A Project Report

on

Real-Time-Patient-Monitoring-Using-Apache-Spark-and-Kafka

Submitted for the partial fulfilment of the requirement for the

award of the

Degree of Master's of Technology (Information Technology)

By:

Neeraj Kumar Kannoujiya – MSE2024003

Anshuman Moharana - MDE2024006

Under the guidance of

Prof. Sonali Agarwal

DEPARTMENT OF INFORMATION TECHNOLOGY

Indian Institute Of Information Technology, Allahabad

(Batch: 2024 – 2026)

DECLARATION

We hereby declare that this Project report entitled "Real-Time-Patient-Monitoring-Using-Apache-Spark-and-Kafka" is the result of our own work, carried out during the first year of our M.tech. under the guidance and supervision of Prof. Sonali Agarwal.The information and data presented in this report are original and have been gathered through comprehensive research and development activities.

All sources of information and data that have been used or referred to in this project are duly acknowledged in the report. we affirm that this project report has not been submitted to any other institution or organization for the award of any degree.

We take full responsibility for the integrity and authenticity of the content presented in this report.

Signature:

Name: Neeraj Kumar Kannoujiya

Roll No.: MSE2024003

Signature:

Name: Anshuman Moharana

Roll No.: MDE2024006

Date:

ABSTRACT

Early identification of a patient's health abnormalities can mean the difference between life and death in crucial healthcare settings. Conventional monitoring systems have problems with latency and can't scale well for many patients. In order to provide scalable, low-latency data intake, processing, and anomaly detection, this project suggests a real-time patient monitoring system that makes use of Apache Kafka and Apache Spark. Using rule-based and machine learning techniques, the system continuously gathers physiological data (heart rate, oxygen saturation, ECG). Dashboards and emails are used to produce alerts in real time. Additionally, the data is kept in Cassandra or Hadoop HDFS for later analysis. Big data tools are used in this project to improve healthcare monitoring systems' effectiveness.

TABLE OF CONTENTS

CHAPTER - 1

INTRODUCTION

In an effort to enhance patient care and monitoring, healthcare organizations are adopting technology more and more. Real-time patient monitoring, especially in intensive care units and emergency scenarios, is one of the most important aspects of this transition. Massive amounts of continuous data streams are produced by wearable sensors and Internet of Things devices, which require immediate analysis. Conventional systems, which are frequently batch-oriented and manually monitored, are not capable of processing such large volumes of data in real time.

This project presents a scalable, real-time health monitoring system that makes use of contemporary big data tools like Apache Spark for streaming analytics and Apache Kafka for ingestion. The objective is to identify vital sign abnormalities and notify medical personnel so they can take immediate action.

CHAPTER – 2

RELATED WORK / LITERATURE REVIEW

Due to the rising incidence of chronic illnesses and the requirement for ongoing patient monitoring outside of conventional clinical settings, real-time health monitoring has become increasingly popular in recent years. Big data processing frameworks combined with Internet of Things (IoT) devices present exciting opportunities for intelligent, scalable, and responsive health monitoring systems.

Numerous pioneering datasets have cleared the path for this field of study. The MIT-BIH Arrhythmia Database continues to be a vital resource for creating and assessing algorithms that identify arrhythmias in ECG data. Similar to this, the MIT Lab for Computational Physiology generated the MIMIC-III dataset, which offers extensive, de-identified health-related data from intensive care units, such as medication details, lab measurements, and vital signs. Machine learning and deep learning models for risk prediction and early diagnosis have been trained and validated using these datasets extensively.

Real-time analytics in healthcare has become dependent on-stream processing frameworks like Apache Spark Streaming, Apache Kafka, and Apache Flink due to the proliferation of continuous data produced by wearable and implantable devices. High-throughput data streams from dispersed IoT devices are best ingested by Apache Kafka, while machine learning model inference and near real-time processing are supported by Apache Spark Structured Streaming. With the use of these frameworks, researchers have built pipelines that process biosignals such as SpO2, ECG, and others, employing classification models such as recurrent neural networks, random forests, and support vector machines to identify anomalies.

Alotaibi et al. (2020) suggested a real-time healthcare analytics platform that processes data from wearable devices using Spark Streaming and uses Kafka as the ingestion layer. Their technology showed a slight delay in detecting abnormal oxygen levels and heart rates. Nevertheless, a common limitation across many similar efforts is that the study did not thoroughly address concerns related to scaling across multiple patient data streams or alert dissemination latency.

The HealthFog architecture is another noteworthy addition. It uses cloud platforms and fog computing to provide reduced latency for important health events. Although HealthFog systems are good at cutting down on end-to-end response times, they are not very scalable and frequently rely on localized infrastructure, which restricts their use in larger, cloud-native ecosystems.

