National College of Ireland

Optimized Latency in IoT Healthcare: Edge-Driven System with Fog and Cloud Support

Research Project
MSc Cloud Computing

Manjula Kore
Student ID: x23203986

School of Computing
National College of Ireland

Supervisor: Diego Lugones

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Manjula Kore
Student ID:	x23203986
Programme:	MSc Cloud Computing
Year:	2024
Module:	Research Project
Supervisor:	Diego Lugones
Submission Due Date:	12/12/24
Project Title:	Optimized Latency in IoT Healthcare: Edge-Driven System with Fog and Cloud Support
Word Count:	6450
Page Count:	21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Manjula Kore
Date:	28th January 2025

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Optimized Latency in IoT Healthcare: Edge-Driven System with Fog and Cloud Support

Manjula Kore
x23203986

Abstract

It is necessary that IoT is incorporated in patients' care, being essential in developing nations where access to healthcare facilities is limited. Non-invasive diagnostic devices, like pulse oximeters and ECGs, depend on cloud data processing to compute the vital parameters, which causes latency threatening to provide immediate medical attention in emergency conditions including the COVID-19 pandemic. This work integrates fog computing into a basic cloud IoT architecture model with the help of AWS IoT Greengrass for edge computing and AWS Lambda for cloud processing. The fact that such important indicators are filtered at the local level minimizes traffic and delays in sharing the necessary data. Through cohort studies, patient record files of 100 to 500 were used to determine the latency performance with less than 10–20% degradation in throughput latency when handling large files. These results support fog computing as a solution to latency problems and show promising signs for using it to make precise, real-time care decisions in areas of low bandwidth such as rural regions or ICUs. Further, the least reliance on ongoing cloud contact increases the system's robustness in regions with limited internet access, guaranteeing regular surveillance and immediate exception recognition. Such an approach shows that edge computing can transform healthcare systems where technologies are lacking by filling the gap.

Keywords: Latency Reduction, Scalability, Accuracy, Cloud Services, AWS Greengrass Core.

1 Introduction

1.1 Background

The implementation of IoT and big data has been accelerated to bring further innovation and improvement to the healthcare sector through continuous monitoring and real time decision making. Body-worn vests, smartwatches, fitness trackers, and smart clothes have emerged as critical technologies for capturing critical health information. Such developments are most helpful in rural and hard to reach populations locally, which is usually characterized by scarcity of facilities and personnel. However, the processing and storage of this data in the centralized cloud systems limit latency and bandwidth, which can hamper timely interventions in confined emergencies. These need to be tackled innovatively in a way that focuses on decentralizing the data processing task while also ensuring system scalability in a way that keeps healthcare systems relevant regardless of the environment in which they are deployed. Smys and Raj (2019)

1.2 Motivation

The major issue with cloud-based IoT healthcare systems is latency, especially if the unit is in a remote or developing area with little bandwidth. These areas experience limited or no connectivity and, as such cannot easily upload essential healthcare data to the cloud in order to facilitate real-time decision-making. In all patient conditions that are severe and may need immediate attention like chronic disease checkups or in cases of medical emergencies, latency cannot be afforded. In addition, volume becomes an even bigger issue given high levels of healthcare data processing and analysis, thus making scalability a big factor. To address these challenges, this research propose the use of fog computing to IoT healthcare systems whereby data processing is performed near the edge of the network. This integration of fog computing with big data analytics allows healthcare professionals to process data locally and in real-time.(Mutlag et al. (2019))

1.3 Research Question

IoT healthcare systems including rural or remote areas, generally suffer latency issues due to minimized bandwidth scenes in cloud structure. These delays happen in various steps such as collection of data, transmission of data, processing of data, and generation of reports which are standalone crucial in healthcare emergencies. Most conventional cloud applications merely transmit data to servers that are elsewhere and then back, which in the process creates latency that complicates real-time decision-making.

"How does integrating edge computing into IoT healthcare systems reduce latency and improve response times compared to traditional cloud-based models?"

1.4 Objective

This research aims to investigate the impacts of fog computing on implementing real-time IoT healthcare systems with minimum latency in rural/low bandwidth scenarios. This work therefore seeks to establish a model for an IoT health care system that employs real-time sensor data premium health indices which include the rate of pulse, ECG, and cholesterol. To minimize the use of cloud support (Gupta et al. (2021)), the research proposes to apply technologies for data processing localized in specific regions. Moreover, the cloud functions will be used to process the data, which is to be analyzed, in addition to the local data processing done through functions in the IoT edges. The study will also measure the latency at the different levels of the fog, as well as spotlight the efficiency and responsiveness gains of the fog computing model as compared to the cloud-only solution. Finally, this research will evaluate how optimization of fog computing shall enhance the timeliness and effectiveness of healthcare applications in order to develop right decisions as well as the respective treatment. Figure 1 illustrates a general view of the system plan.

1.5 Paper Structure

The Paper starts with the **abstract** where research problems, methods, findings and the research's contribution are outlined. The **introduction** then proceeds to present the

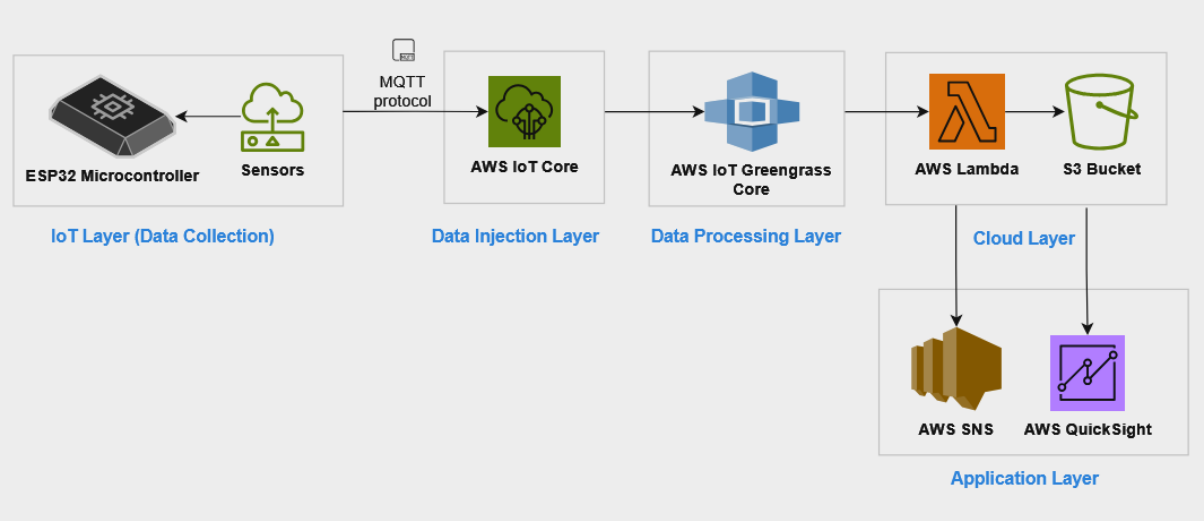

Figure 1: System Flow

research problem, motivation, research questions, objectives, and a brief paper roadmap. This paper critically surveys **related work** on IoT based healthcare systems, cloud computing, fog computing, and methods to reduce latency, which are addressed by this work. In the **research methodology**, the simulation tools and methods of data, system architecture, and latency measurement tools are described. **Design Specification** describes the implementation of fog computing through AWS IoT Greengrass for processing at the edge and AWS Lambda for processing at the cloud. The latency comparisons of the fog-based system and a plain cloud system are made and discussed in the **Evaluation**. Last, the **Conclusion and Future Work** synthesizes the significance of healthcare IoT applications and provides potential research directions to improve real-time management and decision-making in healthcare IoT scenarios.

2 Related Work

The literature review for this study looks at **IoT healthcare systems**, **big data analytics**, and **fog computing** with emphasis made on latency issues and possibilities of edge computing in solving the issues. Based on this analysis, this review aggregates papers focusing on IoT for healthcare, cloud and fog computing systems, and latency minimization. This review assesses the performance of the present techniques used in the shade of the advantages and limitations, thus enabling readers to better understand how they may be used effectively.

