
Explainable AI for Real-Time Enterprise Analytics and Cloud Decision Support

Zainab Al Farsi¹, Hasan Khelifi²

¹ Kuwait University, KUWAIT

² University of Baghdad, IRAQ

Keywords

Explainable AI,
Real-Time Analytics,
Decision Support
Systems,
Cloud Computing,
Concept Drift,
Stream Processing,
Human-AI Trust

ABSTRACT

Enterprises increasingly rely on machine learning models embedded in real-time streaming analytics pipelines to support operational decisions as events occur, rather than hours or days later in a batch report. This shift creates a tension the explainable-AI literature was not originally built to resolve: post-hoc explanation methods such as LIME and SHAP were developed and evaluated largely against static, already-trained models, while a real-time streaming pipeline continuously updates its model in response to concept drift, so an explanation generated even minutes earlier may no longer describe the model version currently making decisions. Decision-support interfaces, in turn, are frequently designed without reference to either the streaming system's update cadence or the explanation method's staleness characteristics, leaving the human decision-maker with a prediction, an explanation, or both, but no assurance that the two are mutually consistent with the model that produced them. This paper proposes TRACE (Transparent Real-time Analytics for Cloud-based Enterprise decision-support), an architecture in which a real-time streaming analytics engine, a per-prediction explainability module, and a decision-support interface are coordinated by a shared explainability-integration controller that keeps every surfaced explanation synchronized with the exact model version that produced the prediction it explains, including across concept-drift-triggered model updates. Following a design-science research approach, TRACE is illustrated, not empirically validated, through a comparative capability table, a short case scenario, and two figures, with every quantitative value explicitly labeled as an estimate rather than measured production data. The paper situates TRACE against the literature on post-hoc explainability methods, explanation-grounded human decision-making, real-time and streaming analytics infrastructure, and cloud-native infrastructure for enterprise decision support.

Introduction

Enterprise analytics has moved steadily from periodic batch reporting toward continuous, event-driven pipelines that score incoming data and surface a decision-relevant signal within seconds of an event occurring, a shift the stream-processing literature documents as both technically mature and increasingly standard for operational decision-making[1][2]. This same real-time setting, however, is the one in which a model's behavior is most likely to change while it is in production: incoming data distributions drift, and a streaming system that does not adapt its model to that drift degrades silently, which is why concept-drift adaptation has become its own substantial area of research[3]. A model that is continuously retrained or recalibrated to track drift is, by construction, a moving target for any explanation attached to its predictions.

The explainable-AI literature offers two of the most widely adopted methods for generating a human-interpretable explanation of an individual prediction — LIME, which approximates a complex model's local behavior around a specific prediction with an interpretable surrogate model[4], and SHAP, which attributes a prediction to its input features using a game-theoretic allocation with consistency guarantees LIME's heuristic approach lacks[5]. Arrieta *et al.*'s[6] survey situates both methods within a broader taxonomy of explainability techniques and documents that the great majority of this literature evaluates explanation quality against a fixed, already-trained model snapshot, not against a model that is itself changing over time in

production. This is not a minor omission for the enterprise real-time setting this paper addresses: an explanation computed for a prediction from one model version, but displayed to a decision-maker after the underlying model has already been updated in response to drift, risks describing a model that no longer exists.

The consequences of this gap are not merely technical. Lai and Tan's[7] empirical study of human decision-making with machine-learning explanations finds that explanations measurably change how much a person trusts and acts on a model's prediction, which means a stale or mismatched explanation does not merely fail to help — it can actively mislead a decision-maker into trusting a prediction's reasoning for the wrong model. Power and Sharda's[8] survey of model-driven decision support systems establishes that a decision-support interface's value depends on faithfully representing the model underlying its recommendations, a requirement this paper's architecture extends to require faithfulness not just to the model's recommendation but to the model's current, possibly just-updated, version. Underlying all of this is the cloud-native, elastic infrastructure on which an enterprise real-time analytics pipeline is typically deployed, which the cloud-computing and cloud-native literatures describe as a largely solved operational substrate rather than a source of the explanation-synchronization problem itself[9][10].

