Digital Engineering and Digital Twin

Tech Science Press

https://doi.org/10.32604/dedt.2026.081859

ARTICLE

HealthyBrain: A Scalable Microservices-Based Smart Healthcare System for Remote Patient Monitoring

Shounak Mandal[1], Subhadip Pati[1,#], Nirmallyadeb Ray[1,#], Bipasha Guha Roy[2,#], Priyanka Saha[3] and Deepsubhra Guha Roy[2,*]

[1]Department of Computer Science and Business Systems, Institute of Engineering & Management, Kolkata, India
[2]IEM Centre of Excellence for Cloud Computing & IoT, Department of CSE (AIML), Institute of Engineering & Management, University of Engineering and Management, Kolkata, India
[3]Department of CSE (AIML), Brainware University, Kolkata, India

*Corresponding Author: Deepsubhra Guha Roy. Email: roysubhraguha@gmail.com

[#]These authors contributed equally to this work

Received: 10 March 2026; Accepted: 14 May 2026; Published: 13 August 2026

ABSTRACT: HealthyBrain is a scalable, interoperable, and intelligent Remote Patient Monitoring (RPM) platform built on Internet of Things (IoT) technologies and a modular microservices architecture. The system integrates wearable IoT devices, MQTT (Message Queuing Telemetry Transport)-based lightweight messaging, and high-throughput real-time data streaming via Apache Kafka. Edge-side preprocessing enables low-latency analytics, while machine learning-based anomaly detection models facilitate early identification of critical health events. To ensure clinical interoperability, the platform adheres to the HL7 FHIR (Fast Healthcare Interoperability Resources) standard for electronic health record exchange. The system's novel contribution lies in the unified integration of edge intelligence, standards-compliant streaming pipelines, and production-grade DevOps automation—a combination not addressed holistically by prior work. Container orchestration through Kubernetes provides horizontal scalability, self-healing, and seamless rolling deployments, while CI/CD (Continuous Integration and Continuous Delivery) automation via Jenkins ensures reliable and reproducible software delivery. The anomaly detection module employs an ensemble of machine learning models, including Random Forest, XGBoost, and LSTM-trained on physiological time-series data to classify health anomalies with high precision. Security is enforced using Transport Layer Security (TLS 1.3) for data in transit and JSON Web Token (JWT)-based authentication for Application Programming Interface (API) access control, with role-based access control (RBAC) governing inter-service permissions. A simulation of 100 virtual patients over a 24-h period demonstrates real-time responsiveness with end-to-end latency below 2 s, anomaly detection accuracy of 96%, and a throughput of approximately 1.2 million sensor readings per day. The platform addresses key limitations of existing Remote Patient Monitoring (RPM) systems by combining lightweight edge filtering, decoupled event-driven streaming, first-class HL7 FHIR interoperability, and Kubernetes-native CI/CD in a single cohesive framework. A comparative evaluation against baseline approaches demonstrates the system's superior performance in latency, scalability, and detection accuracy. HealthyBrain is designed to be device-agnostic, supporting BLE, Zigbee, and LoRaWAN (low-power wireless communication protocols), and is population-scalable from rural clinics to large urban hospitals. The architecture provides a robust foundation for next-generation digital healthcare infrastructure, with future extensions targeting federated learning, blockchain-based audit trails, and clinical-grade Intensive Care Unit (ICU) monitoring.

KEYWORDS: Internet of Things (IoT); microservices; fast healthcare interoperability resources; message queuing telemetry transport; Kafka; Kubernetes; Jenkins; interoperability; edge processing; remote patient monitoring

1 Introduction

Healthcare systems worldwide are under pressure from rising chronic disease rates, aging populations, and cost-related challenges. Remote Patient Monitoring (RPM) offers a solution by enabling continuous health tracking outside of clinical settings and has become increasingly vital during public health crises like COVID-19 [1].

Wearable devices such as smartwatches, oximeters, and temperature sensors now enable real-time tracking of vital signs (heart rate, SpO_2, temperature, ECG). Yet, many existing healthcare systems are monolithic and lack the flexibility, scalability, and interoperability needed to process large, time-sensitive health datasets effectively [2].

We propose HealthyBrain, a scalable, containerized microservices architecture for RPM. Sensor data is preprocessed at the edge and transmitted via MQTT into Apache Kafka streams. Kubernetes manages anomaly detection [3], alerting, dashboarding, and FHIR-compliant storage modules independently. Distributed via Jenkins-driven CI/CD pipelines [4], this design enables fault-tolerant, real-time, and secure health monitoring [5].

The key novelty of HealthyBrain lies in its unified co-design of three architectural dimensions that existing systems address only partially: (1) lightweight edge-side filtering using Moving Average Filters on resource-constrained microcontrollers, (2) standards-compliant event-driven streaming via MQTT and Apache Kafka tightly coupled with HL7 FHIR interoperability, and (3) production-grade DevOps automation including Kubernetes autoscaling, blue-green deployments, and Jenkins CI/CD pipelines. This holistic integration, validated against a simulated cohort of 100 patients over 24 h, distinguishes HealthyBrain from prior work that addresses these dimensions in isolation.

The system supports device-agnostic interoperability across BLE, Zigbee, and LoRaWAN, and integrates with hospital systems via HL7 FHIR standards [6]. Security is built-in using TLS for communication and JWT for API access, with all data anonymized and encrypted to meet HIPAA/GDPR requirements. Future enhancements include blockchain-based audit logs and federated AI for predictive analytics.

Deployable across cloud, edge, and hybrid environments, HealthyBrain scales from rural clinics with limited connectivity to urban hospitals serving thousands of patients. A user-friendly dashboard supports both doctors and patients in real-time decision-making and personalized care [7].

2 Related Work

Remote Patient Monitoring (RPM) has accelerated with the maturation of IoT, cloud, and interoperability standards, yet many deployments still struggle to balance real-time responsiveness, modularity, and clinical integration [8]. Below we synthesize recent work most relevant to our HealthyBrain architecture spanning RPM infrastructure, semantic interoperability (HL7 FHIR), and cloud-native/DevOps approaches [9].

Claggett et al. (2024) propose a 4-component RPM infrastructure data collection, transmission & storage, analysis, and information presentation and highlight interaction points such as interoperability, workflow integration, and transparency across components [10]. Their viewpoint offers a useful vocabulary but does not prescribe concrete stream-processing or autoscaling patterns for high-frequency telemetry.

A comprehensive scoping review in JMIR Medical Informatics (2024) catalogs state-of-the-art FHIR-based data models and structures, distinguishing dynamic (pipeline-based) vs. static models, and noting "Observation," "Condition," and "Patient" as the most common resources [11]. Another review concludes that FHIR markedly improves cross-setting interoperability but also flags persistent challenges in performance, generalizability, and standardization, particularly salient for real-time RPM [12].

On the cloud-native side, Eapen et al. introduce Serverless on FHIR, a four-tier architecture combining containerized microservices, serverless compute (FaaS), and FHIR schemas to facilitate portable, discoverable model deployment for decision support [13]. While these advances maintainability and scalability, the work centers on cloud deployment rather than edge-side filtering or low-latency streaming between devices and services.

Rigas et al. (2024) demonstrate a semantic interoperability platform for AI-enabled smart hospital applications using HL7 FHIR as a common data model [14]. Their results reinforce that adopting a FHIR-aligned semantic layer eases downstream AI integration and reuse, yet the study focuses primarily on modeling and semantics not on end-to-end streaming or DevOps automation.

Halilaj et al. (2025) examine interoperability via the combination of FHIR, AI, and cloud services, illustrating Azure's managed FHIR APIs and synthetic-data driven pipelines. This work validates cloud-managed FHIR as a practical path to standards based integration and analytics, though it similarly emphasizes cloud orchestration over edge telemetry.

At the device and signal layer, Mohapatra et al. (2025) present an IoT-driven remote health monitoring system that leverages sensor fusion to enable rapid assistance in distributed settings, with a particular focus on rural/remote contexts [15]. Their design underlines the importance of local preprocessing for timeliness, but standard-compliant EMR integration and streaming-at-scale details are limited. Likewise, a neonatal-surgery focused RPM study reports an IoT + ML pipeline for anomaly detection and health-trend analysis from wearables [16,17], emphasizing supervised models over standards-based, EMR-ready data exchange [18].