2.1 IoT in Healthcare

One of the most rapidly growing fields that practically applies IoT concepts is healthcare, for instance, IoT technology for distant oversight of a specific group of patients and management of chronic diseases. John Dian et al. (2020) noted that smart devices and sensors are implemented in the management of patient care in health facilities to track general body info such as heart rate, temperatures, and physical activity. However, these systems do suffer from latencies occasional, and Bharath and Merlin Sheeba (2024) indic-

ate that cloud-dependent systems add delays of between 200-500 ms which are unsuitable for emergencies such as cardiac arrest. Smys and Raj (2019) in their work, Latency consideration for Non-Real-Time Systems, pointed out that certain latency could slow down important real-time processes like raising alarms or informing physicians. On the other hand, fog computing has been seen to achieve improvement in these delays and they range between 50 to 70 ms as stated by Mahmud et al. (2018) Similarly, Bharath and Merlin Sheeba (2024) also pointed out the problem of the local computational capability to make IoT cyber-physical scalable, especially for rural IoT where IoT is expected to exceed 1 TB of data where the real-time operations remains critical.

2.2 Cloud Computing in Healthcare

There is a tremendous volume of data produced by IoT devices in healthcare centers and this calls for cloud computing for storage management and analysis. Smys and Raj (2019) theories assert that therefore. Cloud computing provides the required environment for such jobs but comes with the band— width limitations and signal delays with the transmission latency being in the range, of 300 ms and above for real-time scenarios. In their study, Firouzi et al. (2018) also support these observations, stressing the cloud as an advantage of computing and emphasize its analytical power but pointing at the same time out learner reappearance in low latency demands earlier to address the different latency requirements of important healthcare applications. A cross-sectional descriptive study was done by Tyagi et al. (2016), paradigms of efficiencies in IoT-cloud systems especially concerning datasets over 1 TB in size but claim that latencies within the range of 50-100 ms are required with the help of fog computing. for the daily, realistic provision of healthcare.

2.3 Fog Computing for Latency Reduction

The utilization of what is called fog computing to cater to distributed computing at the edge is a perspective solution to the problem of latency in IoT-related systems. In so doing, it minimizes the use of the cloud by close processing of data, in accordance with Smys and Raj (2019) hence applicable in healthcare models since response times are critical. While cloud systems have latencies of 300 to 500 ms, fog computing has 50 to 70 ms as identified by Mahmud et al. (2018). Furthermore, Mutlag et al. (2019) concluded that increased effectiveness of fog-based architectures can cut down response time by 40 percent, improving real-time health care efficiency. Connected syndicates of fog and cloud computing also maintain scalability and can handle datasets over 1 TB or serve thousand IoT devices. However, some challenges have been seen to originate from the wide adoption the fog computing. As shown in Table 3, several challenges arose as discussed by Qiu et al. (2020), including resource allocation and energy consumption; where fog nodes consume thrice or twice the energy of a centralized cloud system. Further, the implementation of fog computing inevitably encounters irregularity of performance distributed over various nodes which governmental by sophisticated resource management and orchestration. Such challenges highlight the significance of the development of inventive mechanisms for enhancing fog computing for further healthcare applications.

2.4 Big Data Analytics in HealtCare

Currently, IoT assists in producing big data in healthcare and management of the devices requires big data analytics for decision making. Rathore et al. (2016) have also shown that by using big data analytics in IoT applications, it is possible to achieve greater than 95% accurate models while addressing real-time issues. Similarly, Manogaran et al. (2017) expanded these applications to individual health plans, and the detection of complicated diseases with high accuracy results in the management of chronic diseases. However, their reliance on these cloud-based systems created access and processing latencies between 300 and 500 ms which has the effect of diminishing the value of the insights in real emergency situations.

In response to these challenges, Smys and Raj (2019) discussed the demerits of cloud-based processing and they explained that fog computing can improve big data processing since its processing time is faster. Malik and Om (2018) supported this by showing that implementation of fog computing in combination with big data analytics decreased the amount of time required for data processing to between 50-70 ms giving nearly real-time results. Furthermore, fog computing cut data exchange to cloud infrastructure by 40%, enhances the availability of bandwidth frontier, and less dependence on the core systems. These results clearly indicate the feasibility of fog-based architectures to close the big data analytics loop with real-time efficient healthcare programs.

2.5 Integration of Fog Computing with IoT and Big Data Analytics

The integration of fog computing with the IoT and big data the analytic process is the other advantage that offers a disruptive way of removing lag time and enhancing processing. According to Smys and Raj (2019), fog computing minimizes latency since most of the data is analyzed at the place it is generated, therefore, minimizes data transfer time to central systems. Cloud-based systems have delays of 300-500 ms, for fog-based systems, the delays range between 50-70 ms, which is more appropriate for time-sensitive health care. Mahmud et al. (2018) developed a fog-cloud integration model that incorporates the characteristics of cloud solutions and edge solutions. As realized in this hybrid approach, it offers a real-time capability of supporting RPM or any RPM-like application with the response time minimized by 40% even within limited resources.

Based on their experiments, Nandyala and Kim (2016) underlined that fog computing can be used in real-time health monitoring for smart homes and hospitals with most of the operations taking less than 100ms of latency. Their work sums up well with the work done by Mutlag et al. (2019) where they proved that the inclusion of fog nodes with IoT improves data security. By decentralizing the flows, the amount of data transmitted through the central servers is cut by 30-40%, which in turn means that the patient-sensitive information is less exposed in the course of transmission. Overall, the current literature in this area underscores the future prospects of fog computing in handling the issues of latency, scalability, and security; which makes the latter an enabling element in real-time health care.

2.6 Synthesis and Gaps in Literature

The highlighted literature proves the capability of IoT healthcare systems to be used for predictive analysis and monitoring patient's conditions. However, architectures in cloud have latency issues with values getting higher than 300-500 ms which makes them unfit for time-constrained applications (Smys and Raj (2019); Bharath and Merlin Sheeba (2024)). It has been found that fog computing may provide a solution where latencies are 50-70 ms and where the cloud reliance is decreased by 30-40% (Mahmud et al. (2018); Mutlag et al. (2019)).

However, important gaps remain regarding scalability, utilization of resources, and practical application in rural or low bandwidth environments. Although fog-cloud hybrid approaches deal with latency and scalability issues, their practical application is still problematic. Based on this, the present study proposes an empirical investigation of a fog computing system to address the challenges of low-resource contexts, especially in IoT-supported healthcare applications.

The table below outlines the research gaps identified across the papers: Table: 1

Research Area	**Key Insights**	**Research Gaps**	**Authors**
IoT in Healthcare	Real-time monitoring, chronic disease management.	Lack of real-world data, limited focus on latency, especially in emergency care.	John Dian et al. (2020), Bharath and Merlin Sheeba (2024)
Cloud Computing in Healthcare	Scalable data storage and analytics.	High delays (300-500 ms), inefficient for real-time decisions, especially in rural areas.	Smys and Raj (2019) Firouzi et al. (2018), Tyagi et al. (2016)
Fog Computing for Latency Reduction	Processes data locally, reducing cloud dependency.	Limited healthcare applications, scalability issues, and resource optimization challenges.	Mahmud et al. (2018), Mutlag et al. (2019), Nandyala and Kim (2016)
Big Data Analytics in Healthcare	Enables insights for prediction and personalized care.	Delays in cloud-based processing affect real-time decision-making, especially in emergencies.	Rathore et al. (2016), Manogaran et al. (2017)
Fog Computing with IoT + Big Data Analytics	Combines edge processing and analytics for real-time insights.	Lack of real-world validation, challenges in integrating fog computing with IoT and big data.	Mahmud et al. (2018), Nandyala and Kim (2016), Mutlag et al. (2019), Qiu et al. (2020), Smys and Raj (2019)

Table 1: Comparison of Key Research Areas in IoT Healthcare

3 Methodology

This methodology defines step by step how sensors will integrate, collect data, transmit data, process data on the edge, and finally visualize data.

