NextGen Scientific Publication
American International Journal of Computer Science and Technology

Volume 8 Issue 3, Pg. No. 73-84, AIJCST-V8I3P107, 2026
https://doi.org/10.63282/3117-5481/AIJCST-V8I3P107

Original Article

Medical Device Development with Automated DevOps Environments and Regulatory Compliance

***Rahul P. Mahajan**

Software/Firmware Architect, Research and Development Department, Healthcare and Medical Device Development Industry, College Engineering, Pune, India.

Abstract:

The demand for reliable, safe, and compliant software-driven medical device development environments has increased with the changing dynamics in the field of healthcare technologies. It also has the need for patient safety and software reliability. The typical medical device software development process may involve a number of problems, including multiple testing environments, slow deployment cycles, lack of visibility, and compliance issues. A medical monitoring framework was established to overcome these challenges, which allowed for the automation of all aspects of DevOps, such as continuous integration, continuous deployment, medical services containerization, automated testing, vulnerability assessment, real-time monitoring and compliance auditing. It utilises some of the most important technologies, including Docker, Prometheus, Grafana, GitHub Actions, and cloud-native deployment methods, to improve the efficiency of software delivery and monitoring. Results of the experiment showed effective utilization of the infrastructure with an average of 1.99% CPU consumption, 376.2 MB memory consumption and application programming interface (API) response time of 1.9–3.1 ms for eight containers deployed. The system alerted 220 alerts and resolved 111 alerts (alert resolution rate = 50.5%) for continuous patient monitoring and clinical intervention. Implementation Mapping with recognised medical and cybersecurity standards provided an opportunity for regulatory alignment. Improvements in automation and observability have been achieved, but further work is needed to address issues of interoperability and more widespread clinical validation.

Keywords:

Medical Device Development, Devops, Regulatory Compliance, Healthcare Monitoring, Medical Software Lifecycle.

Article History:

Received: 26.03.2026

Revised: 29.04.2026

Accepted: 06.05.2026

Published: 14.05.2026

1. Introduction

Healthcare systems are increasingly relying on software embedded in medical devices for diagnosis, continual patient monitoring, therapeutic interventions, and clinical decision-making in hospital, home-care, and connected health care systems [1][2][3]. With the rapid development of digital healthcare technologies, there has been a surge in the adoption of intelligent medical systems that are increasingly integrating sensors, sensor networks, cloud computing, real-time analytics, and automated monitoring features to enhance patient outcomes and healthcare efficiency[4][5]. Today, medical devices like wearable sensors, remote monitoring systems, infusion systems and diagnostics generate important patient data that must be reliably processed and readily available for operations. But as software has become more complex, issues surrounding system reliability, deployment efficiency, cyber security risks and patient safety have grown in significance. Medical device software differs from traditional software applications in its environment, which is highly

Copyright by the Author(s). This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-NC-ND 4.0) License (https://creativecommons.org/licenses/by-sa/4.0/)

regulated and safety-critical, and failures or slow software updates, or operating vulnerabilities could have an impact on clinical outcomes [6][7]. This means that development environments are needed to support software quality, software transparency, and software compliance in the lifecycle of medical technology software [8].

DevOps has become a new software engineering methodology which brings together software development and information technology operations to facilitate software development and delivery efficiency, collaboration and automation across the software lifecycle[9]. DevOps focuses on continuous integration, continuous deployment, automated testing, infrastructure-as-code, monitoring, and speedy feedback mechanisms to deliver reliable software with less downtime[10][11]. DevOps allows for faster software releases, better defect management, better scalability, and operational consistency through technologies like containerization, microservices, Docker, automated build pipelines and cloud-native platforms [12][13]. DevOps practices can offer opportunities in the healthcare industry as well to speed up software updates, enhance testing coverage, enhance vulnerability management, and to be able to continuously monitor critical healthcare systems during medical device development[14]. In healthcare settings, however, the adoption of software is technically complex due to the strict regulations that exist in the medical device industry relating to medical device safety, quality management, cybersecurity, and audit traceability[15][16]. International standards like IEC 62304, ISO 13485, ISO 14971 and FDA 21 CFR Part 11 have strict requirements regarding software lifecycle documentation, software validation, risk assessment and compliance verification [17][18].

The software ecosystems in real-world healthcare systems such as remote patient monitoring, intelligent clinical alert systems, wearable healthcare technologies, and telemedicine systems, demand the ability to support continuous patient care, security, scalability, and continuous monitoring. Current solutions often focus on deployment automation and/or healthcare analytics without providing adequate integration of continuous compliance monitoring and operational observability[19][20]. Several problems, such as insufficient visibility of monitoring, late detection of vulnerabilities, disjointed testing environments and the inability to track compliance, only reinforce the need for integrated solutions. The medical device industry is increasingly looking for automated DevOps processes, secure deployment pipelines, real-time monitoring, and regulatory compliance mechanisms to be integrated into a well-defined process to enhance software reliability, operational efficiency, and accountability.