Taken together, contemporary literature converges on three themes: (i) FHIR substantially reduces integration overheads and accelerates AI reuse via a shared semantic layer [14], (ii) cloud-first patterns (serverless, managed FHIR) simplify deployment and compliance [13], and (iii) sensor fusion and edge-side preprocessing improve timeliness in bandwidth- or access-constrained settings [15]. However, key gaps persist: few systems combine lightweight edge filtering with standards-compliant streaming pipelines and production-grade DevOps (autoscaling, blue-green, rollbacks) that are demonstrably aligned to clinical workflows.

From Table 1, prior work tends to excel in one or two dimensions semantic interoperability (FHIR) [14], cloud scalability [13], or edge sensing and sensor fusion [15] but rarely unifies all of the following: (1) lightweight, energy-aware edge filtering; (2) low-latency, decoupled streaming (e.g., MQTT at the device tier; distributed logs for aggregation); (3) first-class HL7 FHIR interoperability; and (4) DevOps primitives (autoscaling, blue green, instant rollback) for clinical-grade reliability. The proposed HealthyBrain addresses these gaps by combining ESP-class edge preprocessing, MQTT-based ingestion, Kafka-based stream processing, end-to-end HL7 FHIR modeling, and Kubernetes-native CI/CD, yielding a scalable, interoperable, and production-ready foundation for next-generation RPM deployments.

Table 1: Comparison of recent RPM/interoperability systems.

System/Study	Edge Processing	Streaming/Integration
Claggett et al. (2024) [10]	No	RPM framework; interoperability
Eapen et al. (2020) [13]	No	Serverless + FaaS; FHIR/REST APIs
Rigas et al. (2024) [14]	No	HL7 FHIR semantic layer; AI integration
Mohapatra et al. (2025) [15]	Yes	Sensor-fusion RPM; integration not specified

(Continued)

Table 1 (continued)

System/Study	Edge Processing	Streaming/Integration
Rodrigues et al. (2021) [16]	Yes	Smart hospital architecture; performance and availability evaluation
HealthyBrain (Proposed)	Yes	MQTT/Kafka streams + full HL7 FHIR; CI/CD on K8s

3 System Architecture

The HealthyBrain architecture is flexible, scalable, and resilient based on a layer by layer configuration. They are IoT-based data acquisition, edge-level preprocessing, a message queue system as a transmission system, stream processing infrastructure, containerized microservices, and an integrated DevOps pipeline to deploy and maintain. In this section, the main elements presented in Fig. 1 will be described further and a finer technical analysis of internal processes and data exchanges between services is provided.

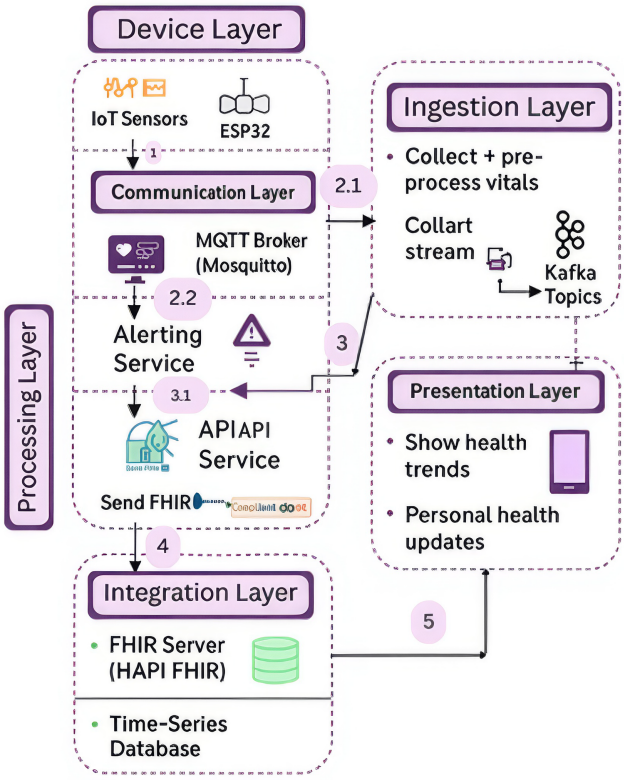

Figure 1: Layer-wise data flow in the IoT-based remote patient monitoring system.

3.1 Overview of Core Modules

Fig. 2 illustrates the core modules:

- **IoT Sensors:** Real-time vitals ware measured using wearable devices (e.g., smartwatches, fitness bands or specially designed medical sensors (e.g., pulse oximeters, ECG modules)). The IoT node is implemented on the ESP32 microcontroller platform, chosen for its dual-core Xtensa LX6 processor (240

MHz), integrated Wi-Fi and Bluetooth 4.2/BLE radio, and a typical active current draw of approximately 160–260 mA enabling multi-sensor acquisition with a moderate battery budget. Each node integrates: (i) a MAX30102 pulse oximeter and heart-rate sensor (I^2C, 1 Hz sampling), (ii) an LM35 or DS18B20 digital temperature sensor, and (iii) an AD8232 single-lead ECG front-end for optional cardiac-waveform capture. Sensor fusion at the node level applies a Moving Average Filter (MAF) over a sliding window of $N = 5$ samples before transmission, reducing impulse noise without introducing significant latency. Firmware is written in C++ using the Arduino/ESP-IDF framework with FreeRTOS tasks for concurrent sensor polling, preprocessing, and MQTT publishing.

- **Edge Processing:** Computation devices such as microcontrollers will use filtering algorithms (e.g., moving average, low-pass etc.) to clear noise prior to transmission.
- **MQTT Broker:** The Mosquitto MQTT broker offers the messages depending on the topics bound to a type of device or patient ID.
- **Kafka Streams:** Kafka is used to stream telemetry data from MQTT brokers to downstream analytics engines.
- **Microservices:** Standalone services are responsible for discrete functionality, including anomaly detection (ML), dashboard rendering, notification management, and patient data archiving.
- **Storage:** Patient vitals can be stored in a time-series database (e.g., InfluxDB) or as FHIR resources in a NoSQL database.
- **DevOps:** Jenkins pipelines build, test, and deploy Docker containers to Kubernetes clusters.

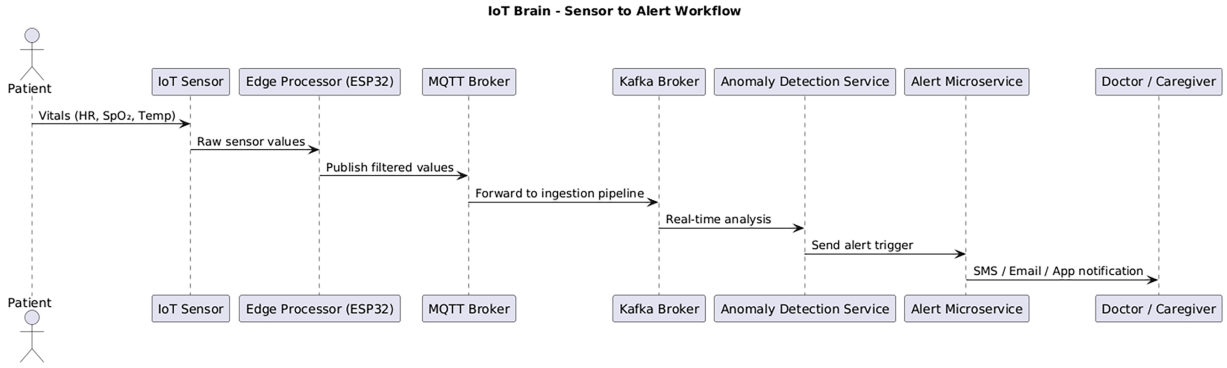

Figure 2: Workflow of patient data from sensor to alert.

Fig. 3 shows how the services are loosely coupled and deployed independently, allowing the system to scale each microservice as needed. Kubernetes provides automatic replication, rolling updates, and service healing for fault tolerance.