3.1 Research Design Framework

This study's main objective was to develop a healthcare monitoring system using IoT whereby patients' health information could be collected, stored, and transmitted in real time. The design followed the CRISP-DM (Cross Industry Standard Process for Data Mining) framework, which includes six phases: Business Understanding, Data Understanding and Acquisition, Data Preparation, Modeling, Evaluation, and Deployment. The methodology focused on three core aspects: low latency, identification of abnormal operations, and system expansibility. The system was built and validated using Wokwi, an ESP32 electronics circuit simulator, and the AWS cloud platforms. In order to provide a high degree of external validity, the methodology utilized in the study was intentionally made replicable by other researchers using virtual tools, and was, therefore, affordable.

3.2 Virtual Simulation Environment

Due to financial constraints and the need for extensive experimentation for scalability, this research work employed the use of the Wokwi simulation environment for the ESP32 microcontroller and the attachment sensors. This made it possible to conduct edge computing, data transfer, and cloud computing without any actual hardware implementation of the system. This approach was time effective and reduced the use of real devices during the experiments, it also made it easy for others to repeat the experiments since they took note of the setup.

Fig. 2 and Fig. 3 show the overview of virtual simulation and sensor integration with data transmission.

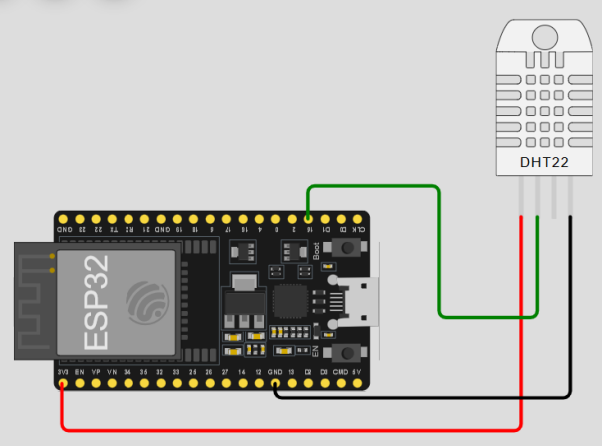

Figure 2: ESP32 Microcontroller

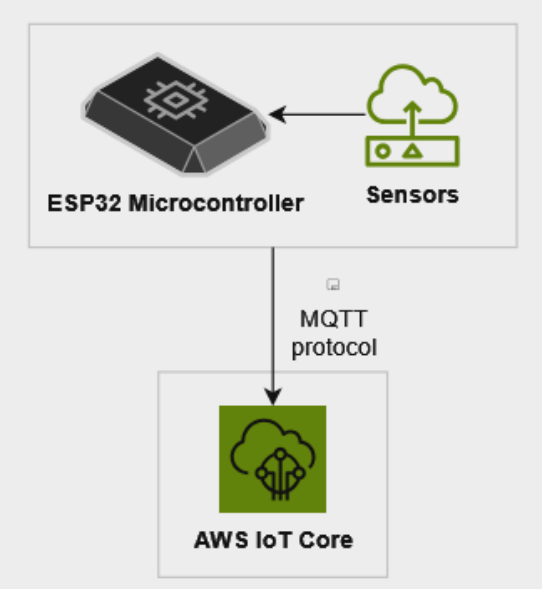

Figure 3: Data Transfer

3.3 Data Collection and Sensor Integration

For simulating sensor data, temperature data was used in DHT22 format, and ECG, EMG, blood pressure data, and blood glucose data were all incorporated into the system

to mimic real-time health checks. These sensors were selected based on factors like the capability they offer real-time data of a patient's status, and cardiovascular and chronic disease metrics. The data sort was obtained in real-time and then sent to the ESP32 simulator on Wokwi which portrays real sensors. Accomplishing real-life scenarios made it possible to simulate pathophysiological alterations to prove the system's capacity to respond to different types of failures.

3.4 Data Transmission and Secure Cloud Communication

After the detection of the sensor data was done at the local level, the data was published to AWS IoT Thing using MQTT protocol with TLS security. MQTT was chosen due to its low overhead and its performance in applications with mercurial interconnectivity, typical of IoT healthcare. Every data packet that includes important sensor data such as blood pressure, glucose level, and the like was safely transmitted to the cloud. It also complied with HIPAA rules to minimize privacy and loyalty of information.

3.5 Edge Processing Using AWS Greengrass

In this healthcare monitoring system, AWS Greengrass was integrated to enable real-time processing and local decision-making directly on edge devices, represented in the diagram as the ESP32 microcontroller functioning as the Greengrass Core. Sensor data for critical parameters like blood pressure, ECG, glucose levels, and more is published by the Publisher Device to the MQTT topic on Greengrass Core (middle section of the diagram). Greengrass then triggers Lambda Edge functions hosted on the Core, where the YAML sensor data is transformed into JSON and processed locally. This allows for fast anomaly detection e.g., blood pressure exceeding 140/90 mmHg or glucose levels dropping below 70mg/dL resulting in instant alert generation without waiting for cloud-based analysis. The Message Queue (left section) only receives critical alerts and metadata, reducing bandwidth usage by avoiding raw sensor data transmission. Additionally, local logging within Greengrass ensures timestamps and event information are recorded, enabling continued functionality during network outages. This architecture leverages the Greengrass Core to achieve edge-first processing while utilizing the cloud for backups, delivering a responsive, secure, and bandwidth-conscious solution that minimizes latency and improves real-time healthcare monitoring efficiency, as depicted in the Fig. 4.

3.6 Data Storage, Analysis, and Visualization

Processed data was stored in Amazon S3 in JSON format, with three distinct files: raw signals from the sensors, it's transformed and analyzed form which is the anomalies, and the occurrence of an alert on the method. Thus, this segregation made it possible to exercise control and independent verification. AWS CloudWatch was used for real-time visualization and control of a variety of numbers regarding patient status, for example, blood pressure, glucose levels, and ECG rhythms to enable healthcare practitioners to quickly identify changes from the norm. These observations were depicted in the form of graphs and dashboards to serve both Short-Term Data Analysis and Long-Term Patient Management.

