festival GDG Montreal, Roman Jaquez, Google Cloud Certified Architect. Icons retrieved from Google LLC and Adobe. Figure created by Muhammad Bangash.

Smart Wearable Devices

In the interest of the project time, Google Fit was not implemented. However, its importance remains as the IoMT system is to be expanded for the current consumer market. Some of the popular consumer devices such as smartwatches and fitness devices in both the Android and Apple ecosystems can be leveraged through official APIs and services (Fig. 84).

Figure 84. Expanding outreach by communicating with smart wearable devices



Note. Icons retrieved from <https://developer.apple.com/health-fitness/> and <https://www.google.com/fit/>. Icons accessed 23 Oct. 2022.

Testing User and Data Accuracy

Throughout the project timeline, the testing has been done on me. As collecting health data from users need approvals and follows certain regulation, this part is crucial for adaptation and feedback. Feedback can be collected anonymously through surveys and ratings. Further iterations can then be applied to meet the patient's needs. Furthermore, testing rigorously the accuracy of vital signs with certified medical devices is important for upholding medical integrity and for a correct diagnosis.

Medical Notifications

In the interest of the project time, AWS SNS was not implemented. However, its importance remains as the IoMT system is to have communication from the medical professional to patients. SNS features message security from the encryption keys provided by AWS, application-to-application messaging, message durability, and application-to-person notifications.

Medical Portal

In the interest of the project time, AWS DynamoDB was not implemented. However, its importance remains as the IoMT system is to have a customized portal for doctors. Although AWS QuickSight was utilized to increase enterprise value, DynamoDB has its own benefits. A case study is as follows.

With the aid of radiopharmaceuticals and imaging agents, GE is well known for its medical imaging equipment (Rockset, 2019). By utilizing cloud access, storage, and computing, the business has employed DynamoDB to boost customer value. Healthcare practitioners in the US have access to a single gateway through the GE Health Cloud to process and distribute patient case images. This is a huge benefit for diagnosis. By having access to these medical records, clinicians can enhance their therapies (Rockset, 2019).

Creating Enterprise Value through AWS SageMaker

The PMR can be further experimented on to find outliers, anomalies, and patterns. By using QuickSight and AWS SageMaker, can apply machine learning models and natural language capabilities. Having more data insights can provide value in important decisions for healthcare stakeholders and institutions.

SUMMARY AND CONCLUSIONS

The project's aim to give researchers a simulation of a working multilayer IoMT system that preserves a patient's personal medical record and health timeline has been accomplished. Provided cloud and mobile solution architectures can be utilized as a baseline for further expandability, improvement, and development.

Here are the key project highlights. Developed a secure Android mobile application is developed to gather real-time user data through internal sensors, external medical sensors, and external services. I successfully created a personal medical record through context-aware (systematic) means. I bridged a secure communication to transfer records from the mobile application to the cloud MQTT broker. I hosted an encrypted middleware (MQTT broker) over the cloud. I utilized, optimized, and bridged an open-source MQTT broker library to securely communicate with cloud services. I configured a daily scheduler over the cloud to further process the medical records. Finally, I implemented a real-time interactive serverless BI dashboard that filters, analyzes, and displays the collected medical records. Thus, creating enterprise value for healthcare professionals and stakeholders.

Furthermore, there are some directions laid out on how mobile architecture can be further improved through a one-code base solution, targeting multiple platforms (web, iOS, Android). Consequently, expanding outreach by communicating with smart wearable devices (health and fitness). Moreover, implementing important medical notifications, a medical portal, machine learning, and natural language capabilities through existing cloud services.

