Digital Egypt Pioneers

Final Project

IoT Healthcare Data Pipeline

Prepared by :

Rawda Abokhalil Youssef Ahmed

Ahmed Youssef Rahma AbdElaleim

Belal Ahmed

Supervisor: Eng. Mohamed Hamed

October 2025

Table of Contents

1-Table of figures:

Abstract

This project presents the design and implementation of a smart IoT-based healthcare data pipeline that simulates, processes, and visualizes patients' vital signs in real time. The system aims to address the challenge of delayed medical response due to the lack of continuous patient monitoring in hospitals.
Instead of using real IoT hardware, a data simulation engine was developed to generate live patient readings that mimic real sensor data, such as heart rate, oxygen saturation, glucose level, blood pressure, temperature, and respiratory rate.

The generated data is streamed through Apache Kafka, processed in real time using Apache Spark Streaming, stored in PostgreSQL, and visualized through an interactive Streamlit dashboard.

This project combines IoT simulation and data engineering concepts to provide a scalable, real-time monitoring and alerting system that supports faster medical decisions and enhances patient safety.

Keywords: IoT, Data Engineering, Healthcare, Spark Streaming, Real-time Pipeline, Streamlit

---

1. Introduction

1.1 Background

In modern hospitals, continuous patient monitoring is a crucial factor in ensuring timely medical intervention. However, manual observation often causes delays in detecting health deterioration, particularly in crowded healthcare environments. The Internet of Things (IoT) provides an opportunity to automate this process through interconnected sensors that collect and transmit patient data in real time.

1.2 Problem Statement

Thousands of patients experience delays in receiving medical attention due to manual and periodic monitoring. The absence of real-time analytics and alerting systems results in missed early warning signs of deteriorating conditions. Moreover, managing the massive data generated from IoT devices remains a challenge for hospitals due to the need for high-speed processing and storage solutions.

1.3 Project Objective

The main objective of this project is to build a complete IoT-based healthcare data pipeline capable of simulating and analyzing patient vital signs in real time. The system should detect anomalies automatically and alert doctors to critical conditions.

Specific objectives include:

- Simulating patient vital signs as continuous IoT data streams.
- Building a full real-time data pipeline for monitoring and alerting.
- Storing and managing large-scale healthcare data efficiently.
- Providing real-time visualization through an interactive dashboard.

1.4 Research Contribution

This project demonstrates how IoT data pipelines can be integrated with real-time stream processing to improve healthcare decision-making. It provides a practical model that can be adapted for real hospital environments, reducing response time and improving patient outcomes.

---

2. Literature Review

2.1 Background

IoT has transformed healthcare by enabling continuous monitoring through wearable and embedded devices. Data from these sensors can be analyzed using stream processing frameworks such as Apache Spark or Flink to detect anomalies in real time. Previous studies highlight the importance of scalable data architectures to handle large data volumes efficiently.

2.2 Related Work

Recent implementations of IoT-based monitoring systems have focused on connecting physical sensors with centralized dashboards. However, most solutions rely on batch processing, which delays detection. Integrating real-time pipelines using tools like Kafka and Spark allows immediate response to health emergencies.

This project builds upon these approaches by incorporating data simulation, streaming, and real-time visualization into one unified architecture.

2.3 Summary

The reviewed literature confirms that real-time processing and IoT simulation are key to enhancing healthcare monitoring. Our system extends these ideas by offering a cost-effective simulated pipeline suitable for research and training environments.

---

3. Methods and Materials

3.1 System Overview

The system architecture is composed of five main layers:

1. Data Simulation
2. Data Streaming
3. Real-time Processing
4. Data Storage
5. Visualization and Alerting

Each layer plays a critical role in ensuring continuous data flow, analysis, and presentation.

3.2 Architecture Description

1. Data Simulation:
A Python script continuously generates random readings for multiple patients. Each simulated stream represents an IoT sensor measuring heart rate, oxygen saturation, temperature, blood pressure, and glucose.

2. Data Streaming:
The generated readings are published to Apache Kafka, acting as a message broker that ensures reliable real-time data transmission.

3. Real-Time Processing:
Using Apache Spark Streaming, the system processes incoming data to detect anomalies. For example:

- Oxygen saturation < 90%
- Heart rate > 120 bpm
  Detected abnormalities trigger alerts that are logged into a dedicated database table.

4. Data Storage:
Processed and cleaned data is stored in PostgreSQL for long-term analysis. This allows tracking of patient history and generating reports.