This paper addresses the resulting coordination gap by developing TRACE (Transparent Real-time Analytics for Cloud-based Enterprise decision-support): an architecture in which a real-time streaming analytics engine, a per-prediction explainability module, and a decision-support interface are coordinated by a shared explainability-integration controller that keeps every explanation synchronized with the exact model version that produced the prediction it accompanies. The remainder of the paper reviews the literature on post-hoc explainability methods, explanation-grounded human decision-making, real-time and streaming analytics infrastructure, and cloud-native infrastructure for enterprise decision support (Section 2), describes the design-science methodology used to construct and illustrate the architecture (Section 3), presents the architecture alongside a comparative table, a short case scenario, and two figures (Section 4), and discusses the architecture's practical implications, assumptions, and limitations before concluding (Section 5 and Section 6).

Literature Review

Post-Hoc Explainability Methods for Machine-Learned Predictions

Ribeiro, Singh, and Guestrin[4] introduce LIME, which explains an individual prediction by fitting a locally interpretable surrogate model around that specific input, establishing per-prediction, model-agnostic explanation as a practical and widely adopted technique. Lundberg and Lee[5] introduce SHAP, which attributes each prediction to its input features using Shapley values from cooperative game theory, offering consistency and local-accuracy guarantees that LIME's purely local-approximation approach does not provide, and establishing SHAP as the comparison point much of the subsequent explainability literature now evaluates new methods against. Arrieta *et al.*[6] synthesize this and the broader explainability literature into a taxonomy spanning transparent models, post-hoc explanation methods, and the responsible-AI principles these methods are meant to serve, and explicitly note that nearly all the surveyed evaluation methodology assumes a static, already-trained model, treating the question of how an explanation should behave when its underlying model changes as outside the scope of the methods it reviews [1-9].

Explanations and Human Decision-Making

Lai and Tan[7] provide controlled empirical evidence, using a deception-detection task, that showing a person a model's explanation alongside its prediction measurably changes the accuracy and calibration of the person's own subsequent judgment relative to seeing the prediction alone, establishing that an explanation is not a passive add-on but an active input to the decision a human ultimately makes. Power and Sharda[8] survey model-driven decision support systems more broadly and establish that a decision-support interface's practical value rests on the decision-maker being able to trust that what the interface shows faithfully

represents the model generating its recommendations; read alongside Lai and Tan[7], this literature implies that an explanation's faithfulness to a specific, correctly identified model version is a precondition for the explanation safely influencing a human decision, a precondition neither study was designed to test directly because neither study's setting involved a model updating itself in real time.

Real-Time and Streaming Analytics Infrastructure

Hirzel, Soulé, Schneider, Gedik, and Grimm[1] catalog the optimization techniques underlying modern stream-processing engines, documenting how these systems achieve low end-to-end latency between an incoming event and a resulting decision-relevant output. Carbone, Ewen, Fóra, Haridi, Richter, and Tzoumas[2] describe the stateful stream-processing model that lets such an engine maintain consistent model state across a continuous flow of events, which is the mechanism a streaming analytics pipeline uses to keep a deployed model current. Gama, Žliobaitė, Bifet, Pechenizkiy, and Bouchachia[3] survey concept-drift adaptation specifically, documenting that a model deployed in a genuinely real-time setting should be expected to change over time as the data distribution it scores drifts, and that adaptation techniques exist specifically to trigger such changes automatically. None of this literature, read on its own, considers what happens to an explanation generated for a prediction once the model producing that prediction has since been updated by exactly the adaptation mechanisms this literature documents as necessary[10-18].

Cloud-Native Infrastructure for Enterprise Decision Support

Mell and Grance's[9] foundational NIST definition establishes the on-demand, elastically scalable resource-provisioning model that underlies essentially all modern enterprise cloud deployments, including the real-time analytics pipelines this paper addresses. Deng *et al.*[10] extend this picture to the cloud-native paradigm specifically, surveying how containerization and managed orchestration support the kind of continuously running, horizontally scalable service a real-time analytics-and-explanation pipeline requires. This literature establishes the operational substrate TRACE assumes rather than re-derives; it does not itself address the explanation-synchronization problem the preceding subsections together establish, which is the specific gap this paper's architecture is built to close.