1.1. Motivation and Contributions of the Study

As device software becomes a larger part of the equation, it is more critical than ever to have secure, scalable development environments that enable continuous software delivery without compromising patient safety, and are compliant with regulatory requirements. Traditional medical device development typically faces issues like siloed testing, slow deployment, restricted view of the system, and manual compliance validation. DevOps practices have improved the agility and efficiency of software and there has been limited research that incorporates automated deployment pipelines and continuous regulatory compliance in a healthcare context. The challenges require an intelligent system that integrates automation, monitoring, security and compliance tools, to improve healthcare system software reliability, deployment efficiency and quality. The key contributions are given as follows:

- Developed an automated DevOps-enabled framework for medical device software deployment and healthcare monitoring.
- Integrated continuous integration and continuous deployment pipelines with containerized services to improve scalability and deployment reliability.
- Designed a real-time alert monitoring and compliance audit logging mechanism for continuous patient supervision and traceability.
- Incorporated Prometheus and Grafana for operational observability and real-time system performance monitoring.
- Evaluated framework effectiveness using infrastructure and clinical metrics, including CPU utilization, memory usage, API response time, and alert resolution rate.

1.2. Novelty of the paper

The innovation in this work is that automated DevOps environments will be coupled to the continuous regulatory compliance processes with an emphasis on medical device software development. The framework provides a unified, healthcare-specific environment that includes continuous integration and continuous deployment pipelines, containerized deployment, automated testing, vulnerability scanning, real-time monitoring, and compliance audit logging, all in a single package, as opposed to individual focus areas studied previously. Regulatory mapping to IEC 62304, ISO 13485, ISO 14971, FDA 21 CFR Part 11, HIPAA and IEC 62443 offers secure, traceable, and standards-compliant medical software management throughout the lifecycle.

1.3. Structure of the Paper

The remainder of the paper is organized as follows. Section II presents a detailed literature review. Section III describes the proposed methodology, system architecture, and automated monitoring workflow. Section IV presents the experimental results and performance analysis. Finally, Section V concludes the study and outlines future research directions and possible improvements.

2. Literature Review

The existing literature is predominantly focused on DevOps, automation, testing, and compliance in healthcare environments, and there are comparatively few studies that combine DevOps automated environments with regulatory compliance in a medical device development context.

A. Kaur (2026) explains the role of AI-driven systems in making cloud-based architectures and Continuous Integration/Continuous Deployment (CI/CD) pipelines more robust, accountable, and ethical. In the medical sector, where critical information and decisions are involved, reliable AI guarantees equity, clarity, and information security. Likewise, enterprise systems can leverage it to improve operational efficiency, automation, predictive analytics, risk management, and governance. Model monitoring, bias mitigation, secure data pipelines, and automated deployment strategies are some of the major elements of the research. It also examines the role of DevOps in accelerating AI lifecycle management and ensuring system resilience [21].

M. Zhang (2025) proposes a software improvement scheme with DevOps as the core, including adopting agile development mode, refining test management and optimizing operation and maintenance process, so as to improve the efficiency of software development, test and operation and maintenance, shorten the delivery cycle and reduce compliance risks. The core philosophy of DevOps is continuous integration, continuous delivery and cross-team collaboration, providing a new approach to the development and management of medical device software. The implementation of this strategy to achieve the goal of improving software quality, improving system reliability, and meeting stringent industry standards [22].

A. V. Barone (2025) proposes an Enterprise Real-Time Healthcare Cloud Framework integrating secure APIs, unified payment systems, and DevOps practices to enhance operational efficiency, regulatory compliance, and patient-centred service delivery. The framework leverages cloud-native architectures, microservices, containerization, and zero-trust security models to ensure scalability and data protection. Secure APIs enable smooth integration between Electronic Health Records (EHR), telemedicine systems, laboratory information systems, insurance companies, and third-party services. A single payment gateway links billing, insurance claims, digital wallet and revenue cycle management in a secure financial environment. DevOps integration facilitates continuous integration/continuous deployment (CI/CD), infrastructure automation, monitoring, and speedy innovation while making sure to stick to healthcare regulations like HIPAA and GDPR. The main challenges tackled by the proposed framework are the issues of data silos, lack of integration of payment systems, cybersecurity risks and slow system updates[23].

J. M. I. Arockiasamy (2025) delves into DevOps principles, such as Continuous Integration and Continuous Deployment (CI/CD) pipelines, Machine Learning Operations (MLOps), cloud-native architectures, and security automation (DevSecOps), that support real-time health data streaming and analytics. This paper provides an example of a cardiac monitoring use case that has measured benefits such as sub-second anomaly detection, 95% accuracy, and a factor of five improvement in data processing throughput. Moreover, the scalable framework can be adapted to other medical applications including early detection of stroke by EEG. In the healthcare industry, DevOps best practices can help organizations rapidly leverage AI-driven insights, optimize compliance, and deliver better patient outcomes at scale [24].