5. Dashboard Visualization:
A Streamlit dashboard was developed for doctors and hospital staff. It displays live patient readings, summary statistics, and alerts through a user-friendly interface.

3.3 Tools and Technologies

Layer	Technology	Description
Data Simulation	Python	Generates randomized patient vital data
Streaming	Apache Kafka	Handles real-time data transmission
Processing	Apache Spark	Performs real-time processing and anomaly detection
Database	PostgreSQL	Stores processed patient data and alerts

---

4. Implementation

4.1 Development Steps

1. Data Simulation

The first stage of the project was to design and implement a Python-based data simulation engine capable of generating continuous, realistic, and medically relevant patient vital signs.
Instead of relying on physical IoT devices, we developed a modular simulation script with the following features:

- Multi-patient support:
  The generator can simulate dozens of patients at once, each with an independent "virtual sensor."
- Realistic medical ranges:
  Values were generated based on medically accepted ranges, including:
  - Heart Rate (60–100 bpm)
  - Oxygen Saturation (95–100%)
  - Blood Pressure (Systolic & Diastolic)
  - Body Temperature (36.5–37.5 °C)
  - Glucose Level
  - Respiratory Rate
- Time-based streaming:
  Readings are produced at regular intervals (e.g., every 2 seconds), imitating real IoT sensor frequency.

- Randomized fluctuations:
  Slight variations and sudden spikes/drops were introduced to simulate real patient behavior.
- JSON formatting:
  Each reading is packaged as a structured JSON object containing:
  - Patient ID
  - Device ID
  - Timestamp
  - All vital metrics

This generator essentially acts as a virtual hospital continuously producing data for further processing.

---

2. Kafka Integration

To handle the continuous flow of simulated data, we integrated Apache Kafka as the data ingestion layer.
Kafka was chosen due to its ability to support high-throughput, low-latency, fault-tolerant message streaming.

Key implementation steps:

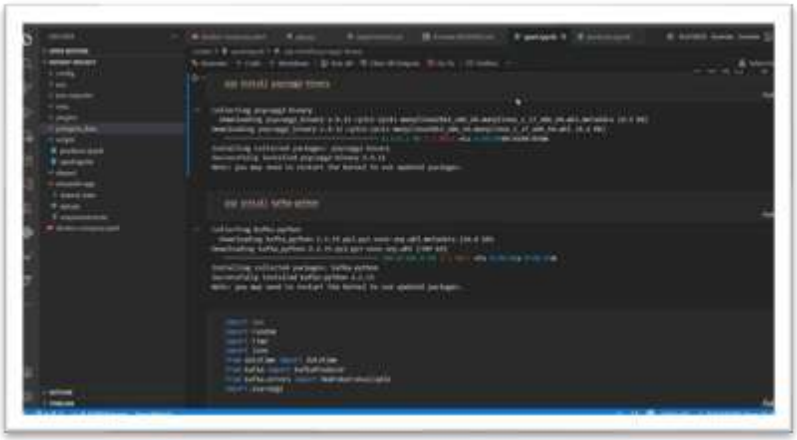

Fig 1 . Kafka

- Topic Creation:
  Kafka topics were created to logically separate streams (e.g., patient_vitals_stream).
- Producers:
  The Python generator functions as a Kafka producer, publishing each JSON reading to the appropriate topic.
- Scalability:
  Kafka's partitioning allows horizontally scaling the system if the number of patients increases.
- Durability:
  Messages are written to Kafka logs to ensure no data is lost even if consumers temporarily fail.

Kafka therefore acts as the central messaging backbone, guaranteeing reliable data delivery before real-time processing.

3. Spark Streaming

Apache Spark was used as the real-time processing and anomaly detection layer.
A Spark Structured Streaming job continuously consumes data from the Kafka topics.

Spark performs several crucial tasks:

A. Data Ingestion

- Subscribes to Kafka topic(s)
-
- Parses JSON messages
- Converts them into a structured DataFrame for processing