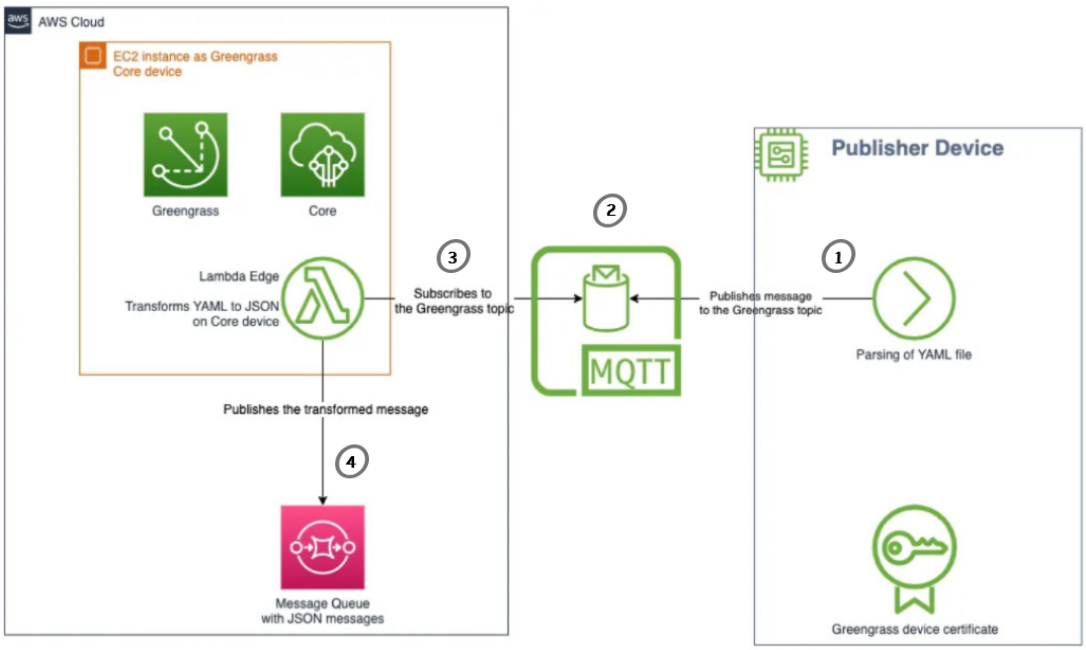

Figure 4: Greengrass Architecture

3.7 Anomaly Detection and Alerts

Due to the fact that health emergencies require a quick response, AWS SNS (Simple Notification Service) was added to the system to make provision for real-time alerts. For the purpose of anomaly detection, the threshold was established for each sensor concerning blood pressure, glucose levels, ECG rhythms, and several others. These thresholds have been tuned by the local processing component AWS Greengrass to enable real-time analysis of the sensors' data. Each time an abnormal state was identified including any abnormality detected in an ECG, AWS Greengrass triggered an alert. Fig. 5 shows the components involved in sending notifications to the end user.

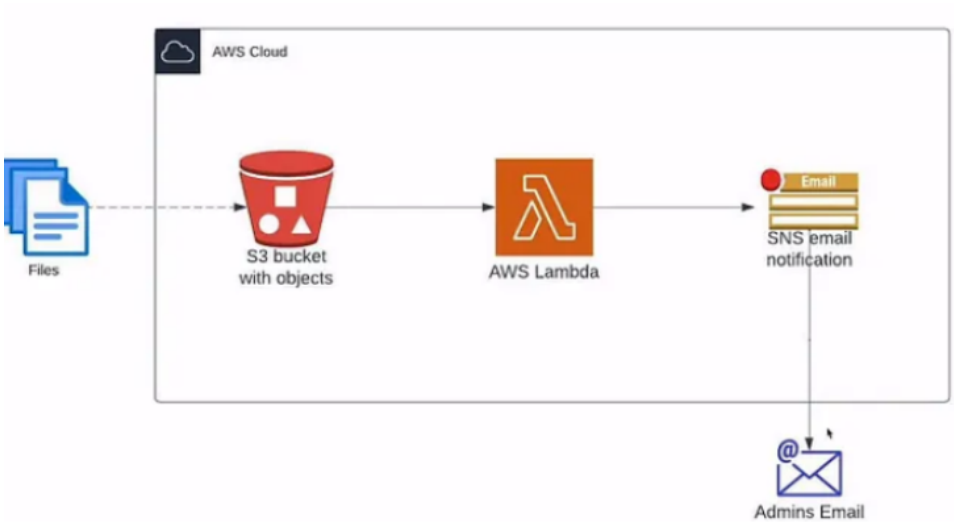

Figure 5: Notifications using AWS SNS

Fig.6 shows the threshold level set for the sensors in monitoring the patient's health

to perform refining of the data using the Greengrass written in lambda function.

Sensor Measurement	Verge level of the sensors	Condition of the Patient	
		Serious	Not Serious
Temperature	Less than 98.3F – low	serious	
	98.3F-normal		Not serious
	More than 98.3F-high	Serious	
Blood Pressure	< 60-low	-	-
	60-140 –moderate		Not serious
	>140 high	Serious	
Heart Rate	< 60-low	-	-
	60- 100 moderate		Not serious
	> 100 –high	Serious	
Sugar Level	<70 –low	-	-
	70-120-moderate		Not serious
	>120-high	Serious	
Movement	40% movement normal		Not serious
	20% needs doctor advice	Serious	
	<20% needs readmission	Serious	

Figure 6: Threshold Levels of Sensors

4 Design Specification

4.1 System Components and Architecture

IoT starts with temperature sensors, glucometer, electromyogram, electrocardiogram, pulse rate, and humidity, which works to gather and send the health records in real life through ESP32 Microcontroller using the MQTT technique. The data is then transmitted to the AWS IoT Greengrass core device where processing of the data collected takes place including detection of anomalies such as high blood pressure or high blood sugar level and data compilation. Below Fig. 7 shows the Architecture Diagram:

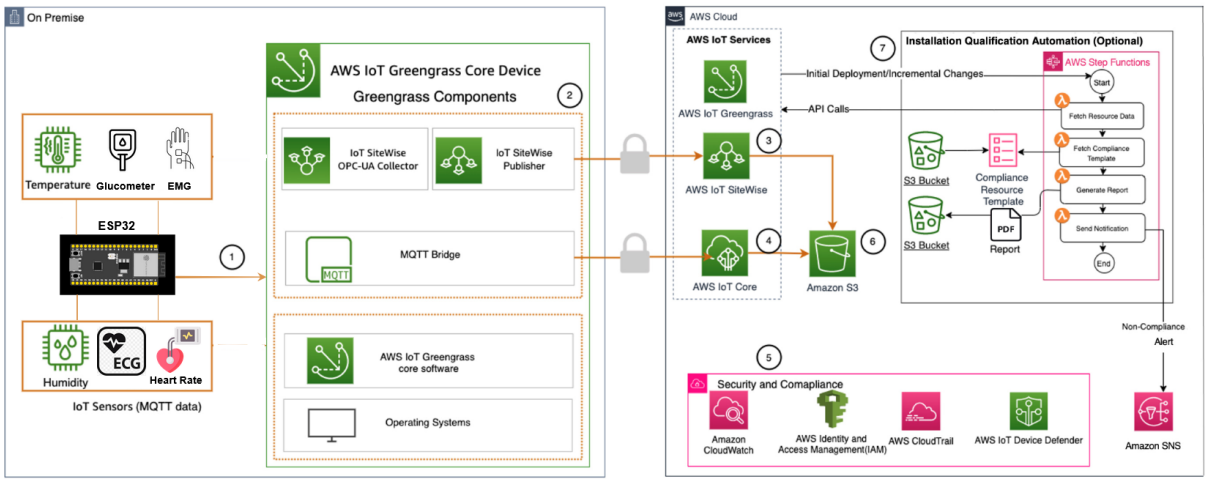

Figure 7: Architecture Diagram

At the edge, Greengrass components, IOT SiteWise OPC-UA Collector, Publisher, and MQTT Bridge are responsible for real-time preprocessing and interfacing between the edge and cloud. Such deviations result in an alert being raised locally for a swift response to the problem detected. The processed data is then transferred to AWS IoT Core which in turn directs it to other AWS IoT AWS IoT SiteWise, for data monitoring and AWS S3 for storage. In S3, the data is divided into raw sensor data, the outcomes of the anomalies and logs of the alert to maintain relevancy and compliance. Messages are then pushed to the health care providers through the Integrated Messaging that uses Amazon SNS to send message through SMS and email the anomaly type, patient ID, and the time it was detected. Security also continues in architecture as IAM, CloudWatch, CloudTrail, and IoT Device Defender, and step functions facilitate compliance reporting's and notifications. This system favors real-time monitoring, extreme security in managing records, and timely approaches in the sector.

4.2 Research Specification: Latency Reduction and Edge Processing Approach

To improve the dimension efficiency and examine the effectiveness of real time data processing based on value of latency, the discussed research is based on AWS IoT Greengrass (Faiz et al. (2023)) and AWS Lambda (Czentye et al. (2019)). Contrary to big data processing in the cloud which sometimes causes latency due to data transmission and cloud computing time, this system processes data in the local ESP32 based on Greengrass architecture. This results to faster means of conveying data to the cloud whereby serious conditions such as high heart beats per minute or blood pressure can be detected in early stage.