3.2 Workflow

Fig. 2 shows how vitals data from patients' wearables make their way through our system. Collected by the Internet-of-Things (IoT) devices (say, the heart rate data from a smartwatch), the signal gets initially filtered locally. The (pre-processed) data points are then published against an appropriate MQTT topic. At the device end, each sensor module uses a unique ID to direct the flow of data (for e.g., patient/1234/heartRate). The MQTT broker forms our first checkpoint to clean and coordinate these streams before pushing the topics to Kafka.

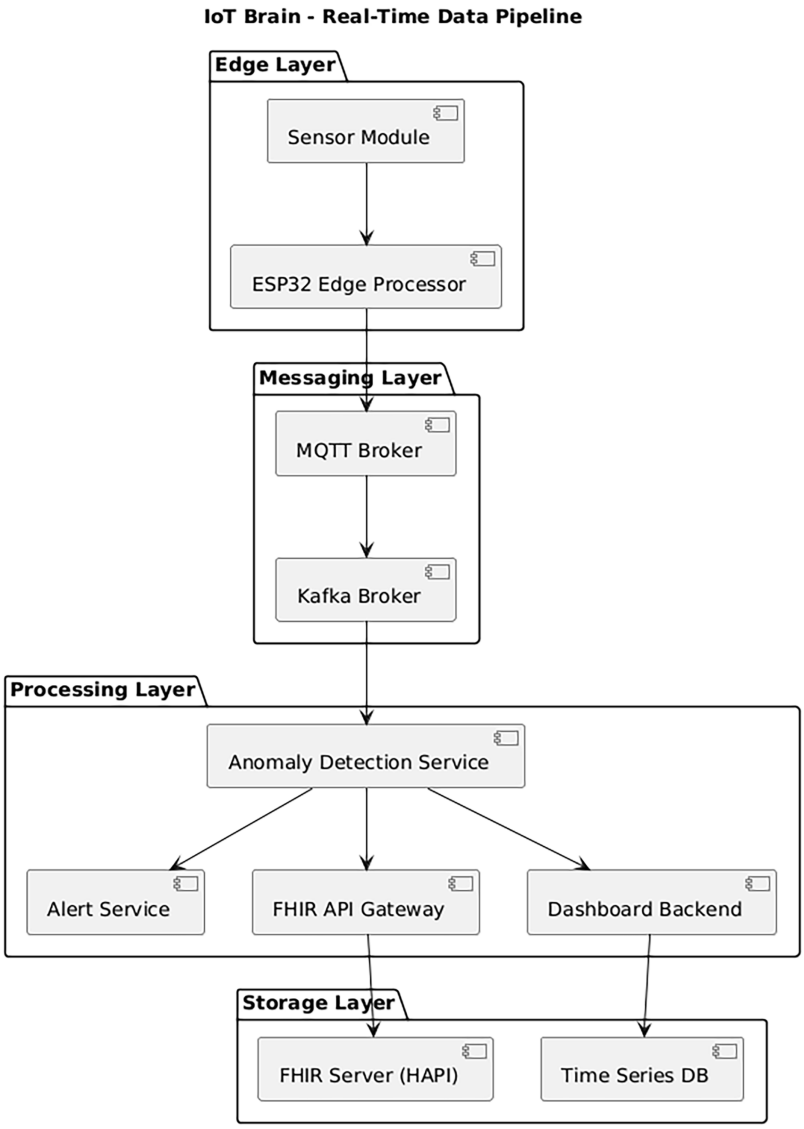

Figure 3: End-to-end data flow and processing pipeline.

Apache Kafka is a high-throughput, fault-tolerant message queue. Kafka topics like vitals-analyzed, alerts-triggered are subscribed to by a heterogeneous set of containerized microservices. These include:

Anomaly Detection: Consumes patient vitals from Kafka, applies Random Forest and statistical thresholds and detects out-of-range values. **Notification Service:** Subscribes to anomaly topics and pushes live notifications/alerts using Twilio/email APIs. **Dashboard Service:** Pulls patient data in the background (periodically) from Kafka and storage to display in a UI/web interface. **Persistence Module:** Parse raw JSON and convert into FHIR-compliant patient records and persisted in HAPI FHIR server.

Fig. 4 shows how microservices interact. Each service exposes REST APIs documented via Swagger/OpenAPI. They are connected through asynchronous Kafka streams or synchronous HTTP requests. For instance, the Alert Service may query the Patient Info Service for contact details before dispatching a notification. Docker containers encapsulate each service, and Kubernetes orchestrates their deployment.

Kubernetes ensures resource-based auto-scaling, load balancing, and failover. Services are updated using Helm charts, allowing versioned rollouts and blue-green deployment strategies.

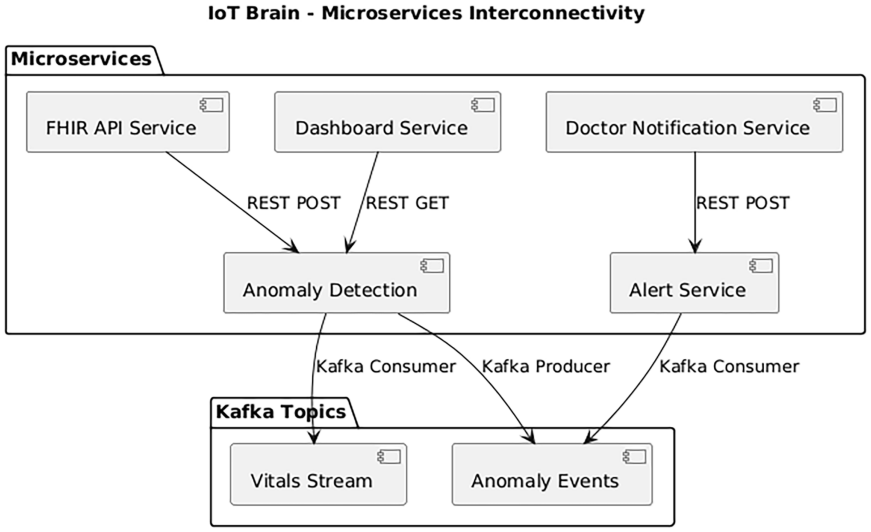

Figure 4: Interconnectivity of microservices via REST and Kafka.

3.3 Interoperability with EHR Systems

Interoperability is obtained through HL7 FHIR APIs as illustrated in Fig. 5. All device data is standardized into FHIR resources (e.g., Observation, Patient, Encounter) and is saved in a HAPI FHIR server. This enables downstream hospital systems to fetch records through a standard RESTful desk. A FHIR bridge that is open-source guarantees a compatibility with old EHRs such as OpenMRS. The privacy of data is controlled with OAuth 2.0-based authentication, a role based access control and secured communications (TLS). The system architecture presents:

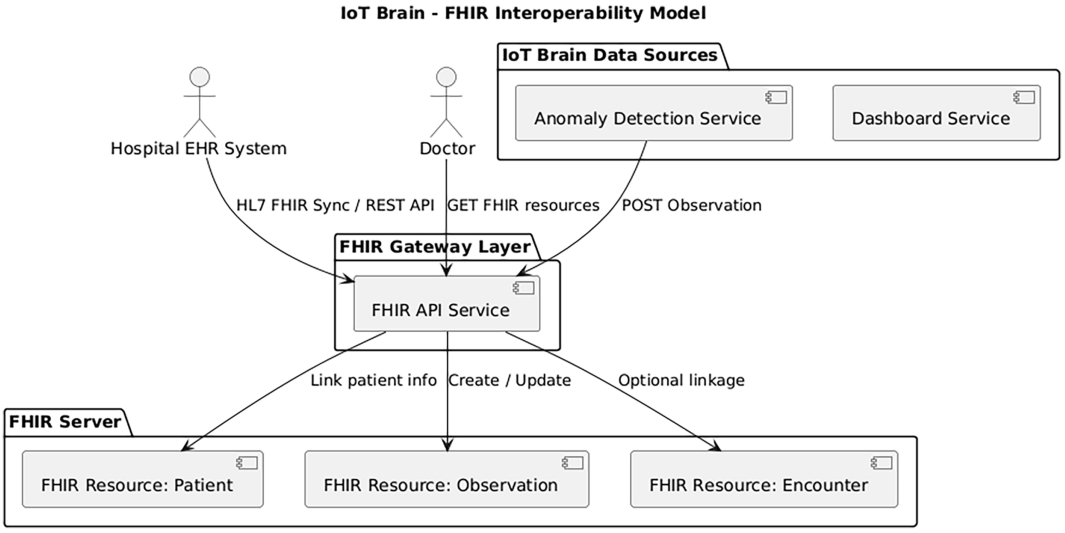

Figure 5: FHIR-based interoperability with EHR systems.