BIBLIOGRAPHY

- Alam, M., Malik, H., Khan, M., Pardy, T., Kuusik, A., & Moullec, L. (2018). A Survey on the Roles of Communication Technologies in IoT Based Personalized Healthcare Applications. *IEEE Access*, *PP*, 1–1.
<https://doi.org/10.1109/ACCESS.2018.2853148>
- Amazon. (2022). *What is Amazon EC2? - Amazon Elastic Compute Cloud*. Amazon.com.
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html>
- Amazon LLC. (n.d.). *AWS IoT Analytics User Guide*. Retrieved October 25, 2022, from
<https://docs.aws.amazon.com/pdfs/iotanalytics/latest/userguide/iotanalytics-ug.pdf>
- Android Open Source Project. (2020). *Sensor types | Documentation*.
Source.android.com. <https://source.android.com/devices/sensors/sensor-types#proximity>
- Apple. (2022). *Health and Fitness - Apple Developer*. Apple Developer.
<https://developer.apple.com/health-fitness/>
- AWS. (2022). *Features and capabilities - Amazon Simple Notification Service*.
Amazon.com. <https://docs.aws.amazon.com/sns/latest/dg/welcome-features.html>
- Bush, T. (2019, November 12). *6 Best Free and Paid Weather APIs | Nordic APIs |*.
Nordic APIs. <https://nordicapis.com/6-best-free-and-paid-weather-apis/>
- Codd, E. F. (2001). A Relational Model of Data for Large Shared Data Banks. *Pioneers and Their Contributions to Software Engineering*, 61–98.
https://doi.org/10.1007/978-3-642-48354-7_4
- Eclipse Foundation. (2022). *Eclipse Paho | The Eclipse Foundation*. Eclipse.org.
<https://www.eclipse.org/paho/index.php?page=clients/android/index.php>

- Economic Strategy Institute. (2015). *Cloud Services will Expand US GDP, Jobs and Tech Spending*. Econstrat.org. <https://www.econstrat.org/research/the-new-ip-and-the-internet-of-things/638-cloud-services-till-expand-us-gdp-jobs-and-tech-spending>
- Evans, R. S. (2016). Electronic Health Records: Then, Now, and in the Future. *Yearbook of Medical Informatics*, 25(S 01), S48–S61. <https://doi.org/10.15265/iys-2016-s006>
- Figma. (2022). *Figma: the collaborative interface design tool*. Figma. <https://www.figma.com/>
- Gartner Identifies Top 10 Strategic IoT Technologies and Trends. (2018). Gartner. <https://www.gartner.com/en/newsroom/press-releases/2018-11-07-gartner-identifies-top-10-strategic-iot-technologies-and-trends>
- Gold, J., & Shaw, K. (2022, June). *What is edge computing and why does it matter?* Network World. <https://www.networkworld.com/article/3224893/what-is-edge-computing-and-how-it-s-changing-the-network.html>
- Google Cloud. (2022). *Firestore: NoSQL document database*. Google Cloud. <https://cloud.google.com/firestore>
- Google Developers. (2021). *Save data using SQLite*. Android Developers. <https://developer.android.com/training/data-storage/sqlite>
- Google Developers. (2022a). *Firebase Realtime Database*. Firebase. <https://firebase.google.com/docs/database/web/start#:~:text=You%20can%20find%20your%20Realtime,for%20databases%20in%20us%2Dcentral1%20>
- Google Developers. (2022b). *Geocoder / Android*. Android Developers Documentation. <https://developer.android.com/reference/android/location/Geocoder>

- Google Developers. (2022c). *Motion sensors*. Android Developers.
https://developer.android.com/guide/topics/sensors/sensors_motion
- Google Fit. (2022). *Google Fit*. Google Fit; Google Fit. <https://www.google.com/fit/>
- Google LLC. (2021). *Environment sensors*. Android Developers.
https://developer.android.com/guide/topics/sensors/sensors_environment
- Google LLC. (2022a). *Firebase Realtime Database*. Firebase.
<https://firebase.google.com/docs/database>
- Google LLC. (2022b). *Google Cloud Adoption Framework*. Google Cloud.
<https://cloud.google.com/adoption-framework>
- Google LLC. (2022c). *Type Temperature Sensor | Android Developers*. Android Developers.
https://developer.android.com/reference/android/hardware/Sensor#TYPE_TEMPERATURE%20Edit%20API%20docs%20paths
- Google Project Management*: (2021). Coursera. <https://www.coursera.org/professional-certificates/google-project-management>
- hannesa2. (2022, October 24). *GitHub - hannesa2/paho.mqtt.android: Kotlin MQTT Android with almost all pull requests from upstream*. GitHub.
<https://github.com/hannesa2/paho.mqtt.android>
- Harahap, Nabila Clydea, Handayani, Putu Wuri, & Hidayanto, Achmad Nizar. (2021). Functionalities and Issues in the Implementation of Personal Health Records: Systematic Review. *Journal of Medical Internet Research*, 23(7), e26236–e26236. PubMed. <https://doi.org/10.2196/26236>