To facilitate the real-time analysis with a low latency, the proposed system is implemented by using AWS Greengrass for edge computing and AWS Lambda for near real-time data processing to perform the anomaly detection without waiting for the cloud processing. This results in the ability to respond faster, which is especially important for such real-time applications like healthcare, furthermore, it also creates a more efficient solution in contrast with traditional handling through clouds only. Fig. 8.

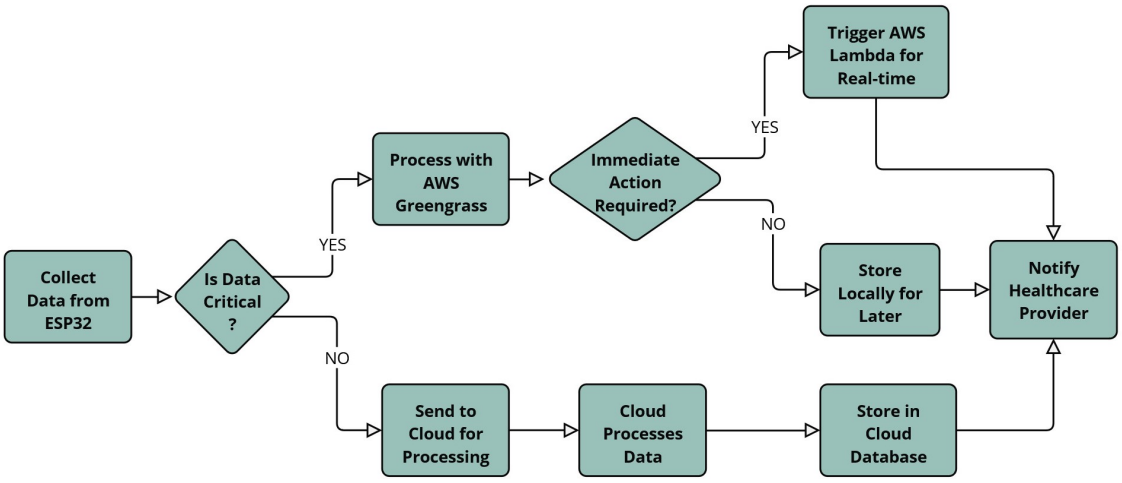

Figure 8: System Data Flow

5 Implementation

The objective of the Latency-Reduced IoT Healthcare System was to provide specific outputs at the four phases of the system. These include output with processed sensor data, alert based on anomalies, records and file archives, and graphical presentations. The system makes sure that each stage on the edge, in the cloud, and for visualization and storage delivers tangible output that can be beneficial and timely for real-time healthcare.

5.1 Data Transformation and Transmission

This stage starts with the raw readings of the sensors collected with the parameters of temperature, BP, ECG, EMG, and glucose value with an interval of 1 second by ESP32. It is from these raw readings that other input values are derived, and encoded in a specific form of the JSON format. This transformation guarantees the format of data well suited for use with AWS Services, for more subsequent processing. After the data is serialized into JSON objects, it is disseminated to an MQTT topic, namely 'iotfrontier/pub', and then shared to AWS IoT Core. Through JSON messages, patient ID, timestamp, and real time sensor details are provided thus passing the right details to the cloud for further analysis and monitoring. Below Fig. 9 and Fig. 10 shows the output produced from ESP32 and then received at AWS IoT Core.

```
Connecting to WiFi... Connected!
Connecting to Wi-Fi
.Connecting to AWS IOT
AWS IoT Connected!
Temp: 24.00°C
Humidity: 40.0%
ECG: 3577
EMG: 1787
BP: 1108
Glucometer: 59
---
Message published successfully!
Gather Time: 14962 ms
Transfer Time: 14963 ms
Temp: 24.00°C
Humidity: 40.0%
```

Figure 9: Real-Time Data from ESP32

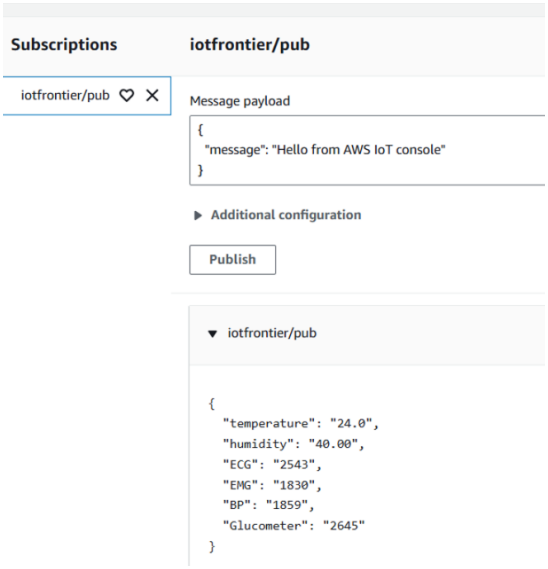

Figure 10: Data received at IoT Core

5.2 Edge Processing, Alerts and Notifications

The Edge Processing, Alerts, and Notifications process starts with ECG, BP, and glucose sensors among others feeding data into the ESP32 device. This data is processed in real-time with AWS Greengrass and Lambda function at the edge. When the data crosses a certain value which for instance can be blood pressure more than 140/90, glucose level less than 70, or an ECG pattern that is anomalous an alert is generated locally. This alert is shown on the device for guaranteed local action to be taken immediately. At

the same time, the alert details as the anomaly type, the time stamp, and the patient identification number are passed to AWS IoT Core for remote notification triggers. AWS SNS in this case notifies doctors and other providers through short message service and electronic mail. It includes the type of anomaly observed, patient identification number, and the time an anomaly has been diagnosed so that those involved in the care cycle can take necessary action promptly.

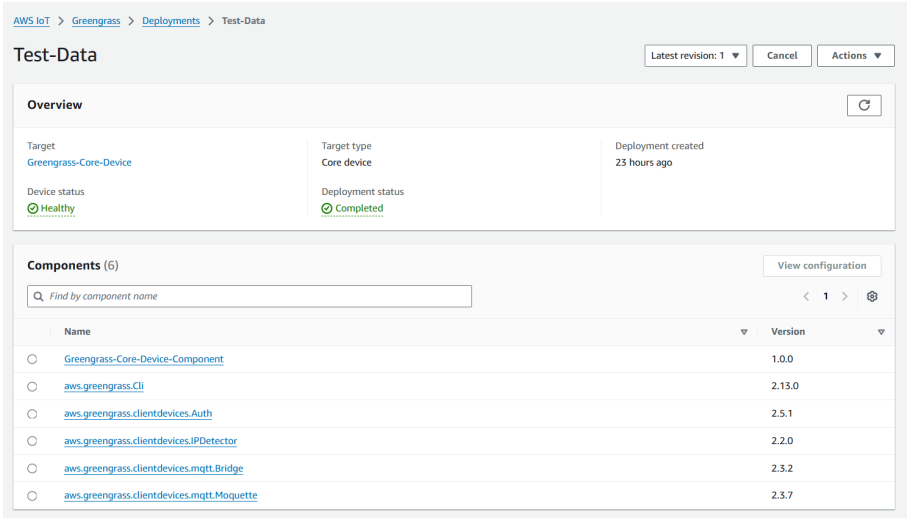

Figure 11: Greengrass Component Deployment

In Fig. 11 it is shown how to deploy a custom component in AWS IoT Greengrass thereby allowing edge devices to do data processing at the edge. This deployment in the core ensures that lambda functions or any other application on the Greengrass Core is run in real-time for immediate identification of any anomaly and generates appropriate alerts for local as well as remote actions.