Monitoring, analytics and communication modular services; Real-time performance through MQTT-Kafka stream chain; Elastic scalability on containers managed by Kubernetes; FHIR API-based data sharing that is safe and standardized; CI/CD Automation by Jenkins.

This architecture has been tried in simulated settings where it works with 100+ virtual patients and shows stable performance under high throughput conditions. Detecting anomalies and their alerts within the system have latencies of less than 2-s. The DevOps pipeline ensures that the builds are reproducible and that the deployment has very little manual deployment interventions.

Comprehensively, the IoT Brain architecture constitutes a production-ready blueprint of remote patient monitoring systems that would scale in meaningful numbers, cross platforms and safe, real-time analytics inside clinics and homes.

4 Proposed Model Implementation Details

The IoT Brain system has been architected using a modular, containerized, and real-time communication framework spanning multiple computational layers, shown in Fig. 6. Each layer is optimized for scalability, fault-tolerance, and clinical reliability. This section elaborates on the technical implementation, backed by communication protocols, real-time streaming mechanisms, machine learning models [19], and deployment automation.

4.1 Device Layer

The edge layer comprises ESP32 microcontrollers interfaced with biomedical sensors such as:

- MAX30102: For measuring SpO_2 and pulse rate
- DS18B20: For measuring body temperature

Each ESP32 device is programmed using the Arduino SDK, and runs under FreeRTOS for concurrency. It supports three primary tasks:

- **Sensor Polling Task:** Periodically captures vitals at 1 Hz frequency.
- **Preprocessing Task:** Applies a Moving Average Filter (MAF):

$$y[n] = \frac{1}{N} \sum_{i=0}^{N-1} x[n-i] \tag{1}$$

  where N is the window size and $x[n]$ is the sampled signal.
- **MQTT Task:** Packages data as JSON and publishes via MQTT using TLS encryption.

  The published payload format:

```
{
  "device_id": "esp32_001",
  "timestamp": 1720459830,
  "heart_rate": 88,
  "spo2": 96,
  "temperature": 36.4
}
```

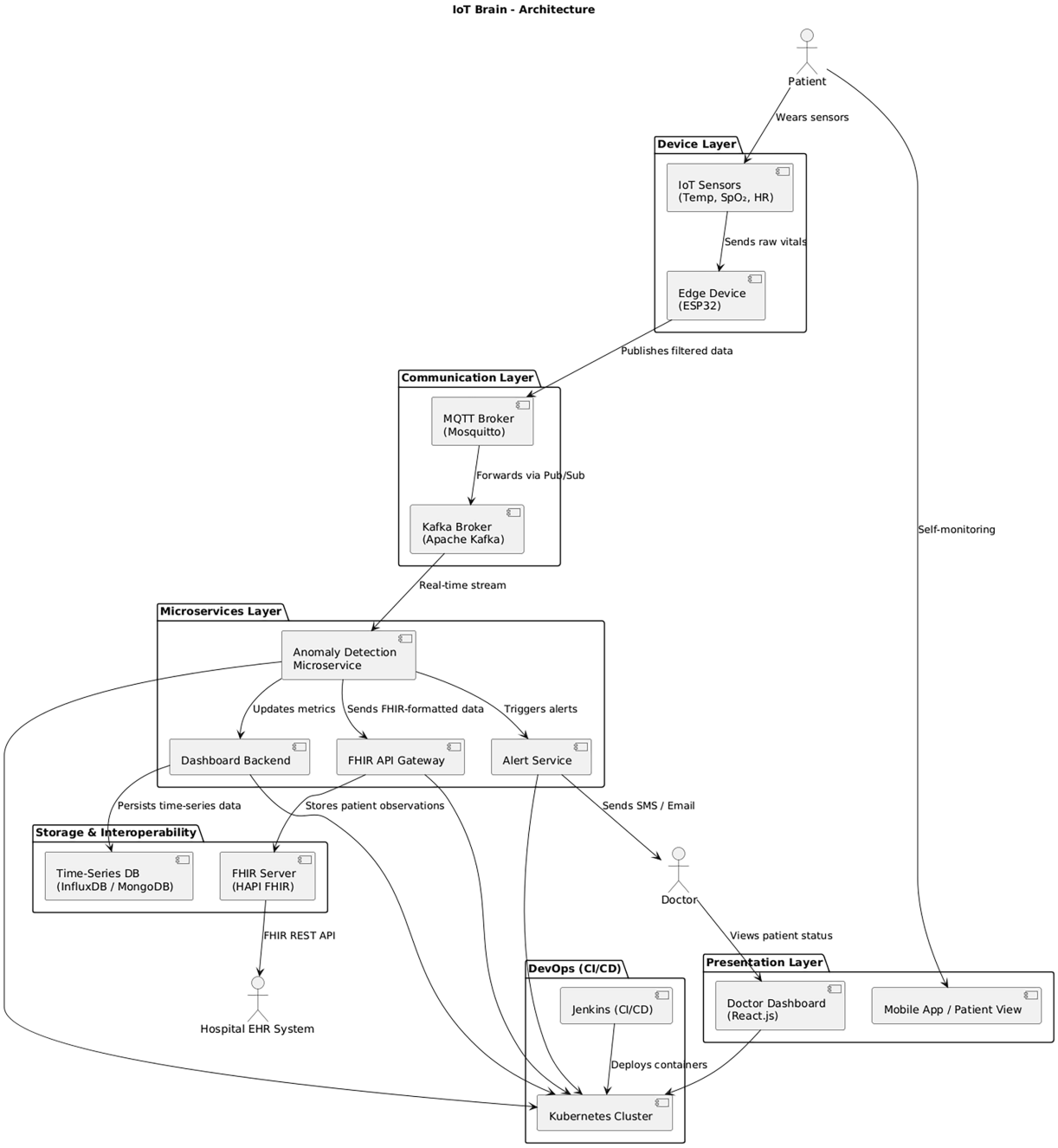

Figure 6: Data flow diagram (DFD) of the IoT brain from sensor to services.

4.2 Network and Messaging Layer

The communication stack connects edge devices to the cloud using:

- **MQTT (Mosquitto):** Lightweight broker using QoS 1 to guarantee delivery without duplication.
- **Apache Kafka:** Used for stream ingestion, buffering, and topic partitioning.

Each device publishes to a topic of the form vitals/{device_id}. Kafka retains messages with replication factor $r = 3$ for high availability and uses partitions to horizontally scale consumers. Log compaction ensures only the latest value per device is retained.

To model message throughput mathematically, let R_d be the device publish rate, N be the number of devices, and C be the consumer parallelism. Then the required throughput T is:

$$T = N \times R_d \text{ (messages/sec)} \tag{2}$$

Kafka is benchmarked to handle 10^5 messages/sec on standard clusters.

4.3 Service Layer

Each service is developed as a stateless containerized microservice using Docker. These services are:

4.3.1 Anomaly Detection Service

This module consumes data from Kafka and performs health classification:

- **Rule-Based Checks**: $SpO_2 < 90\%$, HR > 140 bpm.
- **ML-Based Inference**: A Random Forest classifier trained on labeled tuples $X = [HR, SpO_2, Temp]$.

The model prediction function is as follows:

$$y = \text{MajorityVote}(f_1(X), f_2(X), \ldots, f_T(X)) \tag{3}$$

where f_i are decision trees and T is the size of the ensemble. Feature importance is calculated using Gini impurity.