M. R. Martina et al., (2024) describe a cost-effective and fully customizable solution for software medical devices problem and modification management implemented in group. Key features of the problem and modification management related to the software life cycle have been identified and taken into account. A digital commercial platform for software development and maintenance has been identified and used to implement an architecture satisfying the standard. The architecture solution (SWMA, software maintenance architecture) has been successfully developed and a description of how it addresses IEC 62304 standard requirements both in terms of problem resolution and in terms of change control has been provided. The presented system can be useful for manufacturers/groups to establish software maintenance plans that include activities and tasks related to software problem resolution and change control processes [25].

D. K. Chikwarti and J. Smith (2023) present a novel approach to software defect mining in medical device software through the deployment of automated test pipelines that integrate continuous testing, logging, and defect analysis. The approach leverages automation tools to systematically execute test cases, capture defect data, and perform mining of defect patterns to identify common fault sources. The automated pipelines have been validated using medical device software in real-world settings, showing improvements in efficiency, reduction of manual workload, and valuable information for quality enhancements. The findings of this study demonstrate the potential of automated test pipelines for software quality assurance in the regulated development process [26].

Table I shows the general overview of the previous studies on DevOps, Artificial Intelligence, Cloud-native systems and compliance in healthcare, approach, conclusion, benefit, drawbacks, and future research in medical device software development.

Table 1. Research Gap in Automated Devops for Medical Device Development and Compliance

Authors	Approaches	Findings	Advantages	Limitations	Recommendations
A. Kaur (2026)	AI-driven systems with cloud-native architecture and CI/CD for trustworthy AI.	Improved robustness, accountability, and ethical AI deployment.	Better explainability, privacy, and automation.	Limited focus on medical device regulations and compliance automation.	Integrate DevOps with regulatory validation for medical devices.
M. Zhang (2025)	DevOps-based agile development, testing, and CI/CD in medical software.	Enhanced quality, reliability, and reduced delivery time.	Faster delivery and reduced compliance risks.	Lacks continuous regulatory monitoring mechanisms.	Include automated compliance tracking in DevOps pipelines.
A. V. Barone (2025)	Cloud-native healthcare framework with secure APIs and DevOps.	Improved interoperability and compliance.	Scalability, automation, and secure integration.	Focused on enterprise healthcare, not medical device lifecycle.	Extend framework for medical device traceability and certification.
J. M. I. Arockiasamy (2025)	Used CI/CD, MLOps, and DevSecOps for healthcare analytics.	Achieved faster analytics and improved outcomes.	Real-time monitoring and scalability.	Limited attention to medical device software standards.	Develop DevOps models aligned with IEC 62304 and ISO 13485.
M. R. Martina et al. (2024)	Developed IEC 62304-compliant software maintenance architecture.	Improved software maintenance and change control.	Cost-effective and standard-compliant.	No full DevOps or CI/CD integration.	Add automated testing and deployment pipelines.
D. K. Chikwarti & J. Smith (2023)	Automated testing and defect mining pipelines.	Improved defect detection and quality assurance.	Reduced manual effort and faster testing.	Limited DevOps lifecycle and compliance traceability.	Combine automated testing with compliance-focused DevOps.

3. Methodology

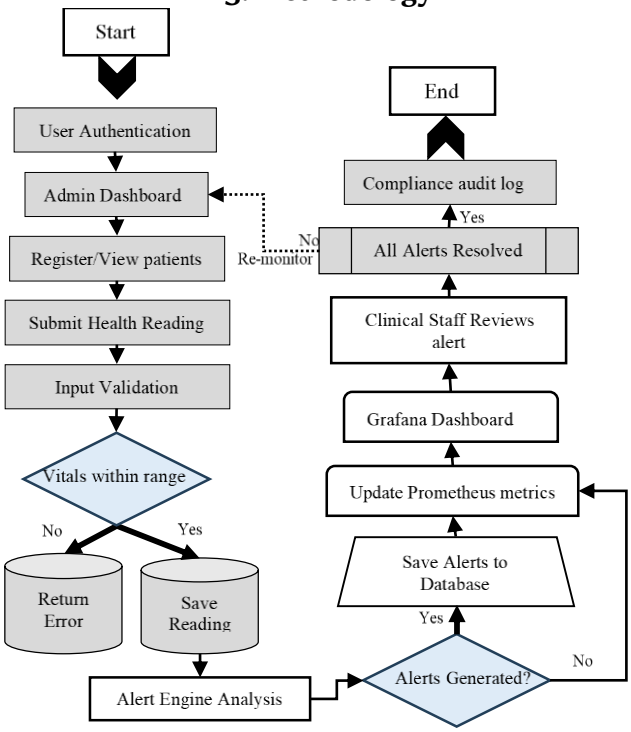

Figure 1. End-To-End Workflow of the Proposed Medical Monitoring and Devops Framework

The summary results of overall clinical monitoring of the proposed system are shown in the Table II. It shows the number of patients being monitored, number of alerts generated, alert statuses, the rate at which alerts are resolved and total health readings. The results indicate that this patient monitoring is effective and the system performs well in the clinical environment when it comes to alerts.