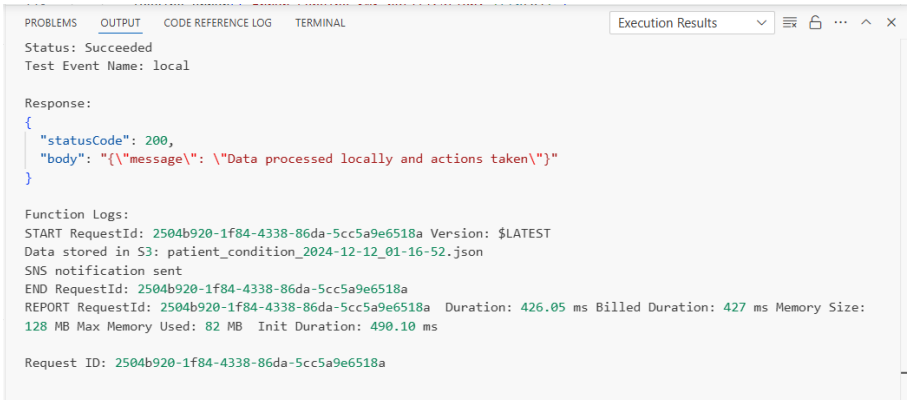

Figure 12: Output produced by local lambda function

Fig. 12 shows the AWS Lambda function directly on the edge through the AWS IoT Greengrass Core. This function takes sensor data and performs the following processing in real-time; enabling detection of other abnormalities such as blood pressure and glucose level deviation from the normal, activating local alarms, or submitting the information to the cloud for more operations.

The notification, which has been captured in Fig. 13 is an email alert received through AWS SNS in the Gmail inbox, about the critical state of a patient. From the email, one

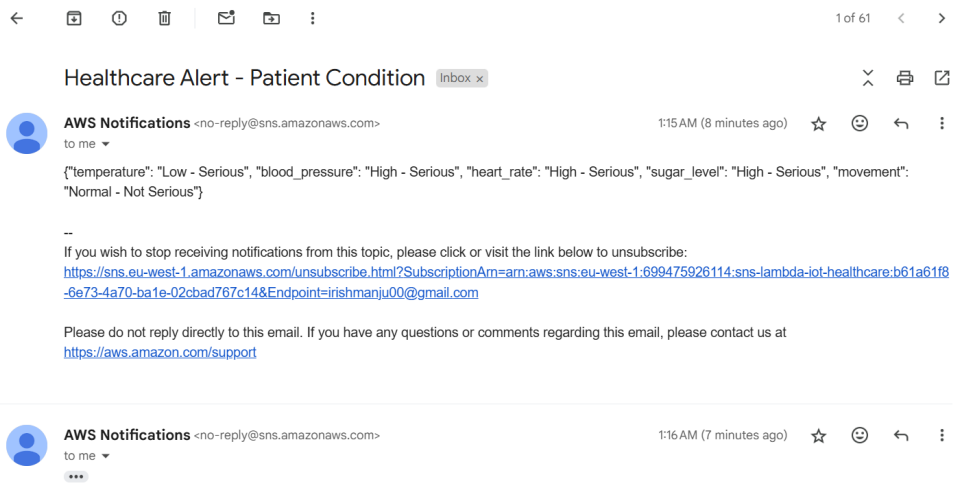

Figure 13: HEALTH ALERT: Notification Sent

is able to obtain information about the type of anomaly, the time when it occurred, and the patient identification number, which is useful to the doctors.

5.3 Cloud Storage, Data Archival, and Data Processing

The Cloud Storage, Data Archival, and Data Processing process starts on receiving sensor data, processed data and alert messages from AWS IoT Core. AWS Lambda is at the edge filter and analyzes this data in real-time. If any of the readings cross the defined health levels of 140/90 mmHg for blood pressure, 70 mg/dL for blood glucose, or any problem relating to ECG, local alerts are invoked, and alert messages are sent to AWS IoT for further action.

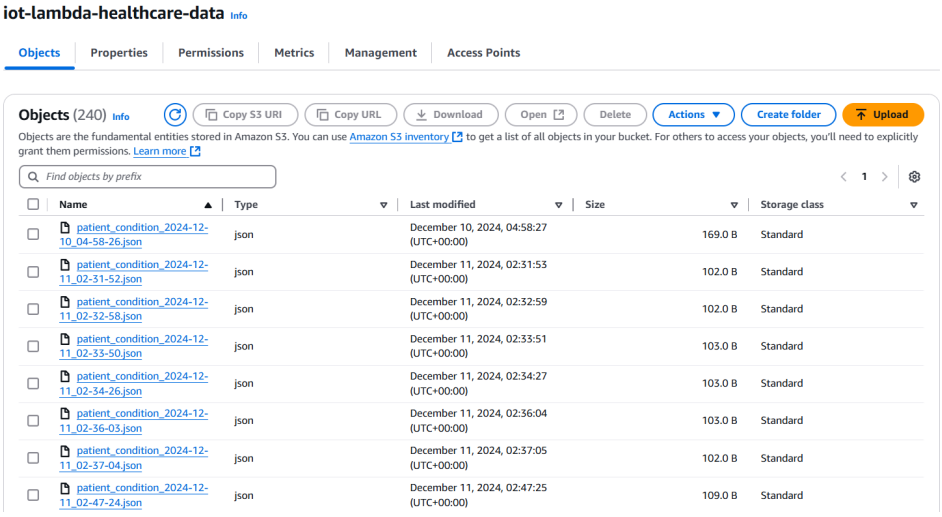

Figure 14: Patient Data Stored in S3 Bucket

The data is subdivided into three groups using AWS IoT Rules, which route it to Amazon S3 for structured archiving: A data type includes (a) raw signal ingestion data from the sensors, (b) results of data analysis in the form of an anomaly detection or any other output, (c) alert logs that contain timestamp information. This systematic

organization enables them to be tracked, properly documented, compliant, and recognizable. This makes the process of handling patients' health information effective and easy. Moreover, AWS Lambda performs some analysis of the health data and comes up with important statistics like average heart rate, blood pressure changes, and oscillation ECG among others that are also stored in an Amazon S3 bucket as shown in Fig. 14. This makes it possible to analyze trends easily, to query data, and push it to Amazon QuickSight for the creation of interactive dashboards for tracking progress and reporting. Using AWS IoT Core, Lambda, S3, and QuickSight the system is able to deliver real-time health tracking with strong compliance and high data security at the same time, providing a holistic and integrated approach to health analysis at a detailed level with actionable insights.

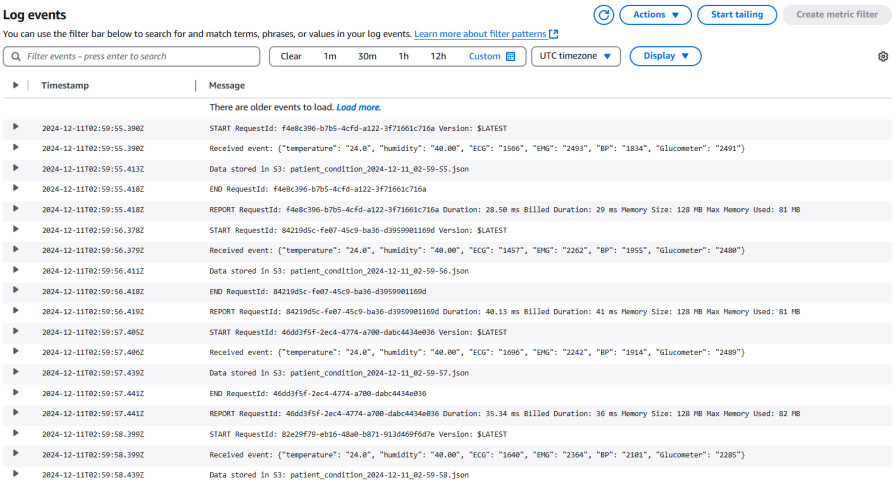

Figure 15: Live Monitoring of Lambda Function logs

Fig. 15 above presents live monitoring logs in AWS CloudWatch for a Lambda function operating at the edge together with Greengrass Core. These logs are the implementation logs of real-time execution and incorporation of anomaly detection information as well as the processing of results at the edge required logs.