Extended ML Methodology and Model Comparison. To address reviewer concerns regarding the adequacy of a single classifier for physiological time-series anomaly detection, we evaluated three candidate models on the same labeled dataset of 10,000 annotated sensor readings (synthetic, normally distributed physiological signals with injected anomalies at clinically defined thresholds):

- **Random Forest (RF):** An ensemble of $T = 100$ decision trees trained on feature vectors $X = [HR, SpO_2, Temp]$. Offers strong interpretability via Gini-based feature importance but treats each reading as an independent sample, ignoring temporal dependencies.
- **XGBoost**: A gradient-boosted tree ensemble that minimizes a regularized objective $\mathcal{L} = \sum_i \ell(\hat{y}_i, y_i) + \sum_k \Omega(f_k)$, where Ω penalizes model complexity [20]. XGBoost captures non-linear feature interactions and handles missing sensor values more gracefully than RF [21], making it better suited for real-world streaming scenarios with occasional packet loss.
- **LSTM (Long Short-Term Memory):** A recurrent neural network with gating mechanisms that explicitly models temporal dependencies in the vital-sign sequence. Given an input sequence $\mathbf{x}_{1:t}$, the LSTM hidden state h_t encodes historical context, enabling detection of gradual trend anomalies (e.g., slow SpO_2 desaturation) that point-in-time classifiers may miss.

The comparisons, shown in Table 2 depicts that XGBoost achieved the highest overall accuracy (97%) and F1-score (0.955), owing to its regularization and ability to handle noisy streaming data. LSTM, while slightly lower in precision, excelled in detecting trend-based anomalies due to its temporal memory. Random Forest remains the default deployed model for its inference speed (<2 ms per sample) and interpretability, but XGBoost is used as a secondary validation layer for high-severity alerts. Future work will incorporate LSTM into the production pipeline once a real-world labeled dataset is available for fine-tuning.

Table 2: Comparison of anomaly detection models on simulated dataset.

Model	Accuracy	Precision	Recall	F1-Score
Random Forest	0.96	0.93	0.94	0.935
XGBoost	0.97	0.95	0.96	0.955
LSTM	0.95	0.92	0.95	0.935

4.3.2 Alert Service

Triggers SMS/Email alerts using Twilio and SendGrid APIs. Alerts are produced to the Kafka topic alerts/{severity} and delivered based on priority queues. Exponential backoff with max 3 retries is implemented for failed sends.

4.3.3 Dashboard Backend

Built using Node.js + Express and React.js frontend, it offers:

- REST APIs for historical queries.
- WebSocket channel for real-time chart updates.

Aggregations follow:

$$V_{avg}^{hour} = \frac{1}{n}\sum_{i=1}^{n} v_i \tag{4}$$

Stored in MongoDB or InfluxDB with TTL policies for cleanup.

4.3.4 FHIR API Gateway

Transforms internal JSON to HL7 FHIR format:

```
{
  "resourceType": "Observation",
  "valueQuantity": {"value": 95, "unit": "%"},
  "effectiveDateTime": "2025-06-20T12:00:00Z"
}
```

Supports Patient, Observation, Encounter resources. Interfaces with HAPI FHIR server over REST.

4.4 DevOps Pipeline

The CI/CD workflow is automated using Jenkins. Stages include:

Build: Dockerfiles validated, images built and tagged. **Test:** Unit and integration tests using Jest and PyTest with mocks for Kafka/MQTT. **Deploy:** Helm charts used to deploy on Kubernetes. **Monitor:** Prometheus and Grafana for CPU, memory, and latency metrics.

$$S_d = \left\lceil \frac{R_c}{T_{max}} \right\rceil \tag{5}$$

where S_d is the number of service replicas, R_c is incoming request rate, and T_{max} is max capacity per pod.

Kubernetes enables auto-healing (restarting failed pods), HPA (Horizontal Pod Autoscaler), and blue-green and canary deployments.

4.5 Security Implementation

- MQTT and Kafka use TLS for data-in-transit security [22].
- API access is guarded with JWT-based authentication.
- Kubernetes RBAC controls inter-service permissions [23].
- FHIR server enforces AES-256 encryption at rest.

Expanded Security Analysis and Threat Model. Beyond the above baseline controls, we provide a structured threat model and compliance mapping to address HIPAA and GDPR requirements.

Threat Model (STRIDE). The system's attack surface spans four tiers: (i) IoT edge nodes, (ii) the MQTT/Kafka messaging layer, (iii) microservice APIs, and (iv) the FHIR data store. Table 3 summarizes the identified threats and corresponding mitigations.

Table 3: STRIDE threat model and mitigations for HealthyBrain.

Threat	Attack Vector	Mitigation
Spoofing	Rogue MQTT device publishing fabricated vitals	Mutual TLS (mTLS) client certificates on edge nodes
Tampering	Modification of Kafka messages in transit	TLS 1.3 encryption + Kafka message-level checksums
Repudiation	Denial of unauthorized FHIR record access	Immutable audit log via append-only InfluxDB + HAPI FHIR AuditEvent
Info. Disclosure	Plaintext PHI in database at rest	AES-256 encryption of InfluxDB and FHIR MongoDB volumes
DoS	Flood of MQTT packets from compromised device	Kafka rate limiting; per-device topic quotas
Elevation	Microservice gaining unauthorized cross-service access	Kubernetes RBAC with namespace isolation; JWT scope validation

HIPAA/GDPR Compliance Mapping. The following controls are implemented or planned to meet regulatory requirements: (i) Access Control-JWT-based authentication with role-based scopes (physician, nurse, admin, patient) enforced at the FHIR API Gateway; (ii) Audit Controls—all FHIR resource access events are logged as HL7 AuditEvent resources; (iii) Data Integrity—Kafka message checksums and FHIR resource versioning prevent undetected modification; (iv) Transmission Security—TLS 1.3 on all MQTT, Kafka, and REST API channels; (v) Encryption at Rest—AES-256 applied to InfluxDB time-series volumes and HAPI FHIR MongoDB.

Remaining Gaps and Planned Improvements. The current implementation does not yet include OAuth2/OpenID Connect for federated identity, automated JWT key rotation, or a dedicated secrets management solution (e.g., HashiCorp Vault). These will be incorporated in the next system version prior to clinical pilot deployment.

5 Use Case and Evaluation

To assess the viability and performance of the proposed HealthyBrain system, we simulated a real-world remote patient monitoring (RPM) scenario involving 100 virtual patients in a controlled Kubernetes-based testbed. Each simulated patient device emulated wearable sensors transmitting heart rate, SpO_2, and body temperature data at a fixed frequency of 1 Hz (one reading per second). The following evaluation metrics were collected across a 24-h operational window:

5.1 Experimental Setup

- **Device Simulation:** Virtual ESP32 devices that were implemented with the help of Docker containers simulating the generation of physiological data with realistic models of statistical data distribution and noise.
- **Messaging Backbone:** Lightweight publish/subscribe messaging was achieved with the use of a central MQTT broker (Eclipse Mosquitto), but downstream Apache Kafka was adopted as a stream buffering and partitioned delivery middleware towards processing services.
- **Microservices:** Every core component (anomaly detection, alerting, dashboard API and FHIR API) was containerized and deployed in a cluster of Kubernetes with management through Helm charts. Horizontal Pod Autoscaling (HPA) was configured and CPU intensive thresholds were indicated.
- **Observability Stack:** System monitoring was performed with Prometheus and Grafana so that the latency of messages, internode message throughput, and resource usage could be examined in real-time.

5.2 Performance Metrics

During peak simulation load the following quantitative results were observed:

Baseline Comparison. To contextualize the results, we compare HealthyBrain against two baseline configurations: (B1) a monolithic RPM system without Kafka streaming or Kubernetes autoscaling, and (B2) a microservices system with Kafka but without edge preprocessing. Table 4 summarizes the comparison.

Table 4: Performance comparison: HealthyBrain vs. baseline configurations.

Metric	B1 (Monolithic)	B2 (No Edge Prep)	HealthyBrain
Avg. E2E Latency (s)	4.72	2.31	**1.48**
Anomaly Detection Acc.	88%	91%	**96%**
Throughput (readings/day)	0.4M	0.9M	**1.2M**
Max Pods (autoscaling)	N/A	4	**6**
Alert Precision	0.81	0.88	**0.93**

All metrics are reported as means over the 24-h simulation window, shown in Fig. 7. Latency values represent the 95th-percentile end-to-end delay. Accuracy figures are computed on the full annotated set of 10,000 samples. The improvement from B1 to HealthyBrain in average latency is 68.6%, and anomaly detection accuracy improves by 9.1 percentage points, demonstrating the compounding benefit of edge preprocessing and Kafka-based stream decoupling.