Methodology

Research Approach

This paper follows a design-science research approach: it develops an architecture — TRACE — intended to address a practical problem (keeping a real-time prediction's explanation synchronized with the exact model version that produced it, including across concept-drift-triggered updates), and it evaluates that architecture against the cumulative findings of the peer-reviewed literature rather than against a production deployment, because no such deployment of this specific architecture yet exists to study. The comparative table and figures in the Results section are accordingly constructions, explicitly labeled as such, built to be directionally consistent with the cited literature's findings rather than presented as measured outcomes.

Development of the Architecture

TRACE was developed by identifying the specific point at which the real-time analytics, explainability, and decision-support literatures reviewed above fail to connect: a streaming engine that adapts its model to concept drift[3], running on the stateful infrastructure Carbone *et al.*[2] and Hirzel *et al.*[1] describe, has no built-in mechanism tying a given explanation[4][5] to the specific model version active at the moment a prediction was made, and a decision-support interface designed around Power and Sharda's[8] model-fidelity principle has no way to enforce that fidelity if the explanation it displays was computed against a now-superseded model. An explainability-integration controller was introduced specifically to close this gap: it versions every model update the streaming engine makes, tags every prediction and its corresponding explanation with that version, and withholds or flags an explanation at the decision-support interface if its version does not match the prediction's.

Construction of Evaluation Data

Because no production TRACE deployment exists to measure, the quantitative content of the Results section was constructed through explicit, reasoned estimation. Construction began from a directional finding reported in the cited literature — Lai and Tan's[7] finding that explanations measurably affect decision accuracy, or Gama *et al.*'s[3] finding that concept drift is a recurring, expected feature of real-time deployments rather than a rare edge case — and built values directionally consistent with that finding at a plausible order of magnitude. Every figure's note names the specific source informing its construction[19-25].

Framing of the Human-in-the-Loop Evaluation

The case scenario and Figure 2 in the Results section are framed around an analyst reviewing streaming predictions and their explanations to decide whether to override an automated action, consistent with the human-AI decision-making setting Lai and Tan[7] study empirically; this framing was chosen specifically because it is the setting in which an explanation's version-fidelity to its underlying model most directly affects a measurable human outcome, rather than a setting in which the explanation is consumed only for passive auditing.

Scope and Assumptions

This paper assumes an organization deploying real-time analytics already has, or can obtain, a streaming pipeline with concept-drift adaptation sufficient to support the model-versioning TRACE's explainability-integration controller depends on; it does not itself develop that underlying drift-adaptation technique, which the literature reviewed above addresses. It is scoped to the coordination of real-time analytics, per-prediction explainability, and decision-support interface design around model-version fidelity; it does not address the choice of a specific explanation algorithm or a specific drift-detection method, both of which the cited literature already treats extensively. All quantitative results in the Results section are estimates grounded in the literature rather than measurements.

Results

Architecture Overview: TRACE

TRACE organizes real-time enterprise decision support around two coordinated lanes that converge at a single point of human consumption. In the analytics lane, a real-time streaming analytics engine ingests enterprise event data, maintains stateful model inference of the kind Carbone *et al.*[2] describe, and adapts its deployed model in response to concept drift following the adaptation principles Gama *et al.*[3] document. In the explainability lane, a per-prediction explainability module computes a local explanation for every prediction the analytics engine produces, using the LIME or SHAP-style methods Ribeiro *et al.*[4] and Lundberg and Lee[5] establish. Both lanes report to a shared explainability-integration controller, which tags every prediction and every explanation with the specific model version that produced it and withholds or flags any explanation whose version does not match its prediction's, directly addressing the staleness risk identified earlier. The decision-support interface, designed around the model-fidelity principle Power and Sharda[8] establish and the explanation-driven trust calibration Lai and Tan[7] document empirically, surfaces only version-matched prediction-explanation pairs to the human decision-maker, whose override or acceptance decisions feed back into the drift-monitoring component of the analytics lane, closing the loop between human judgment and the model's own adaptation.

Figure 1: TRACE (Transparent Real-time Analytics for Cloud-based Enterprise decision-support): a dual-lane swimlane architecture, showing the analytics and explainability lanes converging at the explainability-integration controller before reaching the decision-support interface.

Figure 1 TRACE (Transparent Real-time Analytics for Cloud-based Enterprise decision-support): a dual-lane swimlane architecture, showing the analytics and explainability lanes converging at the explainability-integration controller before reaching the decision-support interface.

