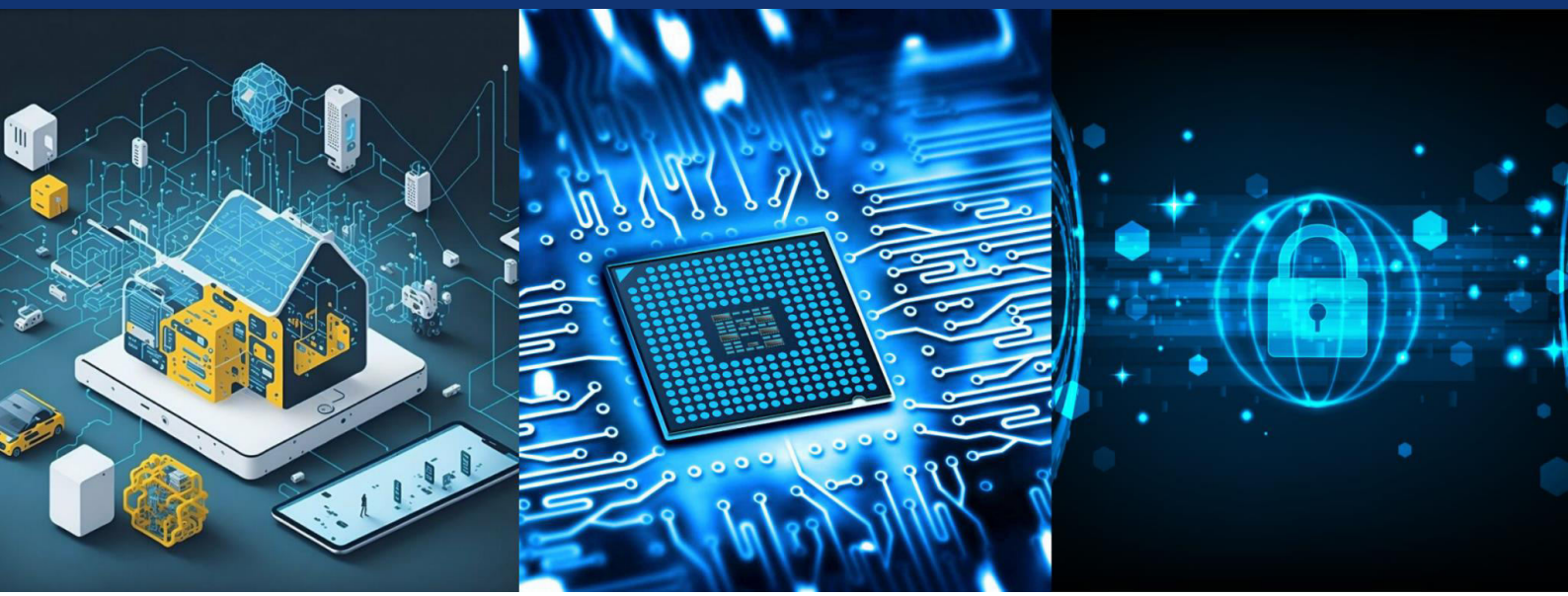
IJIRCCE

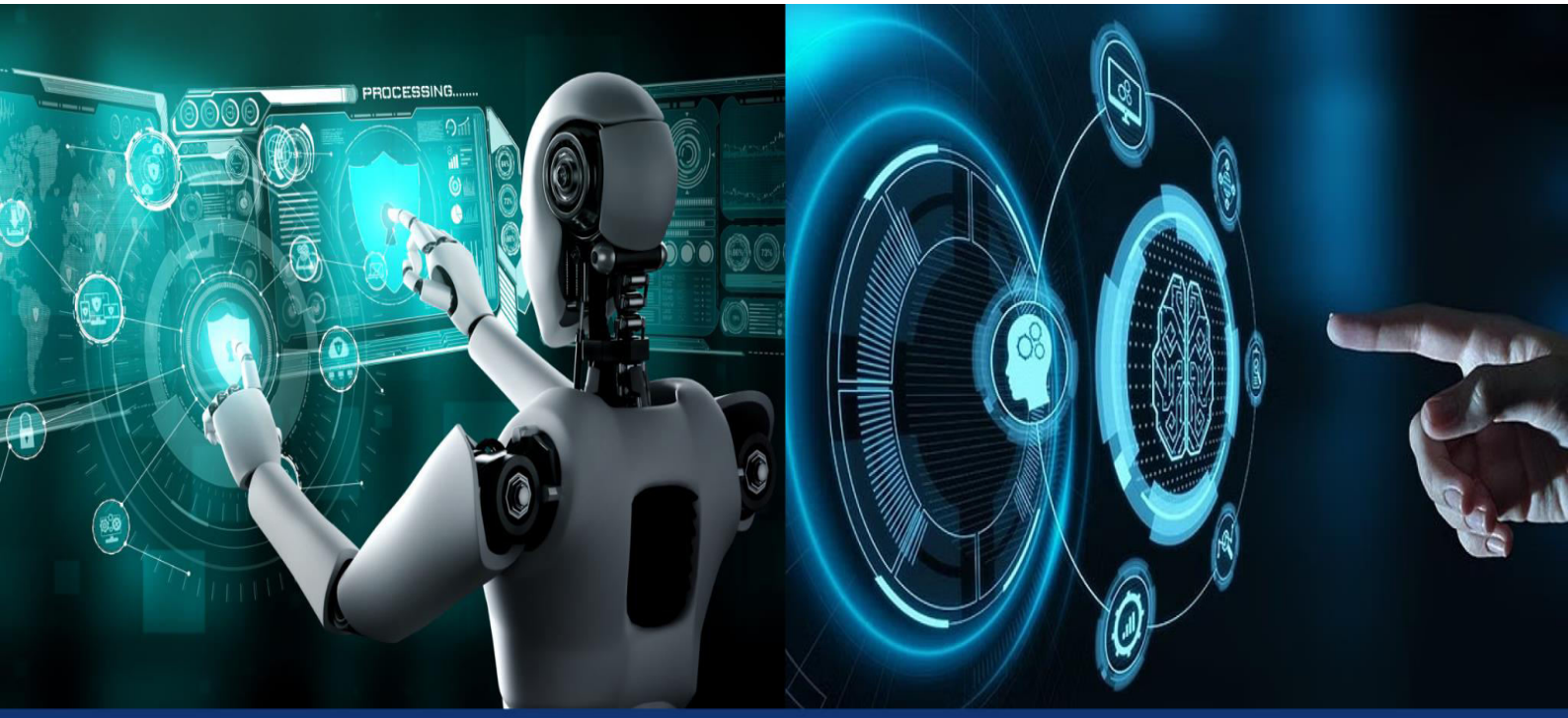
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 7, July 2026

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407050

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Smart DevOps Monitoring System

Dr. Manu M N, Prof. Prarthana J V, Prof. Gayathri S

Dept. of ISE, SJB Institute of Technology, Bengaluru, India

ABSTRACT: In today's digitally connected world, the reliability and uptime of IT infrastructure play a crucial role in the success of businesses. Any unplanned system downtime can lead to revenue loss, reduced productivity, and damaged customer trust. While several tools exist to monitor system health, most solutions are reactive—they alert only after an issue has already occurred. To bridge this gap, this project proposes a Smart DevOps Monitoring & Prediction System, which combines the strengths of open-source monitoring tools and AI-driven analytics to enable both real-time monitoring and predictive maintenance. In the current phase (Phase-1), the project establishes a robust foundation for real-time system monitoring using the Prometheus monitoring system and Windows Exporter, which collects machine-level metrics such as CPU usage, memory availability, and disk space from a Windows environment. These metrics are visualized using Grafana, which offers interactive dashboards that allow developers and system administrators to monitor system health at a glance. The system is capable of scraping and storing time-series metrics from servers, making them available for querying and visualization. The use of Prometheus's pull-based data collection and Grafana's customizable dashboards ensures flexibility and scalability. This forms the basis for the next development phase where AI models—such as anomaly detection and failure prediction using Isolation Forest or LSTM—can be integrated to forecast failures and automatically trigger alerts or recovery mechanisms. By combining DevOps tools with AI, the Smart DevOps Monitoring & Prediction System aims to significantly reduce Mean Time to Detection (MTTD) and Mean Time to Recovery (MTTR), while enabling proactive system maintenance. This project has applications in cloud management, enterprise IT infrastructure, and datacentre operations—where even minutes of downtime can result in high operational costs. In the long run, this system can be deployed across organizations as a cost-effective and intelligent DevOps

I. INTRODUCTION

The increasing complexity of IT systems has made efficient infrastructure monitoring essential, especially in cloud-based and microservices-driven environments. Traditional monitoring tools are often reactive and usually detect issues only after they affect system performance or availability.