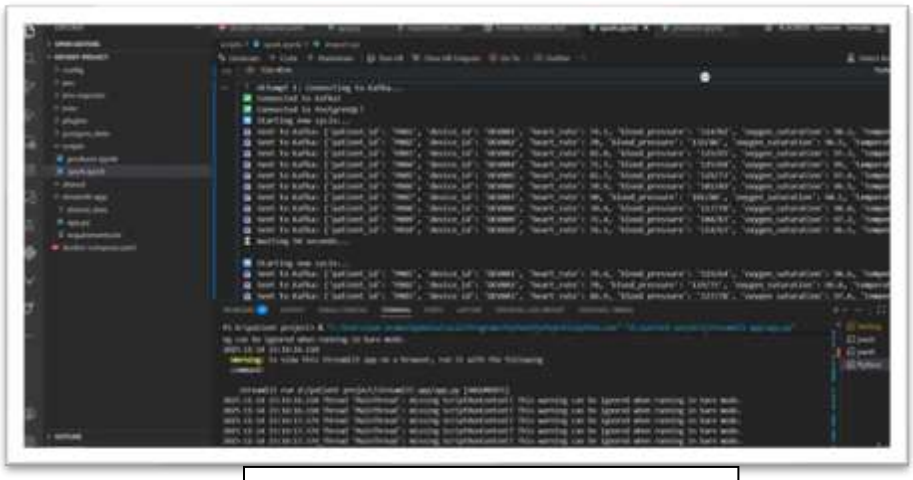

Fig 2 . Spark

B. Data Cleaning & Validation

- Ensures all expected fields exist
- Converts invalid or corrupted values into null
- Removes noise or incorrectly formatted messages

C. Anomaly Detection

Spark applies medical logic to identify abnormal readings, such as:

- Oxygen Saturation < 90% → possible respiratory risk
- Heart Rate > 120 bpm → tachycardia
- Glucose > 180 mg/dL → hyperglycemia
- Low temperature or high fever

- Critical combinations (e.g., high HR + low oxygen)

For each anomaly, Spark creates a structured alert entry containing:

- Patient ID
- Exact metric causing the alert
- Severity level
- Description of the risk
- Timestamp

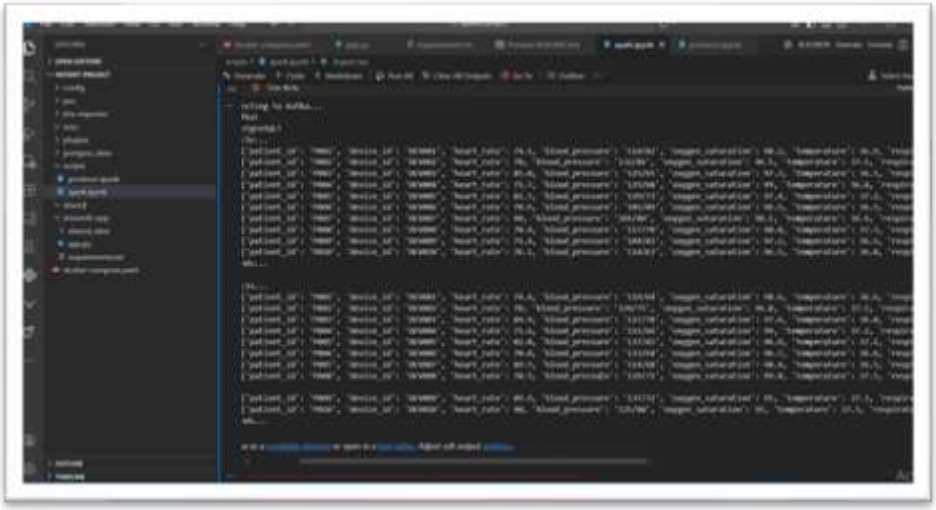

Fig 3 . Spark

D. Database Writing

Spark writes two types of output:

1. Processed Vitals Table:
   Normalized, cleaned vital readings for history and analysis.
2. Alerts_Log Table:
   All anomalies detected in real time.

By doing this, Spark acts as the core intelligence layer of the system.

---

4. Database Schema Design

A relational database (PostgreSQL) was designed to store all structured data.
The schema ensures:

- Accuracy
- Integrity
- Fast querying
- Long-term storage

Main components:

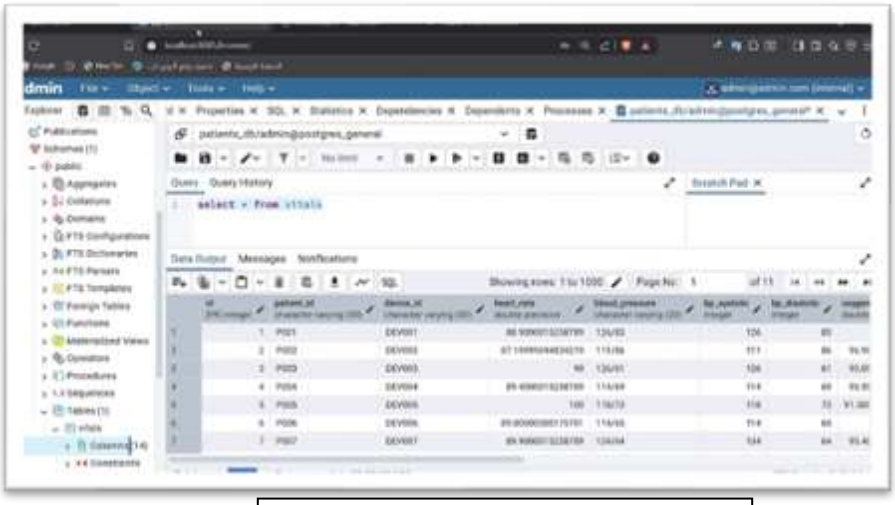

Fig 4 . Data base

A. Doctors Table

Stores doctor profiles, including:

- Name
- Specialty
- Assigned patients

This allows linking each patient to a responsible doctor.

B. Patients Table

Contains:

- Personal details
- Medical ID
- Assigned doctor ID
- Linked IoT device ID

This enables tracking the medical journey of each patient.

C. Devices Table

Each patient is assigned a virtual IoT device:

- Device serial number
- Device type
- Patient ID

D. Vitals Table

This is the main high-frequency data table.

Stores every reading generated by the pipeline:

- Heart rate
- Oxygen
- Glucose
- Temperature
- Blood pressure

- Respiratory rate
- Timestamp
- Device ID

This allows for time-series trending and analysis.

E. Alerts_Log Table

When Spark detects abnormalities, the alert is stored here with:

- Alert ID
- Patient ID
- Alert type (oxygen / HR / glucose / etc.)
- Severity level
- Description
- Timestamp
- Status (handled / unhandled)

This acts as a medical event log.

F. Patient_Summary Table

A dynamic table containing:

- Latest vitals
- Average readings
- Most recent alerts
- Overall health score (optional)

This is used to quickly show doctors the overall condition of any patient.

4.2 Database Relationships

The relationships ensure proper mapping:

- One doctor → many patients
  A doctor supervises multiple patients.
- One patient → many vitals readings
  Vitals are continuously stored over time.

- One patient → many alerts
  Patients may trigger warnings frequently.
- One device → many readings
  The device continually streams data.

The database acts as the long-term memory of the system.

5. Research Results and Discussions

The implemented system successfully demonstrated real-time monitoring and alerting using simulated IoT data.
The Streamlit dashboard visualizes patient vitals with live updates, while the alerting mechanism immediately flags abnormal conditions.

Key results include:

- Continuous data flow between all layers (Kafka → Spark → PostgreSQL → Streamlit).
- Automatic anomaly detection and logging in the database.
- Interactive dashboard with summary
  and alert visualization.

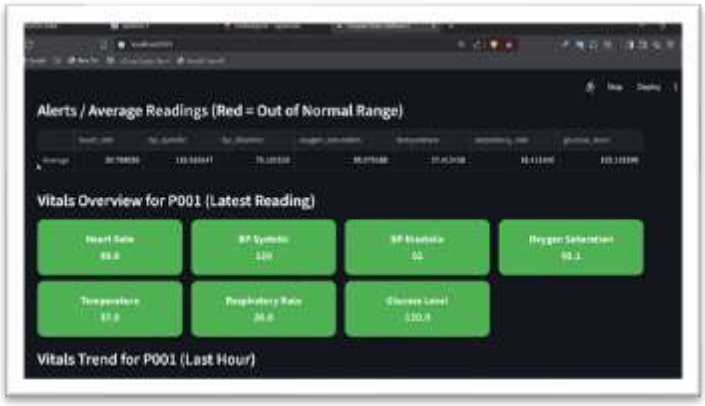

Fig 5 . Streamlit

Fig 6 . Streamlit

This validates the system's capability to act as a real-time decision support tool for hospitals.

6. Conclusions and Future Work

6.1 Conclusions

The project successfully achieved its objectives by designing and implementing a complete IoT healthcare data pipeline that simulates, processes, stores, and visualizes vital signs in real time. The integration of Kafka, Spark, PostgreSQL, and Streamlit proved effective for real-time data engineering applications in healthcare.

This work demonstrates the potential of IoT-driven pipelines in reducing delays in medical response and improving hospital efficiency.

6.2 Future Work

Future improvements may include:

- Integrating real IoT medical devices for live data collection.
- Expanding the anomaly detection module using machine learning models.
- Deploying the system on cloud platforms for scalability and high availability.
- Enhancing the dashboard with predictive analytics for proactive health monitoring.