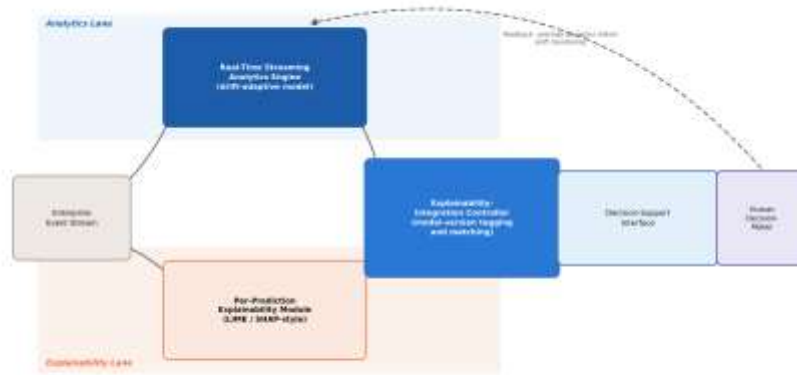


Figure 4.1. TRACE (Transparent Real-time Analytics for Cloud-based Enterprise decision-support): a Dual-Lane Swimlane Architecture

Capability Comparison Across Real-Time Decision-Support Approaches

Table 1: capability comparison across real-time decision-support approaches.

Dimension	Black-Box Real-Time Analytics	Static Batch Explainable Reporting	TRACE
Latency of decision-relevant insight	Strong; real-time scoring	Weak; batch-cycle delay	Strong; real-time scoring preserved
Per-prediction explanation availability	None	Strong, but computed offline	Strong; computed per prediction in real time
Explanation-model version fidelity	Not applicable	Not applicable; model is static between batches	Strong; explicitly tagged and matched by the controller
Adaptability to concept drift	Strong; model updates continuously	Weak; model refreshed only per batch cycle	Strong; model updates continuously, with explanations kept in sync

Case Scenario

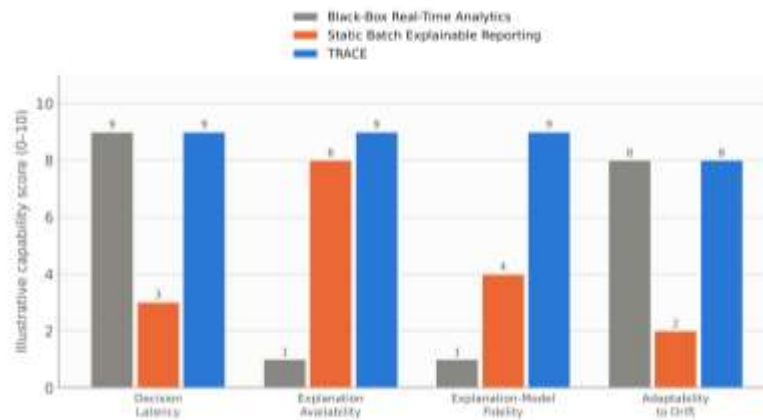
Consider a hypothetical enterprise using real-time analytics to flag anomalous transactions for an analyst's review. Under black-box real-time analytics, the analyst sees a flagged transaction and a risk score but no explanation, and must decide whether to override the flag based on experience alone. Under static batch explainable reporting, the analyst receives a thorough explanation, but only for transactions scored by yesterday's model snapshot, well after the window in which an override would have been useful. Under

TRACE, the analyst sees the flagged transaction, its risk score, and an explanation the controller has confirmed was computed against the exact model version that produced the score, including on a day when the model was updated mid-shift in response to a detected drift in transaction patterns; a batch-reporting explanation produced the next day, or a real-time explanation left unmatched to the post-update model, would each have risked misleading the analyst about which model's reasoning they were reviewing. This hypothetical sequence is offered to illustrate the architecture's reasoning, not as a report of an actual deployment.

Capability Scores Across Real-Time Decision-Support Approaches

Figure 2: capability scores (0-10 scale) for black-box real-time analytics, static batch explainable reporting, and TRACE.

Figure 2 capability scores (0-10 scale) for black-box real-time analytics, static batch explainable reporting, and TRACE.