Table 2. Clinical Data Overview of Patient Monitoring and Alert Generation

Metric	Value
Total Patients	8
Total Alerts Generated	220
Active Alerts	61
Acknowledged Alerts	48
Resolved Alerts	111
Alert Resolution Rate	50.5%
Total Health Readings	216

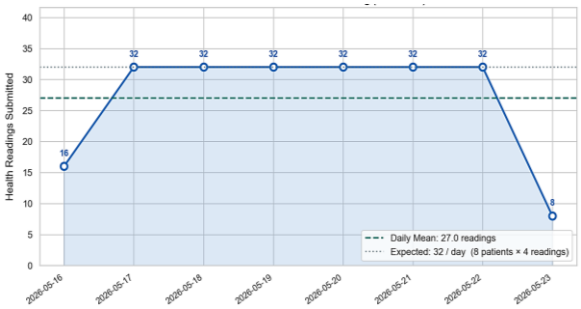

Figure 2. Daily Distribution of Patient Vital Sign Readings Collected During the Monitoring Period

A panel plot of the change in patient vital signs over the course of the day during the monitoring period is shown in Figure 2. This visualization shows the system's ability to automatically gather physiological data like heart rate, oxygen saturation, body temperature, and blood pressure, thus confirming the real-time monitoring of the patient was possible for the system.

3.1. User Authentication and Secure Access

The methodology starts with user authentication to give safe access to the system and ensure privacy of data. The users should log in to the platform with their credentials like administrators, clinical staff etc. The authentication process helps to secure patients' health data from unauthorized access, ensuring adherence to healthcare security standards like access control and data confidentiality. If successful, the users are redirected to the system login screen which is the administrative interface to the system. The dashboard features patient management, health data submission, monitoring and alert tracking capabilities.

3.2. Patient Registration and Monitoring Setup

After authentication, healthcare practitioners will be able to register new patients or view patient information on the dashboard. During this stage patient monitoring environment is set up using patient-specific demographic and clinical information. Doctors can obtain the patient's history from the patient management module, revisit past health records and plan data collection. In clinical practice, with this centralized system, patient tracking and continuity of care will be possible.

3.3. Health Data Submission

The next phase is the submission of health readings where clinical staff or medical devices that are connected to the patient give their clinical vital signs for evaluation. The system is able to record various physiological parameters such as:

- Heart rate (HR)
- Oxygen saturation (SpO_2)
- Body temperature
- Blood pressure

These physiological parameters are conveyed into the monitoring system for additional validation and evaluation. Readings are submitted and retained for future reference and follow-up of patients

3.4. Input Validation Mechanism

The health readings collected would then be checked for data validity and reliability such that they are subjected to an input validation. This step enables to identify missing values, impossible physiologically measurement formats and rejects them. The validation process helps to assure consistency in information received and minimize incorrect predictions and false alarms. This is a vital step to ensure that what's generated from alerts and clinical decisions is of high quality.

3.5. Vital Threshold Assessment

After the validation, the system will decide whether the patient's vital signs entered into the system are within the clinically acceptable ranges. A decision-based threshold assessment mechanism is a mechanism in which the measurements received are compared to medical reference values that are predefined. If the patient's vital signs are within normal physiologic range, the reading is acceptable and automatically entered into the database to be monitored and analyzed in the future. If abnormal, the system recognizes it as a potential high-risk situation and initiates an alert generation workflow to further investigate the situation.

3.6. Error Handling and Data Correction

If the data that is sent is not valid or it exceeds the acceptable limits, it activates a mechanism that returns an error. This step alerts the clinical user to an inconsistency has been identified and requests him/her to update the data or enter the data again. The error handling procedure helps to minimize incorrect health records, and prevents incorrect measurements from impacting patient risk assessment.

3.7. Alert Engine Analysis

If the patient's vital information is not normal, then the abnormal patient's vital information is passed to the alert engine analysis module. This component checks for exceeding abnormal physiological thresholds and decides if an alert should be created, depending on medical thresholds and risk conditions. The abnormality significance is calculated by the alert engine and it stores them in the category

based on the risk level. This automated assessment allows for quick detection of patient deterioration and helps facilitate prompt intervention by health care providers.

3.8. Alert Generation and Database Logging

Automatically generates alerts in the event of critical or abnormal physiological conditions and stores them in a monitoring database. The alert repository stores important metadata such as:

- Patient identifier
- Timestamp of occurrence
- Abnormal parameter detected
- Severity level
- Alert status

If no clinically significant anomaly is identified, the monitoring process continues without triggering further intervention.

3.9. Prometheus Metrics Update and Real-Time Monitoring

Alerts are produced and system performance metrics are sent to the Prometheus monitoring system that continually updates relevant operational metrics. These are the frequency of alerts, changes in patient status and the performance of the monitoring system. In healthcare systems that demand a high level of reliability, Prometheus offers continuous observability and effective system-level monitoring.

3.10. Grafana-Based Visualization Dashboard

Monitoring metrics are now displayed on the Grafana dashboard, giving the user a simple-to-use GUI to display alerts in real time and monitor patients. The dashboard allows for easy trend analysis, alert tracking and visualization of patient health indicators. There is a quick deterioration in the clinical condition (information is presented visually) and the clinician can indicate intervention.