The Smart DevOps Monitoring & Prediction System addresses this gap by combining Prometheus, Windows Exporter, and Grafana to support real-time monitoring, metric collection, and clear dashboard-based visualization of system health. The system gathers important machine-level data such as CPU usage, memory consumption, disk activity, and network statistics, enabling administrators to track performance continuously from a centralized platform.

In future phases, machine learning models such as anomaly detection and failure prediction can be integrated to identify unusual behaviour and forecast possible failures in advance. This transforms the monitoring process from reactive troubleshooting into proactive maintenance, helping reduce downtime, improve reliability, and support faster incident response.

The architecture is modular, scalable, and suitable for environments ranging from personal systems and academic labs to enterprise infrastructure. It also creates a strong foundation for self-healing mechanisms, intelligent alerting, and AIOps-oriented enhancements that can improve operational efficiency over time.

II. LITERATURE SURVEY

The objective of the literature survey for this project is to study existing system monitoring and remote management solutions to understand their architectures, technologies, and limitations. It focuses on agent-server communication, heartbeat mechanisms, logging, and secure data transmission. The survey also helps compare centralized and distributed monitoring approaches and identify best practices for background services, installers, performance, and security.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

1. Agent-Based Network Monitoring for Real-Time System Management (2021)
This study shows that lightweight agents on individual machines can improve scalability, reduce latency, and support real-time fault detection. It also highlights the need for secure communication and efficient resource use.
2. Centralized Monitoring of Distributed Systems Using Heartbeat Mechanisms (2020)
This paper explains how periodic heartbeat messages help detect node failures quickly. It is a simple and reliable method for real-time supervision, though timeout tuning is important to avoid false alarms.
3. Automated System Supervision Using Lightweight Monitoring Agents (2022)
This research shows that background agents can automate availability monitoring and simple remote actions. It improves uptime, reduces manual effort, and supports deployment across multiple machines.
4. Secure Remote Command Execution in Agent-Based Monitoring Systems (2023)
This study focuses on safe remote command execution using whitelisting, authentication, and encryption. It improves administrative efficiency while reducing misuse risks.
5. Monitoring Windows Systems Using Background Services and Logging Mechanisms (2021)
This paper emphasizes running agents as Windows services for uninterrupted monitoring. It also shows that logging is essential for diagnostics, reliability, and long-term stability.
6. Case Study: Monitoring and Availability Tracking in a Computer Laboratory
This case study shows that heartbeat-based monitoring can identify online and offline systems automatically. It reduces manual checking and improves lab management.
7. Case Study: Centralized System Monitoring for Remote Administration
This case study demonstrates that centralized monitoring improves visibility and speeds up issue response. It reduces the need for physical access to machines.
8. Case Study: Automated Background Monitoring Using Windows Services
This study confirms that service-based agents continue monitoring even after reboot or logout. It improves reliability and system uptime.
9. Case Study: Controlled Remote Command Execution for System Management
This case study shows that only approved commands should be allowed for safety. It supports remote control without compromising security.
10. Case Study: Real-Time Status Visualization Using Centralized Dashboards
This case study highlights the value of dashboards for showing live machine status. It helps administrators identify inactive systems quickly and make better decisions.
Existing System: The existing system is mostly manual and depends on separate tools like Task Manager, Event Viewer, and command-line utilities. It does not provide centralized monitoring, real-time alerts, or automated actions. As a result, failures may be detected late, and scalability becomes difficult as the number of machines increases.
Proposed System: The proposed system aims to build a centralized, automated monitoring solution for multiple laptops or machines. It supports device registration, periodic heartbeats, remote command execution, alerting, and logging. The system is designed to be secure, scalable, lightweight, and easy to deploy in academic, office, and small organizational environments.

III. SYSTEM DESIGN

The flow diagram illustrates the overall architecture and operational workflow of the Smart DevOps Monitoring System. The system is designed around a centralized monitoring approach where an administrator or user interacts with the platform through a web browser or mobile interface. All user requests, such as viewing system status or issuing management commands, are transmitted to the backend using RESTful APIs, ensuring standardized and secure communication between the user interface and the server components.

At the core of the system is the Central Backend API developed using the Flask framework. This backend acts as the main control and coordination layer of the monitoring system. It handles the registration of monitored machines when a Windows Agent is installed, manages heartbeat signals received from each client system, processes remote command requests, and continuously calculates the operational status of all registered machines. The backend maintains machine-related data in structured storage formats such as JSON files, which store essential information including machine identifiers, connectivity status, and last heartbeat timestamps.

