Engineering Decision Science

DIGITAL HEALTH SOFTWARE PLATFORMS FOR REMOTE PATIENT CARE, TELECONSULTATION, AND CONTINUOUS MEDICAL FOLLOW UP

Ahmed Hassan
Egypt

ABSTRACT

Digital health software platforms provide an integrated approach to remote patient care, teleconsultation, and continuous medical follow up. This study examines a platform that connects patients, physicians, nurses, and caregivers through secure video consultations, electronic records, appointment scheduling, prescription management, and remote monitoring tools. Data from wearable devices, home diagnostic equipment, and patient-reported symptoms can be transmitted in real time for clinical review. Automated alerts help identify abnormal vital signs, missed medications, or worsening conditions, allowing timely intervention. The platform can also support follow-up reminders, digital care plans, laboratory result sharing, and communication between healthcare teams and patients. Role-based access, encryption, consent management, and audit trails are included to protect privacy and maintain accountability. Overall, digital health platforms can reduce unnecessary hospital visits, improve access for rural and mobility-limited patients, strengthen continuity of care, and enable more responsive, coordinated, and patient-centered healthcare delivery across different locations and stages of treatment.

keywords: Digital Health Platforms; Remote Patient Care; Teleconsultation; Continuous Medical Follow Up; Remote Patient Monitoring.

1. Introduction

Enterprise applications depend on reliable diagnostics to detect runtime errors, failed workflows, API issues, slow transactions, and integration faults. Logs are the main evidence source for understanding what happened before, during, and after an incident. Effective application diagnostics require structured logging, traceability, contextual metadata, error classification, and operational visibility [1]. However, fixed logging strategies often create two problems. Low-detail logs may miss important causes, while high-detail logs may generate excessive storage cost and alert noise.

The main challenge is that diagnostic needs change during runtime. A normal transaction may require only basic logs, but a failing workflow, repeated exception, or security-sensitive event may need deeper traces. Adaptive logging improves this process by changing log depth according to system condition, transaction risk, and error behavior [2]. AI-assisted monitoring can identify when additional diagnostic detail is needed and when logging can return to normal levels. This article proposes an adaptive logging framework for enterprise application diagnostics.

2. Methodology

The proposed framework captures log metadata from application servers, API gateways, databases, background jobs, authentication modules, and integration services. It records event type, severity, timestamp, user role, transaction ID, request path, error frequency, response time, affected component, and workflow state. These features are converted into diagnostic context profiles. Adaptive logging requires context because the same event may be low-risk in one workflow but critical in another [3].

The logging controller classifies runtime states as normal, warning, error-prone, performance-risk, security-sensitive, workflow-critical, or incident-active. Based on this classification, it adjusts log level, trace depth, parameter capture, correlation ID tracking, and diagnostic enrichment. Lightweight scoring and policy rules are used so logging decisions remain fast and explainable [4]. Sensitive fields are masked before storage, and transaction-critical events are linked with workflow and database identifiers for easier diagnosis.

The framework is evaluated using simulated enterprise applications involving APIs, database workflows, authentication events, scheduled jobs, and service integrations. Static logging is compared with adaptive logging under repeated exceptions, slow requests, failed jobs, API timeout, suspicious login, and workflow interruption scenarios. Evaluation metrics include diagnosis time, log volume reduction, root-cause evidence quality, missed-event rate, storage overhead, alert usefulness, and administrator acceptance rate [5]. Feedback from resolved incidents, ignored logs, and administrator notes is used to refine logging policies over time.

3. Results and Discussion

The results show that adaptive logging improves diagnostics by capturing richer evidence only when runtime risk increases. In the baseline condition, static logs either lack enough detail or produce large volumes of low-value records. With the proposed framework, detailed traces are activated for high-risk events and reduced during normal execution. The strongest improvement appears in intermittent failures, where deeper logs are needed only around the failure window.

The discussion shows that adaptive logging must balance diagnostic depth with privacy and storage cost. Excessive logging can expose sensitive data or overload monitoring systems, while weak logging may hide failure causes. The main limitation is that log-level decisions depend on accurate runtime classification. Regular policy review and masking controls are necessary for safe long-term use.

4. Conclusion

This article presented an adaptive logging framework for enterprise application diagnostics. The framework adjusts log detail using runtime context, risk scoring, workflow state, and error behavior. It improves troubleshooting by increasing diagnostic evidence during abnormal conditions while reducing unnecessary log noise during normal operation. The study shows that adaptive logging can strengthen enterprise diagnostics when combined with structured metadata, privacy controls, and continuous incident feedback.

References

1. Keshireddy, S. R. (2019). Oracle APEX Integration with RESTful Services for Lightweight Enterprise Data Exchange. Journal of Grid, Machines and Drives, 6, 7-12.
2. Kavuluri, H. V. R. (2020). Software Reliability Prediction with Weibull Failure Models. Emerging Digital Systems, 7, 7-12.
3. Kavuluri, H. V. R. (2019). Defect Prediction in Object-Oriented Software with Decision Tree Classifiers. Transactions on Smart Electrical and Computational Systems, 6, 6-11.
4. Kavuluri, H. V. R. (2019). Bug Severity Classification in Issue Tracking Systems with Support Vector Machines. High-Performance Distributed Computing Review, 6, 7-12.
5. Kavuluri, H. V. R. (2020). Feature Selection for Machine Learning Fault Prediction in Software Modules. Journal of Privacy-Aware Computing Systems, 7, 7-12.