3.11. Clinical Staff Alert Review

Once viewed, created alerts are then forwarded to clinical personnel for professional evaluation. Health care personnel review the alerts to determine the clinical significance and whether to take action or continue to monitor the alert. This verification process by the human in the loop reduces false positives and ensures clinical validation of automated decision for further action.

3.12. Alert Resolution and Re-Monitoring Process

After the clinical assessment, the system checks for a resolution to all alerts generated.

- If the alerts are resolved successfully then the framework notifies the event in the event in audit and regulatory log, which provides traceability and accountability for future audit and regulatory checks.
- If conditions do not improve, the system will restart a re-monitoring process which will re-enter the patient in the monitoring process, and will continue monitoring and re-evaluating the patient's health.
- It is a process of continuous observation of patient's condition and repeated until the condition is stabilized.

3.13. Compliance Audit Logging

The last step of the methodology is to maintain a compliance audit log. Every action/activity of the system (such as alerts generated, clinical review, resolution, and timing) is being captured for accountability and transparency. The audit trail plays a key role in healthcare compliance, ensuring data integrity, traceability, and compliance. This is especially crucial in medical settings where paperwork and record-keeping are vital.

3.14. Performance Metrics

To measure the effectiveness of the proposed DevOps-based healthcare monitoring framework, various measurable system performance metrics are taken such as CPU usage, memory usage, API response time, alert resolution time, system throughput etc. The metrics selected were focused on the assessment of computational efficiency, responsiveness, reliability and operational performance.

3.14.1. CPU Utilization

The CPU utilization is a percentage of CPU resources used at run-time of the system [27]. Minimize CPU usage which is efficient computation and minimize infrastructure overheads as in Equation (1).

$$CPU\ Utilization(\%) = \frac{CPU\ Time\ Used}{Total\ CPU\ Capacity} \times 100 \quad (1)$$

3.14.2. Memory Consumption

The size of memory refers to the amount of RAM consumed by containers and services deployed. Memory-efficient use helps to ensure the scalability and stability of the monitoring services. The formula to determine memory consumption is given in Equation (2):

$$Memory\ Usage(\%) = \frac{Consumed\ Memory}{Total\ Available\ Memory} \times 100 \quad (2)$$

3.14.3. API Response Time

API response time refers to the responsiveness of the healthcare monitoring system, which is defined as the time from when the user sends a request to the time the server responds (Equation 3).

$$Response\ Time = T_{response} - T_{request} \quad (3)$$

3.14.4. Alert Resolution Rate

The alert resolution rate as defined in Equation (4) indicates the proportion of the alerts that are resolved by the clinical staff (operation effectiveness) after the alerts are generated.

$$Alert\ Resolution\ Rate(\%) = \frac{Resolved\ Alerts}{Total\ Alerts} \times 100 \quad (4)$$

The higher the resolution rate, the more clinical response rate, efficient alert management. The performance metrics offer a complete assessment of the proposed system with respect to its computational efficiency, responsiveness, scalability and clinical operational efficiency.

4. Result Analysis and Discussion

The DevOps containerized healthcare monitoring system is built using the following tools: Node.js, Express.js, React.js, MongoDB, Prometheus, Grafana, Docker, Docker Compose, and GitHub Actions for automated deployment, healthcare monitoring and CI/CD pipelines. The system ran on an Intel Core i5 and i7 processor powered by 8-16GB of RAM and SSD storage, allowing patient vital analysis to be performed in real-time, alerts to be automatically triggered as necessary, and the system to be monitored continuously. Table III gives the distribution of alerts generated by abnormal patient vital signs. The most common abnormal condition identified was high blood pressure, with a high heart rate and several abnormal vital signs following. The outcomes illustrate the framework's potential to recognize a variety of patient risks.

Table 3. Distribution of Alert Types Generated from Abnormal Vital Sign Detection

Alert Type	Count	Percentage
High Blood Pressure	61	29.3%
High Heart Rate	38	18.3%
Multiple Vitals Abnormal	37	17.8%
Low Oxygen	34	16.3%
High Temperature	26	12.5%
Critical Low Oxygen	12	5.8%

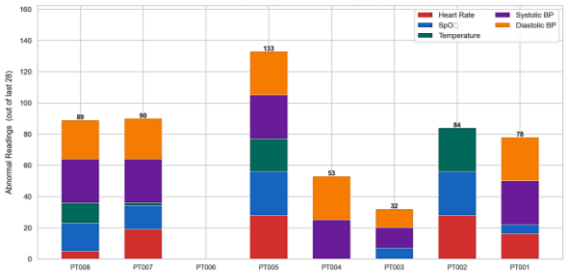

Figure 3. Detection and Classification of Abnormal Patient Vital Signs Based On Clinical Thresholds

The frequency and type of abnormal patient vital signs identified by the proposed framework are illustrated in Figure 3. The system detects deviations from the defined critical values and assigns them a level of severity, so that medical staff can prioritize their medical interventions and efficiently manage patient risks.