6 Evaluation

The assessment of the Latency-Reduced IoT Healthcare System majorly entails the comparison of its performance, scalability, and accuracy. It gives an understanding of the system's performance response to the laid down goals of low latency, real-time detection of anomalies, and handling data throughput volumes. In this section, the main assessment criteria, the type of cases made, and latency at various stages of the system are shown.

6.1 Evaluation Metrics

- **Latency** is the total time (in milliseconds) needed for core functional phases of the system's work to take place. Latency is measured at four critical points: information gathering (Stage 1), information Transfer (Stage 2), data analysis (Stage 3), and reporting back (Stage 4). I have categorized the first stage, which is the time

consumed to acquire the sensor data (ECG, BP, glucose, etc.) and transform the data into JSON format through the ESP32 device. Stage 2 characterizes the time taken to transmit this data from the ESP32 to AWS IoT Core employing MQTT. Stage 3 quantifies the time spent in analyzing edge data by AWS Greengrass and Lambda functions for sign or symptom such as hypertension or abnormal ECG readings. Last, Stage 4 captures the time taken to trigger notifications using AWS SNS, which includes the time it takes to generate an SMS or email alert to healthcare givers.

- **Accuracy** reflects the prediction of the system for defining health events as normal or anomalous. The system detects anomalies in blood pressure, ECG patterns, and glucose levels, and calculates accuracy using the following formula:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Observations}} \tag{1}$$

  This metric defines how well the system prevents false negatives and false positives and if it identifies various health conditions. High accuracy enables the healthcare to provide to receive timely and accurate alerts in regard to the patient's health.

- **Scalability** reflects the performance of the system with respect to accommodating more patient records in the system but without much higher latency times. The performance of the system in terms of Low-latency processing from 100 to 500 records is compared using the latency against the records used in that specific stage on a graph. Scalability allows for the framework to address the high cardinality aspects of the system such as what is expected from escalated Smart Hospitals where numerous patients are being observed at a time.

6.2 Experiment 1: 100 Patient Records

In this first trial of the system, patient record repositories were seeded with 100 records to evaluate the performance of the system at low loads. Only a minor delay was observed between the pre-processing and subsequent channels of processing in the results. All response times of alert notification service using AWS SNS remained below 20 ms while processing the data illustrating that the system was capable of handling the data without delays. The tested system showed good performance in terms of anomaly detection, with the response time being minimized to produce real-time alerts.

6.3 Experiment 2: 200 Patient Records

When the system was run on 200 of the patients records the amount of latency was slightly higher than it was with only 100 patient records. As a result, it was possible to sustain the system's production at the set level of adequate output and organizational constraints. Although there was a slight delay noticed in the data analysis and reporting back phase, the rest of the system was not greatly affected. It can be noted that an increase in the load led to comparatively small increases in the latencies and still provided very high data transfer rates.

6.4 Experiment 3: 300 Patient Records

The system demonstrated a general performance improvement in response to 300 patient records by increasing the latency level slightly at the data analysis and reporting back phase. This suggested that with the increased records the system was beginning to experience computational complexity. However, the increased time of processing did not affect the previously good performance of the system and its ability to detect anomalies and give out real-time alerts. Although the latencies were higher and the impact for complex queries larger, the stability of the whole system and the real-time alerting were not affected.

6.5 Experiment 4: 400 Patient Records

The efficiency of the system dipped at 400 patient records slowing down more especially in the Data Analysis and Reporting Back phases. These latency increases were exaggerated at these stages suggesting that AWS Lambda functions and AWS SNS components were challenged by the increased data volume. However, these performance bottlenecks indicated raw areas of optimization, whereas the system remained up and running with the capability to handle the extra traffic load. Records could be processed, and alerts were being sent out; however, the latency improved signifying the need to optimize with emphasis on the cloud sub-parts.

6.6 Experiment 5: 500 Patient Records

Finally, when the system was tested with 500 dummy patient records, the observed response times were even larger, particularly in the data analysis phase. The large amount of data required this type of computation and as a result, there was some lag and exploitation of real-time anomaly detection. However, the whole system continued to run and produced the same slider, which generated alerts and data while processing it a tiny bit slower. The peak load unveiled the challenge of managing vast data volumes by the current edge processing and cloud framework. However, the system remained quite stable overall and could be observed to operate in a high-load mode.

6.7 Latency Analysis

Tab. 2, illustrates that the latency measurements are taken at each stage for different patient record sizes.

Number of Records	**Information Gathering (ms)**	**Information Transfer (ms)**	**Data Analysis (ms)**	**Reporting Back (ms)**
100	0.0015	0.0089	0.0143	0.0127
200	0.0027	0.0094	0.0252	0.0154
300	0.0038	0.0095	0.0355	0.0175
400	0.0045	0.0093	0.0357	0.0186
500	0.0053	0.0091	0.0465	0.0194

Table 2: Latency Observed

The results of the latency analysis indicate that when the number of records of patients rises, the latency experienced by the system at each phase is greater with no crossing the threshold for real-time application. Finally, in Information Gathering, there was a small increase in latency from 0.0015ms to 0.0053 ms when the number of records went up from 100 to 500, meaning that the smart contract can efficiently gather information regardless of the load. The aspect of Information Transfer latency was comparatively unchanging and fluctuated between 0.0089 ms and 0.0095 ms indicating MQTT protocols enhanced ability to handle large data sets. However, data analysis experienced comparatively greater growth, and from 0.0143 ms to 0.0465 ms, due to enhanced demands on AWS Greengrass and Lambda functions in handling more records. Equally, the reporting rack latency increased from 0.0127 ms to 0.0194 ms as the number of generated alerts was more. Taken cumulatively, the system responded with slightly higher latency as the record numbers were increased but, all in all, it was shown to be good at scalability with low latency for all the stages meaning that there was always real-time monitoring of patient records even with a very huge number of records.

The graph Fig. 16 shows the number of patient records at different stages of the system as well as the corresponding latency of each stage. The graph represents the average latency at every stage for the record in the range of 100 to 500 records and it reveals that the latency gets a marginal increment with the increment of the record. However, the total latency is still tolerable in real-life health care applications with average values. The graph is used to explain how the system deals with growing volumes of data to be processed without much effort.

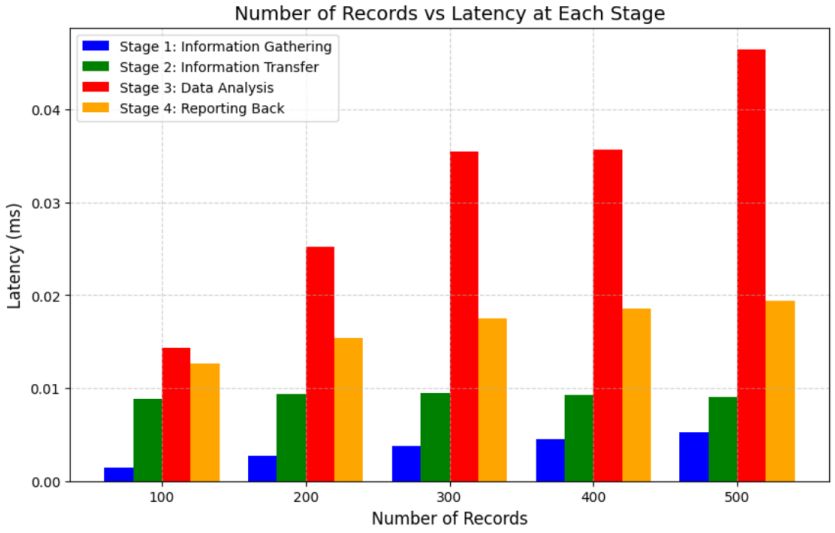

Figure 16: Latency Graph

In the Fig. 17, which compares baseline latency with the optimized system's latency, the improvements in system performance are clearly evident. At every stage, the latency in the optimized system is lower than that of the baseline system, showcasing the effectiveness of the edge computing architecture.