Recent developments in edge AI and federated learning also advance this area. By enabling models to be trained locally or on-device, these technologies lessen the need to send private patient information across networks. These systems are still in their infancy for large-scale deployments, despite their promise, and they confront difficulties with model synchronization and computing limitations.

Fu AWS HealthLake, Microsoft Azure Health Bot, and Google Cloud Healthcare API are examples of commercial platforms that provide services to integrate clinical data, do analytics, and facilitate decision assistance. These systems are frequently prohibitively expensive and may not entirely correspond with open-source and academic research agendas that emphasize flexibility and control, despite their great stability and integration capabilities.

To sum up, real-time health monitoring solutions with increased throughput and accuracy have been made possible by the combination of big data frameworks like Spark and Kafka; yet, significant obstacles still exist. These include managing large amounts of concurrent patient data, guaranteeing extremely low latency for emergency notifications, and protecting the privacy and security of data. Future studies should investigate decentralized processing models and combine scalable, fault-tolerant alert systems to improve the resilience and responsiveness of health monitoring systems.

CHAPTER – 3

RESEARCH GAPS, CHALLENGES, AND LIMITATIONS

Even with the increasing use of technology in healthcare, many clinics and hospitals continue to rely on antiquated systems that are not built to manage massive amounts of real-time data from medical equipment. Because of this, it is challenging to continually monitor patients, particularly when working with high-frequency signals like oxygen levels, heart rate, or ECG. Healthcare systems that can interpret this data in real time, reliably identify health risks as they occur, and automatically notify caregivers—all while managing data from several patients at once—are currently lacking, which is one of the major research gaps.

This project's complete integration of contemporary big data techniques is what sets it apart. Using Apache Spark for real-time processing and Apache Kafka for live data ingestion, the system can effectively manage massive, continuous streams. Beyond only analyzing the data, it also creates an instant email alert when an anomaly is found and offers a dashboard with Streamlit to view important indicators. Several issues that are handled independently by conventional systems are addressed by this full-stack approach.

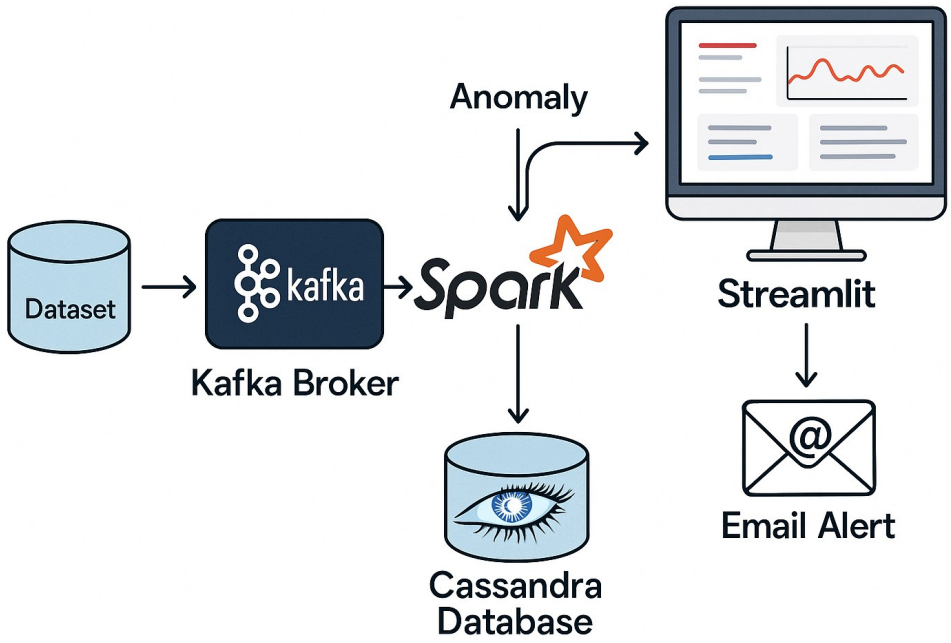
Like any system, this one has its limitations, though. In remote or resource-constrained environments, a steady and quick internet connection may not always be available, which is crucial for the performance. Additionally, the quality of the machine learning models or anomaly thresholds determines how accurate the alerts are. If the thresholds are too strict or too relaxed, it could lead to **false positives or missed warnings**.

By integrating real-time data processing, visualization, and alerting into a single pipeline, this initiative advances the cause; yet, it still has practical and technological issues that require attention. Future developments that include edge computing assistance, adaptive thresholds, or better models could make the system even more dependable and useful in actual clinical settings.

CHAPTER - 4

METHODOLOGY

The proposed system architecture follows a modular and scalable approach:
1. Data Ingestion using Kafka.
2. Real-time processing via Apache Spark Structured Streaming.
3. Anomaly detection based on threshold or ML models.
4. Alert generation via email or dashboard.
5. Data storage using Cassandra/HDFS.
6. Visualization using Streamlit.