The CPU utilization of the containerized services deployed in the proposed medical monitoring framework is displayed in Figure 4. The results indicate that the system's computational resources are being utilized efficiently by all system components, thus its architecture is lightweight and is able to support scalable and reliable healthcare monitoring operations.

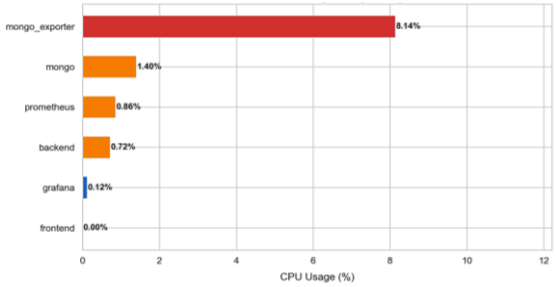

Figure 4. CPU Utilization of Containerized Services in the Proposed Medical Monitoring Framework

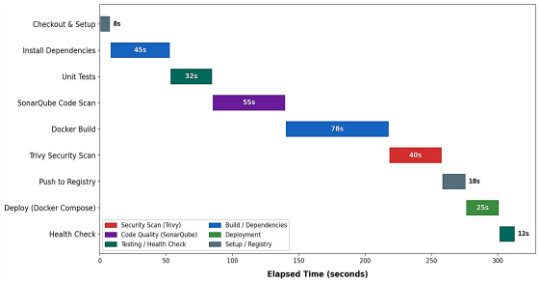

Figure 5. Automated CI/CD Pipeline for Secure Medical Device Software Deployment

This is Figure 5 in which medical device software development has been automated with Continuous Integration and Continuous Deployment (CI/CD) pipeline. The automated testing, vulnerability scanning, code quality evaluation, and deployment phases of the workflow guarantee secure, reliable, and regulatory-compliant software delivery in the healthcare sector.

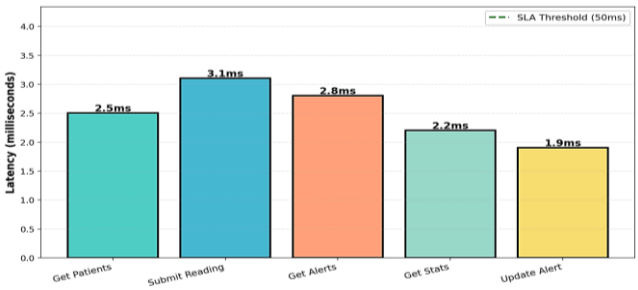

Figure 6. API Response Time Performance of the Proposed Remote Monitoring System

The response time of the application programming interfaces (APIs) of the proposed remote monitoring system is shown in Figure 6. The findings indicate that the system is both responsive and low-latency, showcasing its potential for real-time healthcare monitoring and timely clinical decisions.

Table IV gives the infrastructure efficiency and DevOps performance of the proposed system. The framework has shown that it uses a lightweight number of resources with just 1.99% CPU usage and 376.2 MB memory consumption, with API response time of 1.9-3.1 ms. The deployment consisted of 8 services deployed inside containers, automated CI/CD pipeline with a maximum of 20 minutes execution time, and continuous monitoring with Prometheus and Grafana.

Table 4. Infrastructure and Devops Performance Metrics

Metric	Value
Total CPU Utilization	1.99%
Total Memory Consumption	376.2 MB
Average API Response Time	1.9–3.1 ms
Number of Containers Deployed	8
CI/CD Deployment Time	~20 Minutes

Table V summarizes implementation of the proposed framework with internationally recognized medical and cybersecurity standards. It combines medical device lifecycle management, risk assessment, electronic record security, quality assurance, privacy protection, and cybersecurity features to facilitate the deployment of reliable and compliant healthcare software.

Table 5. Regulatory Compliance Mapping of the Proposed Medical Monitoring Framework

Standard	Focus Area	Implementation
IEC 62304	Medical Device Lifecycle	Containerised deployment with version control
ISO 14971	Risk Management	Alert severity classification (high/critical/medium)
FDA 21 CFR Part 11	Electronic Records	JWT authentication, audit logging, timestamps
ISO 13485	Quality Management	Automated testing, code quality gates (SonarQube)
HIPAA Technical	Data Privacy	Role-based access control, encrypted communications
IEC 62443	Cybersecurity	Vulnerability scanning (Trivy), secure CI/CD pipeline

4. Discussion

The results of the experiments suggest that connection between automated DevOps environments and medical device monitoring systems leads to better deployment efficiency, monitoring and observability of the system, software reliability, and maintains the regulatory compliance. Performance metrics of the framework, such as CPU usage of 1.99%, memory usage of 376.2 MB, and API response time ranging from 1.9 to 3.1 ms, indicate that the framework has great potential in achieving low computational overhead, which can effectively support real-time healthcare operations without requiring large infrastructure demands. It can also generate alerts at a high rate (71.6%), as well as resolve the same in time (50.5%) in the correct time range. Traceability, cyber security and lifecycle management are enhanced by the combination of Prometheus and Grafana, automated testing and vulnerability scanning. However, there are some

limitations to broader clinical validation, interoperability with diverse health systems and predefined thresholds. Predictive analytics and adaptive compliance monitoring systems could contribute to improving the scalability, intelligence, and sustainable compliance of medical device software systems in the future.