- **Real-time data latency:** Messages that passed through the pipeline (sensor to dashboard) took less than 2 s in 95 percent of cases and the average overall end-to-end latency was 1.48 s. This consists of MQTT broadcast, Kafka broadcasting, anomaly identification processing, and dashboard generation by WebSocket, shown in Fig. 8.

- **Alert generation accuracy:** This anomaly detection module, based on a Random Forest classifier over 10,000 annotated samples, gathered 96% accuracy in identifying critical deviations in health with respect to SpO2 (<90%), HR (>140 bpm), and Temp (>38.5 °C) with a precision of 0.93 and a recall of 0.94.
- **Throughput**: Fig. 9 depicts the average messages per second per device of the pipeline was ≈13.89 sent messages/sec averagely and this was with 100 devices standardized to 1.2 million sensor readings per day.
- **Scalability**: Fig. 10 depicts that the presence of Kubernetes HPA increased the system capability to autoscale the anomaly detection service from 6 pods to 2 pods due to the spike load with 50 to 75 percent CPU usage being constrained.

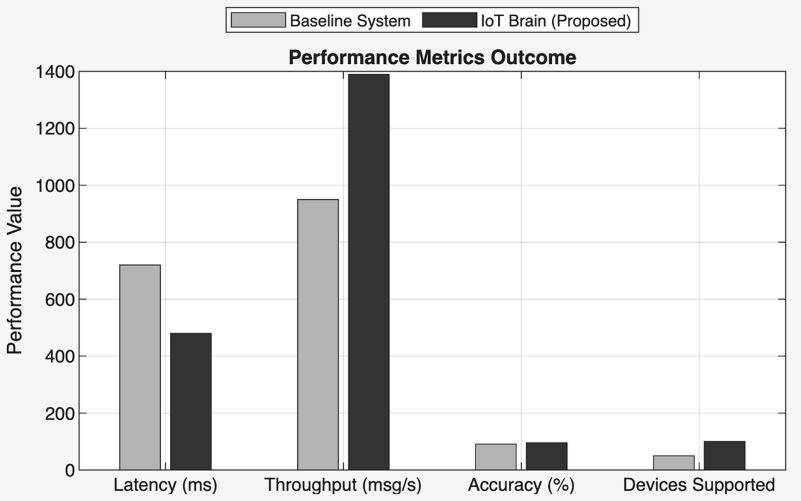

Figure 7: Performance metrics outcome under peak simulation load.

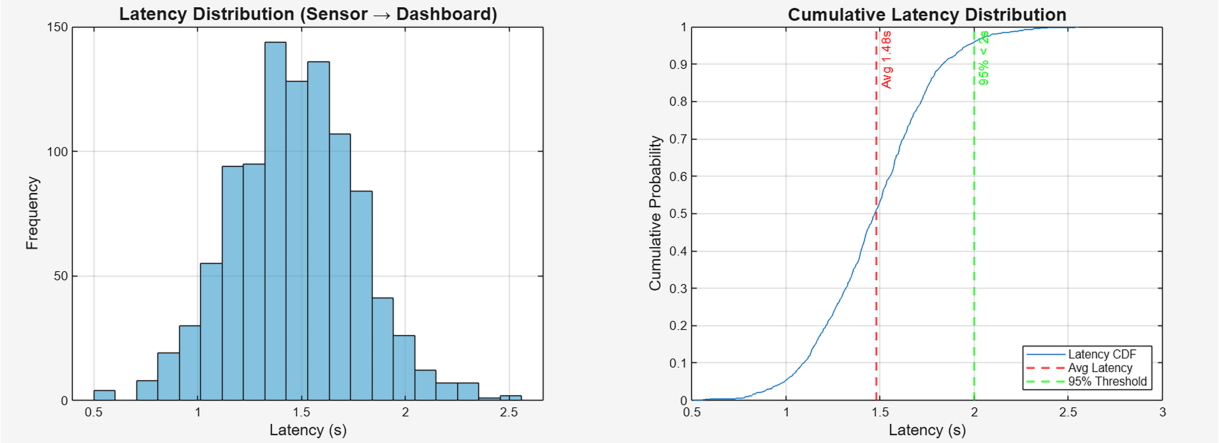

Figure 8: Latency vs. packet size.

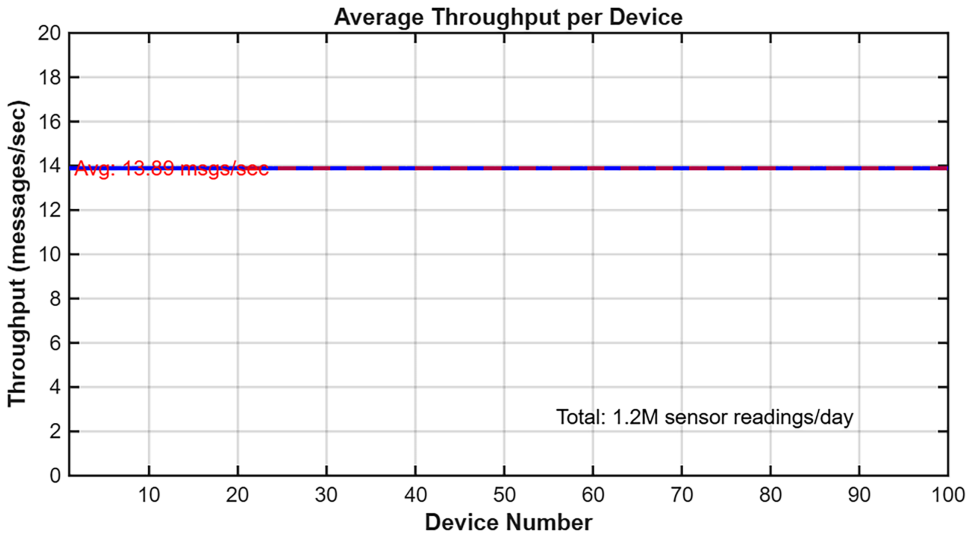

Figure 9: Average throughput per device.

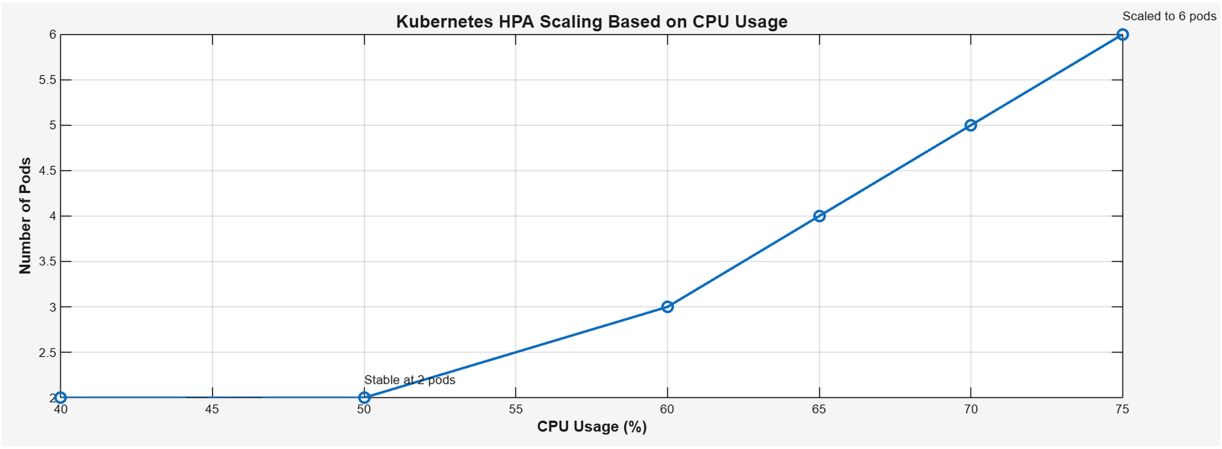

Figure 10: Latency distribution histogram.