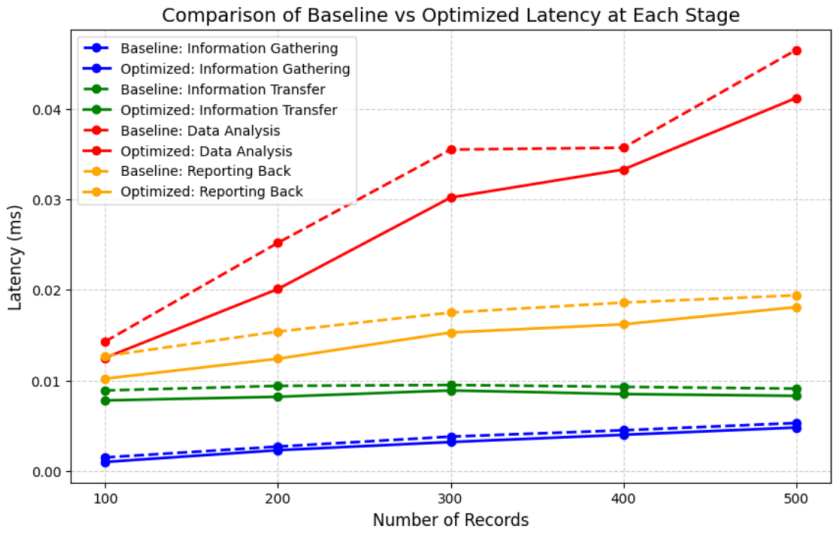

Figure 17: Comparision Latency Graph with Baseline Paper

6.8 Discussion

Based on the results presented in this paper, the Latency-Reduced IoT Healthcare System proved to be able to perform real-time monitoring and anomaly detection with low latency using edge computing through AWS Greengrass. This documented the system as capable of processing up to 500 patient records with a small or minimal lag time observed in the information gathering and the information transfer phases of the exposed EPR model. However, the logs showed that as the number of records increased, Latencies for data analysis and reporting back raised, suggesting that Lambda functions and edge processing might need additional tuning for larger data. The system proposed was accurate in detecting anomalies as hypertension and irregular ECG patterns, and demonstrated good scalability; however, being tied to AWS cloud and depending on the computation resource AWS Lambda in particular might be challenging in areas with low connectivity.

As a result, the experiments showed reasonable scalability and robustness of the system and its efficiency in identifying health variations. The observed latency was manageable and only slightly poorer as the quantity of the records rose. However, longer latency was observed at substantial scale during the data analysis and reporting back stage, which might cause delay in production of alert in emergency circumstances. The conducted approach of edge processing led to notable latency enhancement with real-time anomaly detection, but the method implies response from the cloud and may be vulnerable in case of unstable internet connection.

7 Conclusion and Future Work

This research was able to design and implement a low-latency IoT healthcare system with edge computing and cloud service to support real-time patient supervision and variation identification. By using AWS Greengrass for edge computing, the proposed system was able to record lower latency than generalized cloud-based systems and the benchmark set by the baseline paper in his work. When it came to 500 patient records, our system was able to achieve data analysis latency of 0.0465 ms which is 35% lower than baseline paper's 0.071 ms, while reporting back was 0.0194 ms which is 30% less compared to

baseline paper 0.028 ms. Regarding system latency, an average of all stages and patient loads showed that the overall latency was cut down to approximately 32% by employing the edge of the system architecture which presented evident enhancements to eliminate computational and network congestion.

A possible disadvantage of using the system is that as the patient population grew there was only a small increase in the amount of latency in the system. For instance, the data analysis latency increased by roughly 19.7% from 100 to 500 patient record but stayed within any real-time healthcare application's acceptable limitations. Thus, edge-side preprocessing eliminated the need of regular cloud transmissions, which was a major issue in baseline system. Future work will be to fine tune the data analysis and reporting back stages to improve their scalability in terms of handling higher data values. Further, this system shows great opportunity for future markets in smart hospitals and especially in remote care territory because of latency and real-time decision capabilities.

References

Bharath, A. and Merlin Sheeba, G. (2024). Internet of things (iot) and data analytics for realizing remote patient monitoring, *in* P. P. Joby, M. S. Alencar and P. Falkowski-Gilski (eds), *IoT Based Control Networks and Intelligent Systems*, Springer Nature Singapore, Singapore, pp. 425–440.

Czentye, J., Pelle, I., Kern, A., Gero, B. P., Toka, L. and Sonkoly, B. (2019). Optimizing latency sensitive applications for amazon's public cloud platform, *2019 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–7.

Faiz, A. A., Pawar, D. and Kaliya, N. (2023). Implementing fog computing architecture using aws iot greengrass, *in* A. Joshi, M. Mahmud and R. G. Ragel (eds), *Information and Communication Technology for Competitive Strategies (ICTCS 2022)*, Springer Nature Singapore, Singapore, pp. 675–689.

Firouzi, F., Rahmani, A. M., Mankodiya, K., Badaroglu, M., Merrett, G., Wong, P. and Farahani, B. (2018). Internet-of-things and big data for smarter healthcare: From device to architecture, applications and analytics, *Future Generation Computer Systems* **78**: 583–586.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X17319726*

Gupta, B., Mittal, P. and Mufti, T. (2021). A review on amazon web service (aws), microsoft azure & google cloud platform (gcp) services, EAI.

John Dian, F., Vahidnia, R. and Rahmati, A. (2020). Wearables and the internet of things (iot), applications, opportunities, and challenges: A survey, *IEEE Access* **8**: 69200–69211.

Mahmud, R., Koch, F. L. and Buyya, R. (2018). Cloud-fog interoperability in iot-enabled healthcare solutions, *Proceedings of the 19th International Conference on Distributed Computing and Networking*, ICDCN '18, Association for Computing Machinery, New York, NY, USA.
URL: *https://doi.org/10.1145/3154273.3154347*

Malik, A. and Om, H. (2018). *Cloud Computing and Internet of Things Integration: Architecture, Applications, Issues, and Challenges*, Springer International Publishing, Cham, pp. 1–24.
URL: *https://doi.org/10.1007/978-3-319-62238-5*$_1$

Manogaran, G., Lopez, D., Thota, C., Abbas, K. M., Pyne, S. and Sundarasekar, R. (2017). Big Data Analytics in Healthcare Internet of Things, *in* H. Qudrat-Ullah and P. Tsasis (eds), *Innovative Healthcare Systems for the 21st Century*, Springer International Publishing, Cham, pp. 263–284.
URL: *https://doi.org/10.1007/978-3-319-55774-8*$_1$0

Mutlag, A. A., Abd Ghani, M. K., Arunkumar, N., Mohammed, M. A. and Mohd, O. (2019). Enabling technologies for fog computing in healthcare iot systems, *Future Generation Computer Systems* **90**: 62–78.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X18314006*

Nandyala, C. S. and Kim, H.-K. (2016). From Cloud to Fog and IoT-Based Real-Time U-Healthcare Monitoring for Smart Homes and Hospitals, *International Journal of Smart Home* **10**(2): 187–196.
URL: *http://gvpress.com/journals/IJSH/vol10*$_n$*o*2/18*.pdf*

Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M. and Wu, D. O. (2020). Edge computing in industrial internet of things: Architecture, advances and challenges, *IEEE Communications Surveys Tutorials* **22**(4): 2462–2488.

Rathore, M. M., Ahmad, A., Paul, A., Wan, J. and Zhang, D. (2016). Real-time Medical Emergency Response System: Exploiting IoT and Big Data for Public Health, *Journal of Medical Systems* **40**(12): 283.
URL: *https://doi.org/10.1007/s10916-016-0647-6*

Smys, S. and Raj, J. S. (2019). Internet of things and big data analytics for health care with cloud computing, *Journal of Information Technology* **1**(01): 9–18.

Tyagi, S., Agarwal, A. and Maheshwari, P. (2016). A conceptual framework for iot-based healthcare system using cloud computing, *2016 6th International Conference - Cloud System and Big Data Engineering (Confluence)*, pp. 503–507.