- Harrington, C. N., Lyndsie Marie Koon, & Rogers, W. A. (2020). *Chapter 17 - Design of health information and communication technologies for older adults* (Arathi Sethumadhavan & Farzan Sasangohar, Eds.; pp. 341–363). Academic Press.
<https://doi.org/https://doi.org/10.1016/B978-0-12-816427-3.00017-8>
- Healthy Ads. (2021, September 19). *Latitude and Longitude Targeting*. Healthy Ads.
<https://www.healthyads.com/targeting/latitude-and-longitude-targeting/>
- HeartBIT*. (2022). Plux Wireless Biosignals.
<https://www.pluxbiosignals.com/collections/bitalino/products/heartbit>
- hivemq. (2022, October 7). *hivemq/hivemq-community-edition: HiveMQ CE is a Java-based open source MQTT broker that fully supports MQTT 3.x and MQTT 5. It is the foundation of the HiveMQ Enterprise Connectivity and Messaging Platform*. GitHub. <https://github.com/hivemq/hivemq-community-edition>
- IJCSMC Journal, & Hasan, H. (2018). IoT Protocols for Health Care Systems: A Comparative Study. *IJCSMC*.
https://www.academia.edu/37772860/IoT_Protocols_for_Health_Care_Systems_A_Comparative_Study
- Jaquez, R. (2022). *Flutter Festival GDG Montreal*. Google Developer Groups.
<https://gdg.community.dev/events/details/google-gdg-montreal-presents-flutter-festival-montreal-online/>
- Jashapara, N. (2021, June 2). *AMAZON AND SURVEILLANCE CAPITALISM: DO WE WANT A NETFLIX FOR BOOKS?* Bad Form.
<https://www.badformreview.com/read/rfka>

- Kelly, D. (2020, October 29). *A Brief History of Databases*. Cockroach Labs.
<https://www.cockroachlabs.com/blog/history-of-databases-distributed-sql/>
- Kim, G. J. (2015). *Human-Computer Interaction*. Auerbach Publications.
<https://doi.org/10.1201/b18071>
- Masykur, F., Prasetyo, A., Widaningrum, I., Cobantoro, A. F., & Setyawan, M. B. (2020, September 1). *Application of Message Queuing Telemetry Transport (MQTT) Protocol in the Internet of Things to Monitor Mushroom Cultivation*. IEEE Xplore. <https://doi.org/10.1109/ICITACEE50144.2020.9239118>
- MongoDB. (2021). *Documents*. Mongoddb.com.
<https://docs.mongodb.com/manual/core/document/>
- National Geographic Society. (2010, November 11). *Introduction to Latitude and Longitude*. National Geographic Society.
<https://www.nationalgeographic.org/activity/introduction-latitude-longitude/>
- Office of the National Coordinator for Health Information. (2019, January). “*Office-based Physician Electronic Health Record Adoption*,” *Health IT Quick-Stat #50*. Healthit.gov. <https://www.healthit.gov/data/quickstats/office-based-physician-electronic-health-record-adoption>
- OpenWeatherMap. (2017). *Weather API - OpenWeatherMap*. Openweathermap.org.
<https://openweathermap.org/api>
- Perakslis, E. D., & Huang, E. (2020, March 12). *Covid-19 will be the ultimate stress test for electronic health record systems*. STAT.
<https://www.statnews.com/2020/03/12/covid-19-huge-stress-test-electronic-health-record-systems/>