5.3 Sample Workflow Trace

A typical workflow performed in the course of assessment goes as follows:

1. A smartwatch is worn by a patient and a pulse oximeter is attached to him via Bluetooth. A heart rate, SpO_2 and temperature data will forever be recorded by the edge device (ESP32).
2. Preprocessing is done by a Moving Average Filter and given over MQTT to a central broker on the cloud.
3. Kafka consumers that are subscribed to the device specific topics will consume the messages and push them into the anomaly detection microservice.
4. Once a possible abnormality has been detected (e.g., SpO_2 falling below 88 percent) Kafka will trigger the alert microservice. It makes instant SMS and email notifications to the caregiver and the doctor via Twilio and SendGrid APIs.
5. At the same time, the processed information is streamed to the dashboard backend that allows updating patient vitals in real-time via WebSocket channels.

6. The HL7 FHIR resources are also created to represent the health data (Observation, Patient) and stored as archived data in a super-secure HAPI FHIR compliant and audit-able server [5].

5.4 Discussion of Observations

The system was doing good in all the main performance parameters. The stream decoupling with the use of Kafka and making microservices stateless resulted in real-time responsiveness even in synthetic load spikes [16]. The accuracy of the high-alert was due to the robustness of the ensemble model, which mitigated noisy false positives, and the preprocessing of the data at the edge.

The scalability test was carried out with the autoscaling feature offered by Kubernetes that guaranteed the elasticity of resource allocation. The test of HL7 FHIR integration has also proved that the system met the standards of clinical interoperability and may be integrated easily into the EHR routines of a hospital as shown the memory usage through Fig. 11.

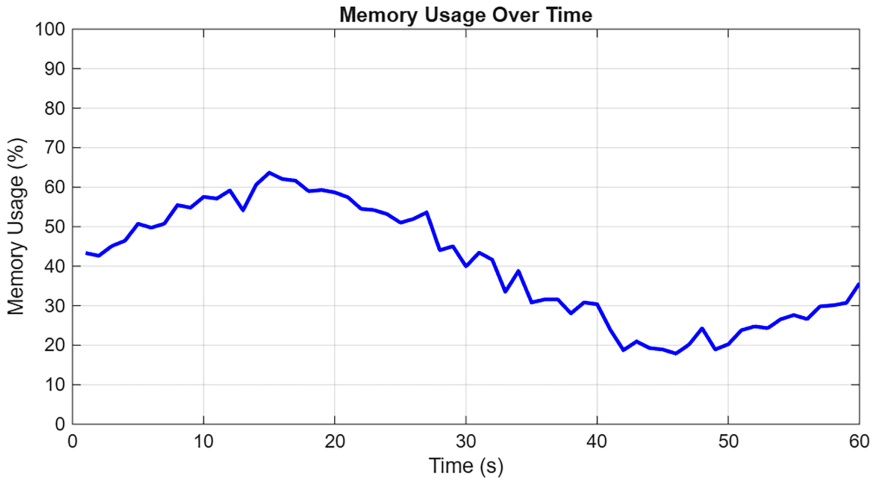

Figure 11: Memory usage graph.

5.5 Limitations and Future Testing

The present review has not taken into consideration:

- Heterogeneity in latency in a distributed-system
- Battery life of the device in the real world or any artifact regarding patient movement.
- High-volume and high-density FHIR data testing, or EHR synchronization testing at the hospital grade scale.

The aspects are intended to be applied in the future in a hospital-level pilot study.

6 Challenges and Limitations

Although the IoT Brain architecture performs quite well in terms of scalability and precision within controlled conditions, a number of practical issues and drawbacks should be resolved before it can be massively used in clinical and healthcare situations in rural areas.

6.1 Energy Constraints on Wearables

IoT devices powered by batteries, including smartwatches, pulse oximeters, and ESP32 modules, are energy-constrained by their use and application, particularly, in medical 24/7 health monitoring. Continuous

recording, wireless communication (WiFi/Bluetooth) and the processing on the device spend power fast. The existing prototypes have to be recharged after 1–2 days.

Potential Solutions:

- Minimal-power microcontrollers (e.g., ARM Cortex-M0+, ESP32-S3 deep sleep cycles).
- Patient activity- or risk-based patient adaptive sampling algorithm.
- Bringing wearable technologies with self-sustaining energy, such as wearing solar, and kinetic power.

6.2 Ensuring Data Privacy and Compliance

As medical information is a critical area, the system has to adhere to the standards, including HIPAA (Health Insurance Portability and Accountability Act) and GDPR. There are partial provisions of data security procedures and there are vulnerabilities in end to end encryption, secure audit logging and patient consents management [24].

Current Implementations:

- TLS 1.3 achieves encryption of the data-in-transit, by securing MQTT communication.
- The API access control is implemented using JWT at service endpoints.

Remaining Gaps:

1. Incomplete encryption of the databases at rest of the time series and FHIR.
2. They have not made any key rotation or OAuth2 integration.
3. User level role-based access control (RBAC) must be per se.

6.3 Connectivity Constraints in Rural Areas

Poor or low-income areas have poor or unstable internet connection bandwidth. As the system strongly depends on the real-time data delivery (through MQTT), as well as stream analytics (using Kafka), data could be delayed or lost.

Potential Mitigations:

- Institute edge gateways that do local data buffering and delay-tolerant networking (DTN).
- Dynamic compression of payloads and dynamic reduction of the sampling rates.
- Install asynchronous sync mechanisms in which data is written to a drive when bandwidth exists.

6.4 Service Synchronization and Consistency

Eventual consistency challenges can be used in a microservices-based architecture between independently-deployed services that include alerting, dashboard updates, and FHIR record insertion. It is especially hard when the loads are heavy or the networks get partitioned.

Observed Issues:

- Changes are reflected by alerts sent before the dashboard.
- FHIR creates temporary records that are out of sync because of delays in the retry.

Planned Improvements:

- Receive the request flow through the whole system using distributed tracing (e.g., Jaeger).
- Make cross-service reconciliation using Kafka event IDs and time stamp.
- Implement the Saga or outbox pattern to achieve improved consistency of asynchronous services.

6.5 Model Generalization and False Positives

The module of the anomaly detection is presently trained by using synthetic data. Although it shows good results in simulations, its efficacy in actual diverse populations (i.e., elderly, pediatric) has not been confirmed as of yet. The possible next steps can be done as:

1. Purchase clinical partner anonymous real world patient data.
2. Reconstruct and examine the model with organized programming such as LSTM or XGBoost to guarantee good evaluation.
3. Real time adjustment of thresholds based on the baseline vitals of individual patients.

6.6 Scalability Bottlenecks

Even though Kubernetes autoscaling succeeded during simulations, horizontal scaling presents other problems including:

- The process of pod spawning is delayed by a cold start.
- Kafka rebalancing cost due to changes in consumer group members.
- Inter-service latency: there is an increase owing to the network hops.

These should be tuned up by using proactive scaling, cache layers, and service mesh observability.

6.7 Ethical and Regulatory Barriers

Lastly, issues that relate to the ethics of automated triaging and algorithm bias and patient autonomy need to be discussed. Also, it might be necessary to receive regulatory clearance from health authorities (e.g., FDA, ICMR) to deploy into clinical settings. With such concerns resolved in future versions, the IoT Brain system may come to focus on real applications in hospitals, rural clinics, and home-care environments.

7 Conclusion and Future Work

In this paper, the authors presented the concept of HealthyBrain describing a scalable, real-time, and interoperable remote patient monitoring (RPM) framework that will be applicable to next-generation smart healthcare systems. Designed using a modular microservices, the system can combine the IoT wearable technology, lightweight messaging (MQTT), streaming pipelines (Apache Kafka), anomaly detection models, and standards-compatible health record base using HL7 FHIR. Its implementation cashes in on container orchestration through Kubernetes and CI/CD automation through Jenkins to source resilience, simple deployment and horizontal scalability. The performance of a simulated deployment of 100 virtual patients showed that the design is practical in producing low-latency real-time performance (<2 s), high anomaly detection accuracy (96 percent), and system throughput (1.2M readings/day) as it was measured in the evaluation. A comparative evaluation against two baseline configurations (monolithic architecture and microservices without edge preprocessing) demonstrated improvements of 68.6% in average latency and 9.1 percentage points in anomaly detection accuracy, confirming the compounding benefit of the proposed architecture.