Once a machine is registered, the Windows Agent runs as a background service on the client system. The agent periodically sends heartbeat signals and system-level information to the backend server, indicating that the machine is

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407050

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

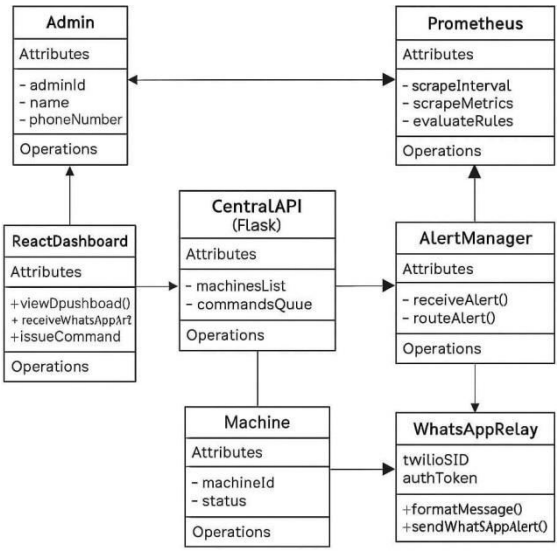
active and functioning correctly. In addition to heartbeat communication, the agent exposes system performance metrics such as CPU usage, memory utilization, and disk statistics. These metrics are collected by Prometheus, which continuously scrapes the data at predefined intervals for monitoring and analysis.

Alerting is an integral part of the monitoring workflow. When Prometheus detects abnormal conditions such as missed heartbeat signals, threshold violations, or service unavailability, it triggers alert events. These alerts are routed to a WhatsApp Relay API implemented using Flask and Twilio. The relay service sends real-time notifications to administrators via WhatsApp, ensuring immediate awareness of critical system issues and enabling prompt corrective action. Overall, the flow diagram represents a tightly integrated monitoring ecosystem that combines distributed agents, centralized data processing, real-time metrics analysis, and instant alerting. By leveraging lightweight agents, a Flask-based backend, Prometheus monitoring, and WhatsApp-based notifications, the Smart DevOps Monitoring System provides a scalable, efficient, and reliable solution for real-time system supervision and remote management.

Class Diagram Components

Admin/User (Web Browser/Mobile): Provides an interface for administrators and users to view system status, monitor machines, and manage alerts using REST APIs.

- Central Backend API (Flask): Acts as the core controller of the system, handling machine registration, heartbeat management, command processing, and overall system status calculation.
- Machines JSON (Machine Data): Stores configuration and metadata of registered machines, which is used by Prometheus for metric labeling and monitoring.
- Prometheus (Metrics Collection): Collects and stores monitoring metrics from agents and exporters, enabling real-time performance analysis and alert evaluation.
- Blackbox Exporter (Website Monitoring): Monitors external services and websites by probing endpoints and forwarding availability metrics to Prometheus.
- Windows Agent (Service): Runs as a background service on Windows machines to register the system, send periodic heartbeats, fetch commands, and execute OS-level operations.
- Windows Machine: The physical or virtual system being monitored, where resource usage and system health data are collected.
- WhatsApp Relay API (Flask + Twilio): Handles alert notifications by forwarding critical system alerts from Prometheus to users via WhatsApp messages.



Class Diagram of MonitoringAgent

Figure 1: Class Diagram

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407050

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The class diagram represents the static structure of the Monitoring Agent System by illustrating the main classes, their attributes, methods, and relationships.

Admin monitors machine status, receives alerts, and issues remote commands.

- CentralBackendAPI acts as the core controller, processing heartbeats, managing machine data and commands, and enabling communication between the admin and agents.
- WindowsAgent collects system metrics, sends heartbeats, retrieves commands, and executes them on monitored machines.
- SystemMetrics stores real-time performance data such as CPU, memory, disk usage, and uptime.
- Prometheus collects and stores system metrics for monitoring and threshold evaluation.
- Alert Manager processes alerts generated by Prometheus and forwards critical notifications.
- WhatsAppService sends real-time alert notifications to the administrator.
- Machine Data and Command Data store machine status information and command details, respectively.