5. Conclusion and Future Scope

Automated DevOps environments can be incredibly useful for medical device software development to facilitate deployment, enhance software reliability, ensure adherence to regulations, and ensure observability of the healthcare system. The achieved integration of Continuous Integration, Continuous Deployment pipelines, Containerized services, automated testing, Vulnerability scanning, Real-time monitoring and Compliance auditing in the implemented framework allowed for a safe and efficient medical monitoring process. Eighteen containers were used in this way and continuous monitoring was demonstrated with experimental results showing that the infrastructure resource consumption was low (CPU: 1.99% memory: 376.2 MB) and the API response time was low (1.9-3.1ms). This validated the ability of the framework to support timely clinical intervention, operational transparency, as 220 alerts were generated, 111 of which were resolved (50.5%). Regulatory alignment via IEC 62304, ISO 13485, ISO 14971, FDA 21 CFR Part 11, HIPAA and IEC 62443 enhanced software traceability, software quality assurance and software cyber security readiness. Despite these benefits, there are issues to consider, including the need for large-scale clinical validation, interoperability with other health-care systems, and reliance on threshold-based alert systems. Further research and development may include deployment environments in multiple hospitals to improve deployment scalability, efficient automation, and clinical decision making, as well as AI-based predictive monitoring, adaptive risk assessment, intelligent anomaly detection, and others.

References

[1] M. R. Anand and A. K. S, "Transforming Energy-Intensive Smart Factories with AI: TCN-based Forecasting and DQN-Driven Operational Optimization for Healthcare Manufacturing," in *Proceedings of the 2025 International Conference on Intelligent Computing, Information and Control Systems*, 2025, pp. 508–515. doi: 10.1109/ICOIICS67115.2025.11390244.

[2] C. Tayal, S. Murumkar, and B. Mohan, "Semantic Web Healthcare Chatbot Using Ontology and Data Mining for Patient Information Assistance," in *2026 18th International Conference on Knowledge and Smart Technology (KST)*, IEEE, Jan. 2026, pp. 465–471. doi: 10.1109/KST67832.2026.11432402.

[3] J. Mishra, S. Rai, B. Moharana, V. K. Singh, S. Dey, and T. Sarkar, "Revolutionizing Financial Fraud Detection in Healthcare Insurance Through Blockchain & AI-Driven Predictive Analysis," in *2025 International Conference on Technology Enabled Economic Changes (InTech)*, IEEE, Feb. 2025, pp. 1516–1522. doi: 10.1109/InTech64186.2025.11198190.

[4] S. Kilaru, "Automated ETL Intelligence: Metadata-Orchestrated Framework with Rule-Based Heuristics for Monitoring and Reporting," *Int. J. Inf. Electron. Eng.*, vol. 3, no. 6, p. 14, 2013.

[5] A. V. S. R. Dantuluri and S. Kumar, "A Governance-Driven, Real-World Data-Calibrated Health Informatics Framework for Longitudinal Utilization Forecasting in Oncology and Complex Chronic Conditions," Feb. 26, 2026, *Cold Spring Harbour Laboratory Press*. doi: 10.64898/2026.02.23.26346919.

[6] M. Indirani, S. Sudheer, R. Mahaveerakannan, and others, "Gallstone Disease Prediction Using Clinical and Biochemical Features Through Ensemble Learning Techniques," *Int. J. Comput. Intell. Syst.*, vol. 19, no. 19, p. 19, 2026, doi: 10.1007/s44196-025-01083-0.

[7] J. A. Kachhia, "Healthcare Predictive Analytics based on Machine Techniques for Identifying Cardiovascular Risks Screening," *Int. J. Curr. Eng. Technol.*, vol. 13, no. 6, pp. 635–642, 2023, doi: 10.14741/ijcet/v.13.6.17.

[8] S. Ganesan and G. Arulkumaran, "AI-driven software testing and development: Enhancing automation, efficiency, and reliability in agile and DevOps environments," *Int. J. Multidiscip. Curr. Res.*, vol. 9, no. 2, 2021.

[9] A. A. Soni, M. Parikh, R. N. K. Dhenia, J. A. Soni, A. R. Jha, and S. M. Shah, "Reinforcement Learning for Dynamic Workflow Optimization in CI/CD Pipelines," in *2025 IEEE 17th International Conference on Computational Intelligence and Communication Networks (CICN)*, IEEE, Dec. 2025, pp. 638–644. doi: 10.1109/CICN67655.2025.11367872.

[10] T. Laukkarinen, K. Kuusinen, and T. Mikkonen, "DevOps in Regulated Software Development: Case Medical Devices," in *2017 IEEE/ACM 39th International Conference on Software Engineering: New Ideas and Emerging Technologies Results Track (ICSE-NIER)*, IEEE, May 2017, pp. 15–18. doi: 10.1109/ICSE-NIER.2017.20.