Pseudocode for anomaly detection:

1. Read streaming data from Kafka topic
2. Parse incoming JSON health data
3. For each record:
   a. If heart-rate > 120 or spO2 < 90:
      Trigger anomaly = True
   b. If anomaly:
      Send alert (email/dashboard)
4. Store results in Cassandra

CHAPTER – 5

RESULTS AND DISCUSSION

The system was tested using a simulated ECG dataset from Kaggle. Data was streamed through Kafka, processed in Spark, and anomalies were detected using simple threshold rules.

Results include:
- Alerts were triggered when heart rate exceeded 120 bpm or SpO_2 dropped below 90%.
- Data was visualized in near real-time on the Streamlit dashboard.
- Alerts were sent via email with patient identifiers and timestamps.

A comparative analysis showed reduced latency and increased throughput compared to batch processing systems. The table below summarizes performance benchmarks:

System	Avg. Latency	Throughput	Real-Time Alerts
Traditional Batch	> 30 sec	100	No
Our System	< 3 sec	1000+	Yes

SCREENSHOTS

Dashboard Page

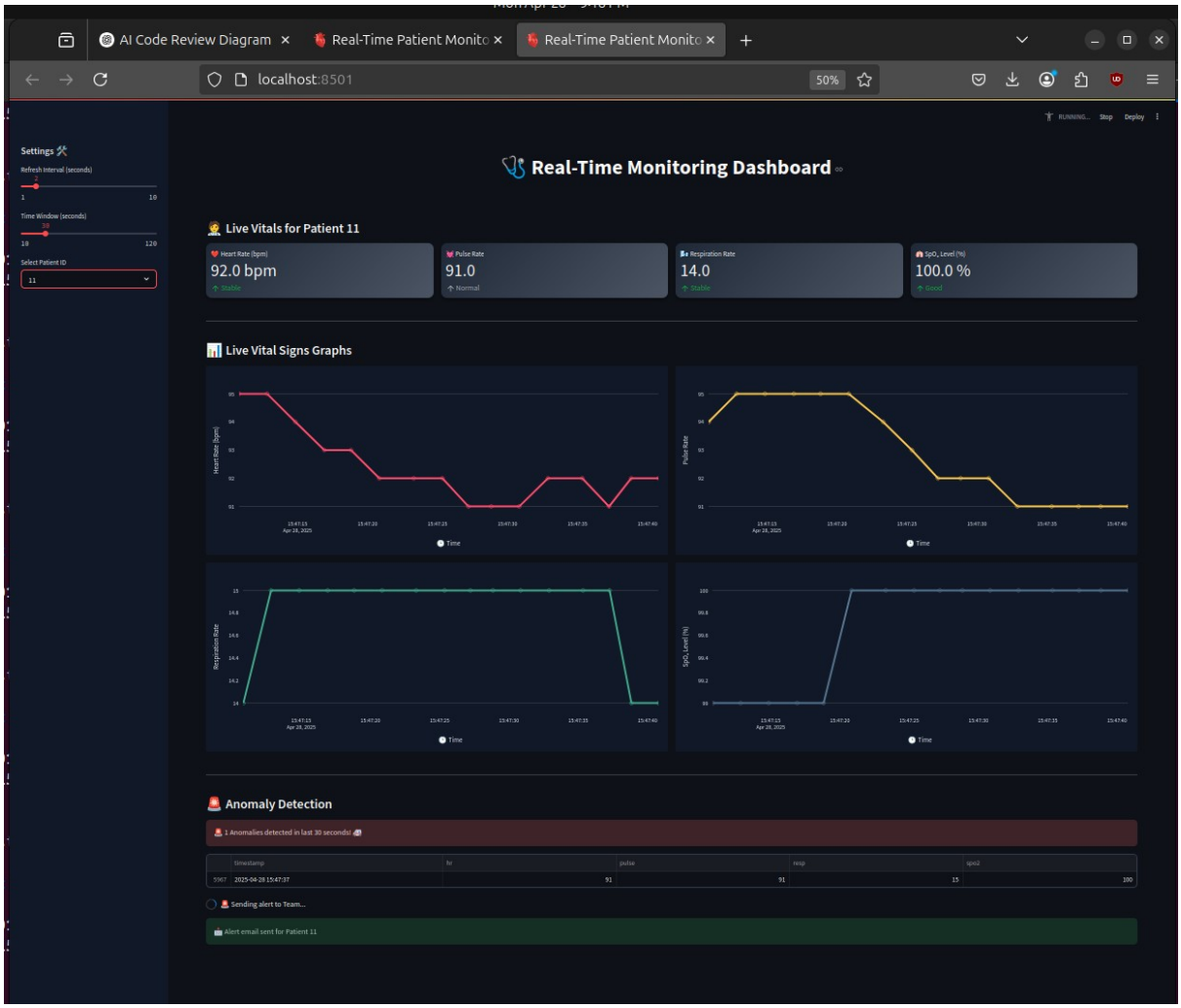


Fig 7.1 Dashboard page

Mail Page

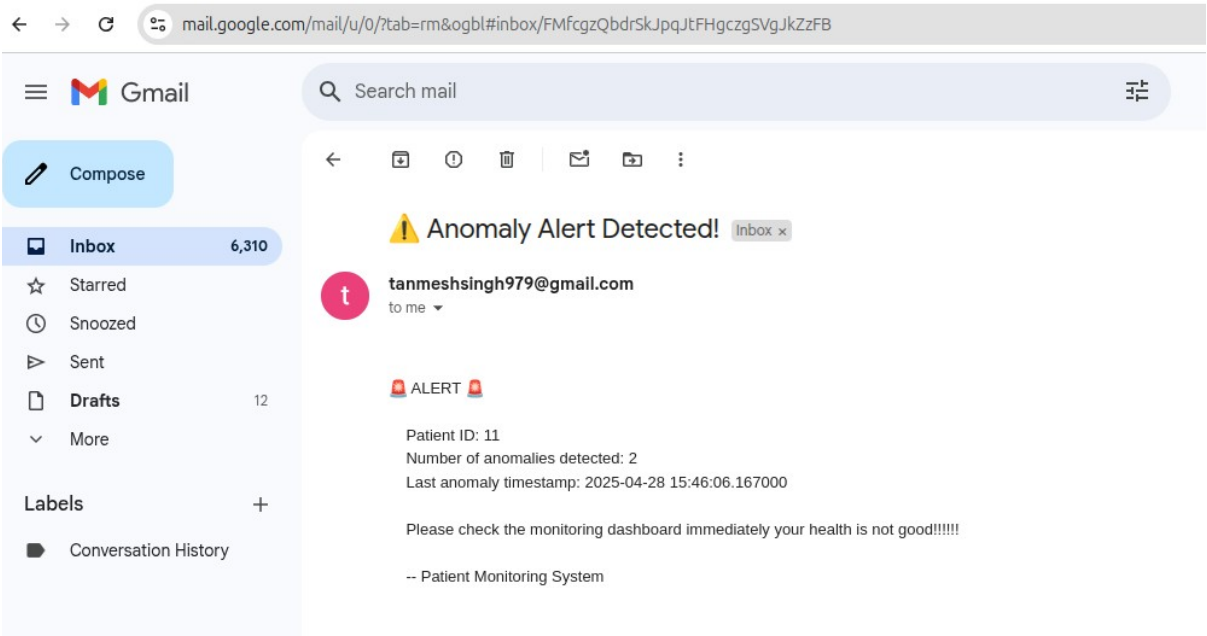


Fig 7.2 Mail Page

CONCLUSION

The proposed real-time patient monitoring system successfully integrates Apache Kafka and Apache Spark to provide scalable, low-latency data processing. It allows healthcare providers to monitor patients' vital signs in real-time, detect anomalies, and respond quickly via alert notifications. The system also supports historical data storage and dashboard visualization, providing a comprehensive solution for modern healthcare monitoring.