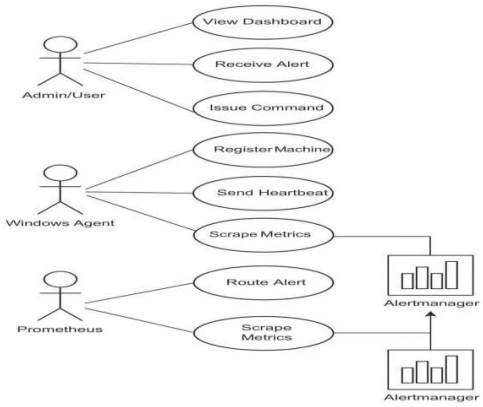


Figure 2: Usecase Diagram

The use case diagram represents the functional behavior of the Monitoring Agent System by showing how users interact with the system.

- Admin monitors machine status, receives alerts, and issues remote commands through the dashboard.
- Windows Agent sends system metrics and periodic heartbeat signals to the backend.
- Monitor System Metrics collects CPU, memory, and disk usage data for performance monitoring.
- Send Heartbeat ensures machine availability and helps detect failures.
- Generate Alerts is triggered when system metrics exceed predefined thresholds.
- Receive Alerts allows the admin to receive real-time WhatsApp notifications.
- Issue Remote Commands enables the admin to send commands to monitored machines.
- Execute Command allows the agent to execute commands and report the results.

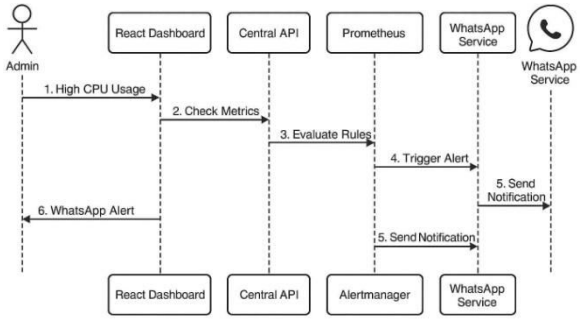


Figure 3: Sequence Diagram

© 2026 IJIRCCE | Volume 14, Issue 7, July 2026 DOI: 10.15680/IJIRCCE.2026.1407050

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

The Data Flow Diagram (DFD) illustrates how monitoring data and remote commands move through the Monitoring Agent System.

- The Windows Machine sends system metrics to the Monitoring Agent System for storage and analysis.
- The system evaluates the metrics and generates alerts when thresholds are exceeded.
- Alerts are sent to the Admin through the WhatsApp Service.
- The Admin issues remote commands, which are stored, executed by the Windows Machine, and their execution status is returned to the system.

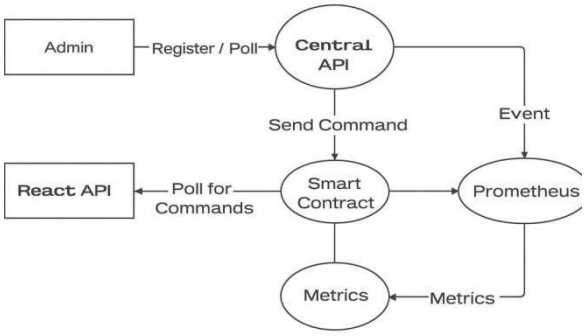


Figure 4: Dataflow Diagram

RESULTS AND DISCUSSIONS

Figure 5: Prometheus.yml

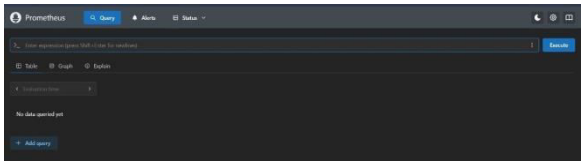


Figure 6: Prometheus Page

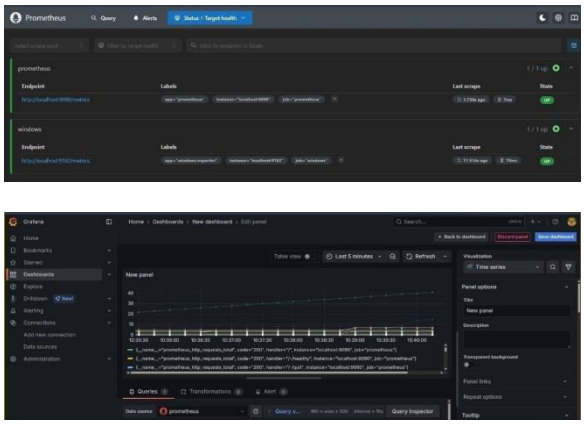


Figure 7: Dashboard

The Smart DevOps Monitoring System was successfully designed, implemented, and tested as a reliable solution for real-time infrastructure monitoring. By integrating the Windows Monitoring Agent, Prometheus, Central Backend API, Alertmanager, and WhatsApp notification service, the system provides continuous monitoring, proactive alerting, and remote command execution.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

Comprehensive unit, integration, and system testing confirmed the correct functioning of all components and validated the system's performance in real-world scenarios. The solution effectively meets its objectives by improving system visibility, enabling timely notifications, and supporting efficient remote administration.