[11] P. R. Mudunuri, "Automating Compliance in Biomedical DevOps: A Policy-as-Code Approach," *Int. J. Res. Appl. Innov.*, vol. 5, no. 2, pp. 6770–6783, 2022.

[12] P. Kumar, "Edge Computing and IoT for Real-Time Healthcare Data Processing and Integration," in *2025 4th International Conference on Applied Artificial Intelligence and Computing (ICAAIC)*, IEEE, Dec. 2025, pp. 105–110. doi: 10.1109/ICAAIC64647.2025.11331211.

[13] S. Dharmavaram and P. Bhanushali, "Machine Intelligence-Driven Forecasting for ED Triage and Dynamic Hospital Patient Routing," Feb. 20, 2026, *Cold Spring Harbour Laboratory Press*. doi: 10.64898/2026.02.18.26346566.

[14] R. Kant, R. Rao Thallada, B. Pandey, and P. Srivastava, "AI-Based Cybersecurity in Healthcare: A Data-Driven, Governance-Aware Framework for Secure Clinical Systems," in *2026 IEEE 5th International Conference on AI in Cybersecurity (ICAIC)*, 2026, pp. 1–5. doi:

10.1109/ICAIC67076.2026.11395836.
[15] V. V. R. Boda and H. Allam, "The AI Revolution in Healthcare DevOps: What You Need to Know," *Int. J. Artif. Intell. Data Sci. Mach. Learn.*, vol. 5, no. 4, pp. 39–47, 2024, doi: 10.63282/3050-9262.IJAIDSML-V5I4P105.
[16] Z. Babar, "A study of business process automation with DevOps: A data-driven approach to agile technical support," *Am. J. Adv. Technol. Eng. Solut.*, vol. 04, no. 04, pp. 01–32, Dec. 2024, doi: 10.63125/3w5cjn27.
[17] A. Parupalli and S. Pandya, "Compliance-Driven Data Governance : A Survey on GDPR , and HIPAA in Cloud Databases," vol. 12, no. 6, pp. 828–836, 2022, doi: 10.14741/ijcet/v.12.6.18.
[18] R. S. Snehamruth, "Data-Driven Optimization of Pharmaceutical Manufacturing Processes using Quality by Design (QbD) Frameworks," *Int. J. Curr. Eng. Technol.*, vol. 14, no. 6, pp. 557–566, 2024, doi: 10.14741/ijcet/v.14.6.19.
[19] A. Nerella and J. W. Sajja, "Responsible AI in Enterprise Applications: Balancing Innovation and Compliance," *Comput. Fraud Secur.*, vol. 2023, no. 7, Jul. 2023, doi: 10.52710/cfs. 744.
[20] S. Tatavarthi, S. Tarakampet, and R. R. Koilakonda, "Leveraging Generative AI for Adaptive Compliance Workflows in Enterprise Platforms," *Int. J. Res. Appl. Sci. \& Eng. Technol.*, vol. 14, no. 1, 2026, doi: 10.22214/ijraset. 2026.77213.
[21] A. Kaur, "Advancing Autonomous Intelligence through Trustworthy AI Cloud and DevOps for Enterprise and Healthcare Systems," *Int. J. Multidiscip. Res. Sci. Eng. Technol. Manag.*, vol. 2, no. 3, 2026, doi: 10.15680/IJMRSETM.2026.0203002.
[22] M. Zhang, "Optimization of Medical Device Software Lifecycle Management Based on DevOps," *J. Med. Life Sci.*, vol. 1, no. 3, pp. 8–13, 2025, doi: 10.71222/etnrdv93.
[23] A. V. Barone, "Enterprise Real Time Healthcare Cloud Framework with Secure APIs Unified Payments and DevOps Integration," *Int. J. Eng. \& Ext. Technol. Res.*, vol. 7, no. 6, pp. 11122–11130, 2025, doi: 10.15662/IJEETR.2025.0706032.
[24] J. M. I. Arockiasamy, "DevOps-Driven Real-Time Health Analytics: A Scalable Framework for Wearable IoT Data," *Int. J. Multidiscip. Res.*, vol. 7, no. 1, pp. 1–10, Feb. 2025, doi: 10.36948/ijfmr.2025.v07i01.37358.
[25] M. R. Martina, E. Bianchini, S. Sinceri, M. Francesconi, and V. Gemignani, "Software medical device maintenance: DevOps-based approach for problem and modification management," *J. Softw. Evol. Process*, vol. 36, no. 4, p. e2570, Apr. 2024, doi: 10.1002/smr. 2570.
[26] D. K. Chikwarti and J. Smith, "Software Defect Mining in Medical Devices Using Automated Test Pipelines," vol. 14, no. 1, pp. 1320–1331, 2023.
[27] I. Karamitsos, S. Thabit, and C. Apostolopoulos, "Applying DevOps Practices of Continuous Automation for Machine Learning," *Information*, vol. 11, no. 7, p. 363, Jul. 2020, doi: 10.3390/info11070363.