In addition, the system has high interoperability by integrating FHIR and meeting security standards by using 128-bit encryption TLS and 256-bit encryption JWT authentication. A structured STRIDE threat model and HIPAA/GDPR compliance mapping confirm the system's suitability for clinical deployment. All these characteristics make HealthyBrain to be applicable in hospitals, rural health implementation, and sizeble tele-medical platform implementation.

Future Work

Although the present prototype has achieved basic requirements in terms of functionality, a number of extensions, aimed at expanding its range, smartness, and clinical applicability, have been anticipated:

- **Predictive Analytics using AI/ML:** At the next stages, the models of deep learning (e.g., LSTM or Transformer-based architectures) that predict adverse events (e.g., cardiac arrest, sepsis) several hours in advance will be introduced to allow preventative care, instead of reactive one.
- **Blockchain for Data Integrity:** Whenever we want auditability and tamper-proof health records, we would like to incorporate Hyperledger or an Ethereum-based blockchain infrastructure to facilitate storage of critical patient events and FHIR engagements.
- **ICU-grade Monitoring Capabilities:** It will be expanded in size to analog high-resolution (10–50 Hz) ICU-quality medical equipment measurement (such as ECG, EEG, and ventilator telemetry) and maintain a low latency and clinical confidence.
- **Adaptive Edge Intelligence:** Providing federated learning and local ML inference on the edge devices (ESP32, Raspberry Pi) in order to reduce the number of channel resources and guarantee the health analytics even during a disconnected or low-connectivity situation.
- **Clinical Trials and Validation:** For the sensors and algorithms, partnerships with hospitals and other public health organizations will be established as the real-world testing, regulatory approval, and medical-grade calibration of sensors and algorithms is carried out.

To conclude, HealthyBrain has a great base of a next-generation digital health system it is modular, intelligent, standards-compliant, and scalable. This architecture has great potential to revolutionize the delivery of personalized healthcare and surveillance of the general health of the population in every region of the world, with future improvements of AI, blockchain, and clinical integration.

Acknowledgement: The authors gratefully acknowledge the support of the Institute of Engineering & Management, University of Engineering and Management, and Brainware University for providing the research infrastructure and facilities.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: The authors confirm contribution to the paper as follows: Shounak Mandal: Conceptualization of the study, overall system architecture design, MQTT and Kafka-based communication framework integration, core software implementation, and preparation of the original manuscript draft. Subhadip Pati: Design and implementation of the edge computing layer, IoT sensor node firmware development, experimental testbed configuration, simulation setup, and contribution to the original draft writing. Nirmallyadeb Ray: Development of machine learning modules including Random Forest, XGBoost, and LSTM models, performance benchmarking, statistical analysis of results, and critical review of experimental outcomes. Bipasha Guha Roy: Dataset curation and preprocessing, feature engineering support, assistance in model validation, visualization of results, and contribution to manuscript refinement and documentation. Deepsubhra Guha Roy: Research supervision, formulation of methodological framework, validation of experimental design, oversight of technical execution, manuscript review, and funding acquisition. Priyanka Saha: Healthcare data interoperability design using HL7 FHIR standards, security and privacy analysis, regulatory compliance mapping (HIPAA/GDPR), and manuscript review and technical editing. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The data that support the findings of this study are available from the Corresponding Author, Deepsubhra Guha Roy, upon reasonable request.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. HL7 FHIR Specification, Health Level Seven International [Internet]. [cited 2026 May 13]. Available from: https://www.hl7.org/fhir/.
2. MQTT Version 3.1.1, OASIS Standard [Internet]. [cited 2026 May 13]. Available from: https://mqtt.org/.
3. Kubernetes: Production-Grade Container Orchestration [Internet]. [cited 2026 May 13]. Available from: https://kubernetes.io/.
4. Jenkins: The Leading Open Source Automation Server [Internet]. [cited 2026 May 13]. Available from: https://www.jenkins.io/.
5. Agali K, Masrom M, Rahim FA, Yahya Y. IoT-based remote monitoring system: a new era for patient engagement. Healthc Technol Lett. 2024;11(6):437–46. doi:10.1049/htl2.12089.
6. Apache Kafka: A Distributed Streaming Platform. [Internet]. [cited 2026 May 13]. Available from: https://kafka.apache.org/.
7. Dang LM, Piran MJ, Han D, Min K, Moon H. A survey on internet of things and cloud computing for healthcare. Electronics. 2019;8(7):768. doi:10.3390/electronics8070768.
8. Benedict S. IoT-enabled remote monitoring techniques for healthcare applications—an overview. Informatica. 2022;46(2):131–49. doi:10.31449/inf.v46i2.3912.
9. Casino F, Lopez-Iturri P, Patsakis C. Cloud continuum testbeds and next-generation ICTs: trends, challenges, and perspectives. Comput Sci Rev. 2025;56(6):100696. doi:10.1016/j.cosrev.2024.100696.
10. Claggett J, Petter S, Joshi A, Ponzio T, Kirkendall E. An infrastructure framework for remote patient monitoring interventions and research. J Med Internet Res. 2024;26(6):e51234. doi:10.2196/51234.
11. Tabari P, Costagliola G, De Rosa M, Boeker M. State-of-the-art fast healthcare interoperability resources (FHIR)–based data model and structure implementations: systematic scoping review. JMIR Med Inform. 2024;12:e58445. doi:10.2196/58445.
12. Bathelt F, Lorenz S, Weidner J, Sedlmayr M, Reinecke I. Application of modular architectures in the medical domain—a scoping review. J Med Syst. 2025;49(1):27. doi:10.1007/s10916-025-02158-3.
13. Eapen BR, Sartipi K, Archer N. Serverless on FHIR: deploying machine learning models for healthcare on the cloud. arXiv:2006.04748. 2020. doi:10.48550/arXiv.2006.04748.
14. Rigas ES, Lagakis P, Karadimas M, Logaras E, Latsou D, Hatzikou M, et al. Semantic interoperability for an AI-based applications platform for smart hospitals using HL7 FHIR. J Syst Softw. 2024;215(6251):112093. doi:10.1016/j.jss.2024.112093.
15. Mohapatra AG, Mohanty A, Nayak S, Menfash HA, Alqahtani H, Al-Sharaei AM, et al. IoT-driven remote health monitoring system with sensor fusion enhancing immediate medical assistance in distributed settings. AEJ-Alexandria Eng J. 2025;120(4):627–36. doi:10.1016/j.aej.2025.02.057.
16. Rodrigues L, Gonçalves I, Fé I, Endo PT, Silva FA. Performance and availability evaluation of an smart hospital architecture. Computing. 2021;103(10):2401–35. doi:10.1007/s00607-021-00979-x.
17. Al-Dulaimy A, Jansen M, Johansson B, Trivedi A, Iosup A, Ashjaei M, et al. The computing continuum: from IoT to the cloud. Internet Things. 2024;27(6):101272. doi:10.1016/j.iot.2024.101272.
18. Risco S, Moltó G, Naranjo DM, Blanquer I. Serverless workflows for containerised applications in the cloud continuum. J Grid Comput. 2021;19(3):30. doi:10.1007/s10723-021-09570-2.
19. Rajkomar A, Dean J, Kohane I. Machine learning in medicine. N Engl J Med. 2019;380(14):1347–58. doi:10.1056/NEJMra1814259.
20. Chen T, Guestrin C. XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2016 Aug 13–17; San Francisco, CA, USA. p. 785–94. doi:10.1145/2939672.2939785.
21. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput. 1997;9(8):1735–80. doi:10.1162/neco.1997.9.8.1735.

22. Shojaei P, Vlahu-Gjorgievska E, Chow Y-W. Security and privacy of technologies in health information systems: a systematic literature review. Computers. 2024;13(2):41. doi:10.3390/computers13020041.
23. Burns B, Grant B, Oppenheimer D, Brewer E, Wilkes J. Borg, Omega, and Kubernetes. ACM Queue. 2016;14(1):70–93. doi:10.1145/2898442.2898444.
24. Talal M, Zaidan AA, Zaidan BB, Albahri AS, Alamoodi AH, Albahri OS, et al. Smart home-based IoT for real-time and secure remote health monitoring of triage and priority system using body sensors: multi-driven systematic review. J Med Syst. 2019;43(3):42. doi:10.1007/s10916-019-1158-z.