Future enhancements include AI-based predictive analytics, cross-platform support (Linux and macOS), a mobile application for alert management, and enhanced security features to further improve scalability, usability, and enterprise readiness

REFERENCES

[1] Foundational DevOps/CD Humble, J., & Farley, D. (2010). Continuous delivery: Reliable software releases through build, test, and deployment automation. Addison-Wesley Professional.
[2] DevOps Methodology Forsgren, N., Humble, J., & Kim, G. (2018). Accelerate: The science of lean software and DevOps: Building and scaling high performing technology organizations. IT Revolution Press.
[3] DevOps/CI/CD Research Fitzgerald, B.,&Stol, K. J.(2017).Continuous software engineering: A roadmap and agenda. Journal of Systems and Software, 123, 176–189.
[4] DevOps Culture (SLR) Sánchez-Gordón, M., & Colomo-Palacios, R. (2018). Characterizing DevOps culture: A systematic literature review. In Communications in Computer and Information Science (Vol. 888, pp. 3–15). Springer.
[5] Continuous Monitoring in CI/CD Karamitsos, I., Albarhami, S., & Apostolopoulos, C. (2020). Applying DevOps practices of continuous automation for machine learning. Information, 11(7),363.
[6] DevOps in Cloud/IoT Horvath,K., Venter,L., &Schimka,H. (2024).Cloud-based infrastructure and DevOps for energy fault detection in smart buildings. Computers, 13(1), 23.
[7] Monitoring Stack (Prometheus/Grafana) Elradi, M. D. (2025). Prometheus & Grafana: A metrics-focused monitoring stack. Journal of Computer Allied Intelligence, 3(3), 28–39.
[8] Prometheus/Grafana for Microservices Jani, Y. (2024). Unified monitoring for microservices: Implementing prometheus and Grafana for scalable solutions. Journal of Artificial Intelligence, Machine Learning and Data Science, 2(1), 848–852.
[9] Infrastructure Monitoring with Prometheus P. B. C., Maddirala, H., & M. S. (2024). Implementing an effective infrastructure monitoring solution with Prometheus and Grafana. International Journal of Computer Applications, 186(38), 7–15.
[10] DevOps AdoptionTrigo, A.,Manteiga,A., Abar,S.,&Pardo,I.(2022). DevOpsadoption: Insights from a large European Telco. Cogent Engineering, 9(1), 2083474.
[11] Threshold-BasedAlertingTeklehaimanot,H.D.,Lipsitch,M.,Schwartzkopff,W.,&Eshetu,M. M.(2004). Alert threshold algorithms and malaria epidemic detection. Emerging Infectious Diseases,10(7),1221–1226.
[12] AdaptiveAlertingZhou,X.,&Liao,P.(2022).EEG-basedperformance-drivenadaptive automated hazard alerting systemin securitysurveillance support. Sustainability, 15(6), 4812.
[13] CIandNon-FunctionalRequirements(Monitoring)Yu,L.,Alégroth,E.,Chatzipetrou,P., & Gorschek, T. (2020). Utilising CI environment for efficient and effective testing of NFRs. Information and Software Technology, 117, 106199.
[14] Continuous Software Engineering & Architecture O'Connor, R. V., Elger, P., & Clarke, P M.2017).Continuoussoftwareengineering,Amicroservicesarchitectureperspective.JournalofSoftware: Evolution and Process, 29(11), e1866.
[15] Remote Command Execution Security (Detection) Meier, R. (2019). Detection of maliciousremoteshellsessions.In201911thInternationalConferenceonCyberConflict.Silent Battle (pp. 377–388). NATO CCD COE Publications.
[16] Remote Access Security Standards Ylonen, T., & Saarinen, S. (2015). Security of interactive and automated access management using Secure Shell(SSH) (NIST Inter agency Report No. NIST IR 7966). National Institute of Standards and Technology.
[17] Time-Series Database (TSDB) Comparison Grzesik, P., & Mrozek, D. (2020). Comparative analysis of time series databases in the context of edge computing for low power sensor networks. In Lecture Notes in Computer Science (Vol. 12204, pp. 317–330). Springer.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details