- pluxbiosignals. (2021, December 27). *Official PLUX Application Programming Interfaces (APIs) – Support PLUX Biosignals official*. Pluxbiosignals.com. <https://support.pluxbiosignals.com/knowledge-base/official-plux-application-programming-interfaces-apis/>
- pluxbiosignals. (2022). *BITalino (r)evolution Plugged Kit BLE/BT*. Plux Wireless Biosignals. <https://www.pluxbiosignals.com/collections/bitalino>
- Rockset. (2019). *5 Use Cases for DynamoDB*. Rockset.com. <https://rockset.com/blog/5-use-cases-for-dynamodb/#:~:text=GE%20Healthcare&text=The%20company%20has%20used%20DynamoDB,a%20great%20advantage%20for%20diagnostics.>
- Roshina, D. (2022, October 4). *SQLite vs Realm database for Android apps*. Cleveroad Inc. - Web and App Development Company; Cleveroad Inc. <https://www.cleveroad.com/blog/realm-vs-sqlite-what-is-the-best-database-for-android-app-development/>
- Salesforce Trailhead. (2022). *API Basics*. Salesforce.com. <https://trailhead.salesforce.com/content/learn/modules/pw-api-basics/make-apis-for-you-and-me>
- Saminathan, S., & Geetha, K. (2018). Real-Time Health Care Monitoring System using IoT. *International Journal of Engineering & Technology*, 7(2.24), 484. <https://doi.org/10.14419/ijet.v7i2.24.12141>
- Schaefer, L. (2020, March 18). *Mapping Terms and Concepts from SQL to MongoDB*. Mongoddb.com; MongoDB. <https://www.mongodb.com/developer/article/map-terms-concepts-sql-mongodb/#std-label-sql-mdb-1-document-model>

Schobel, J., Schickler, M., Pryss, R., Nienhaus, H., & Reichert, M. (2013). Using Vital Sensors in Mobile Healthcare Business Applications: Challenges, Examples, Lessons Learned. In *WEBIST 2013 Proceedings of the 9th International Conference on Web Information Systems and Technologies*.

Shams. (2016, September 12). *A few Android Phones that have Temperature Sensor*.

WebCusp.com - Make Living with Blogging and No Coding Web Design.

[https://webcusp.com/a-few-android-phones-that-have-temperature-](https://webcusp.com/a-few-android-phones-that-have-temperature-sensor/#:~:text=Almost%20every%20device%20has%20an,and%20battery%20temperature%20of%20device.&text=These%20directions%20are%20there%20so,aaccurate%20idea%20of%20ambient%20temperature)

[sensor/#:~:text=Almost%20every%20device%20has%20an,and%20battery%20temperature%20of%20device.&text=These%20directions%20are%20there%20so,aaccurate%20idea%20of%20ambient%20temperature](https://webcusp.com/a-few-android-phones-that-have-temperature-sensor/#:~:text=Almost%20every%20device%20has%20an,and%20battery%20temperature%20of%20device.&text=These%20directions%20are%20there%20so,aaccurate%20idea%20of%20ambient%20temperature).

Silva, A. F., & Tavakoli, M. (2020). Domiciliary hospitalization through wearable biomonitoring patches: Recent advances, technical challenges, and the relation to covid-19. *Sensors*, 20, Article 23. <https://doi.org/10.3390/s20236835>

The Internet of Things Connects Customers to Their World. (2019). Salesforce.com.

<https://www.salesforce.com/products/platform/best-practices/internet-of-things-connects-customers/>

Tsao, Y.-C., Cheng, F.-J., Li, Y.-H., & Liao, L.-D. (2022). An IoT-Based Smart System with an MQTT Broker for Individual Patient Vital Sign Monitoring in Potential Emergency or Prehospital Applications. *Emergency Medicine International*, 2022, e7245650. <https://doi.org/10.1155/2022/7245650>

User Story Templates in Agile. (2015, December 17). Agile Alliance |; Agile Alliance.

<https://www.agilealliance.org/glossary/user-story-template/>

- Verma, N., Singh, S., & Prasad, D. (2022). A review on existing IoT architecture and communication protocols used in healthcare monitoring system. *Journal of the Institution of Engineers (India): Series B*, 103, 245–257.
<https://doi.org/10.1007/s40031-021-00632-3>
- Wickramasinghe, S. (2021, July 14). *MongoDB vs DynamoDB: Comparing NoSQL Databases*. BMC Blogs. <https://www.bmc.com/blogs/mongodb-vs-dynamodb/>
- Wu, F., Wu, T., & Yuce, M. R. (2018). An internet-of-things (IoT) network system for connected safety and health monitoring applications. *Sensors*, 19(1), 21.