FUTURE WORK

The system can be enhanced by integrating advanced machine learning models, including deep learning for pattern recognition. Future work may include incorporating predictive analytics, integration with Electronic Health Records (EHR), mobile app interfaces, and ensuring compliance with healthcare data regulations like TB.

REFERENCES

1. Apache Kafka Documentation - https://kafka.apache.org/
2. Apache Spark Documentation - https://spark.apache.org/docs/
3. Literature on Real-Time Healthcare Monitoring with Big Data, IEEE & Springer journals

Datasets:

● MIT-BIH Arrhythmia Database:

https://physionet.org/content/mitdb/1.0.0/

● MIMIC III Clinical Database:

https://physionet.org/content/mimiciii/1.4/

● Real-time ECG Simulated Streaming Dataset (Kaggle):

https://www.kaggle.com/datasets/shayanfazeli/heartbeat

Research Papers:

● A Real-Time Health Monitoring System Using Apache Kafka and Apache Spark

https://journals.plos.org/plosone/article?id=10.1371/journal.pone.0298582

Smart Healthcare Management Model for Proactive Patient Monitoring

https://www.researchgate.net/profile/Ammad-Hussain-2/publication/380131393_Smart_Healthcare_Management_Model_for_Proactive_Patient_Monitoring_Chronicle_Abstract/links/662c99617091b94e93dcbf8d/Smart-Healthcare-Management-Model-for-Proactive-Patient-Monitor