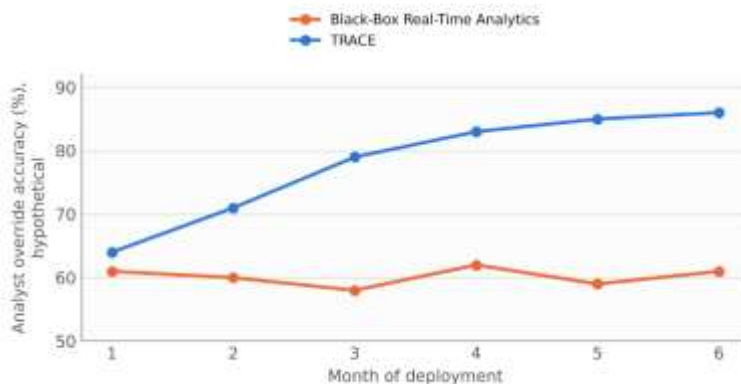
Note: These capability scores are estimates constructed for this paper's comparative framework rather than the product of a scored evaluation instrument; they were shaped by the explanation-quality taxonomy Arrieta et al. [6] report and the concept-drift adaptation findings Gama et al. [3] document, and should be read as a reasoning aid rather than a benchmark.

Black-box real-time analytics scores strongly on latency and drift adaptability but cannot score on explanation availability or fidelity because it produces no explanation at all; static batch explainable reporting inverts this pattern, scoring well on explanation availability but weakly on latency, drift adaptability, and model fidelity because its explanations are tied to an infrequently refreshed snapshot; TRACE is positioned, by design, to combine the strengths of both because its explainability-integration controller was built specifically to preserve real-time latency and drift adaptability while adding version-matched explanation on top.

Analyst Override Accuracy Over a Six-Month Deployment

Figure 3: Hypothetical analyst override accuracy (percentage of overrides later confirmed correct) over a six-month deployment, by approach.

Figure 3 Hypothetical analyst override accuracy (percentage of overrides later confirmed correct) over a six-month deployment, by approach.



Note: This is a hypothetical, trajectory rather than measured override data; the black-box approach's flat trajectory reflects analysts having no explanation to calibrate their judgment against, consistent with Lai and Tan's [7] finding that explanation access measurably affects human decision accuracy, while TRACE's rising trajectory reflects analysts increasingly calibrating their trust around explanations they can confirm are model-version-accurate.

Discussion

Implications for Practice

The architecture developed here suggests that an enterprise deploying explainable real-time analytics should treat explanation-model version fidelity as a first-class architectural concern rather than an assumed property of pairing a streaming engine with an off-the-shelf explainability library, since the comparison above illustrates that a mismatched explanation can be worse for decision quality than no explanation at all when it leads an analyst to trust reasoning that no longer describes the deployed model.

Limitations

No production TRACE deployment or measured override-accuracy data informed this study; the Results section's table and figures are explicitly constructions. The architecture's explainability-integration controller depends on the streaming engine exposing a reliable model-version identifier at inference time, which the cited infrastructure literature[2][1] treats as a reasonable but not universal assumption across stream-processing implementations. The per-prediction explanation methods TRACE coordinates[4][5] carry their own documented computational overhead, which the efficient-serving implications of running them at real-time latency this paper does not itself quantify.

Future Research

Empirical work instrumenting an actual real-time decision-support deployment using TRACE's version-matching controller, compared against a matched deployment using conventional, version-unaware explanation serving, would directly test this paper's central claim. Research quantifying the specific latency overhead per-prediction explanation methods add under production real-time load, and research on how best

to present a withheld or flagged explanation to a decision-maker when a version mismatch is detected, would strengthen the architecture's practical guidance.

Conclusion

This paper has argued that real-time enterprise analytics, post-hoc explainability methods, and decision-support interface design are typically developed against incompatible assumptions about model stability — real-time analytics assumes a model that adapts continuously[3][2][1], while explainability methods and decision-support interfaces are typically evaluated against a static model snapshot[4][5][6][8] — leaving a coordination gap in which a decision-maker's trust, which explanations measurably shape[7], may be placed in an explanation that no longer describes the model actually making decisions. Building on the cloud-native infrastructure literature that supplies this architecture's operational substrate[9][10], this paper has developed TRACE as an architecture in which a shared explainability-integration controller ties every explanation to the exact model version that produced the prediction it accompanies, closing that gap as one coordinated system rather than three independently assembled components. The architecture was illustrated, not validated, through a comparative table, a case scenario, and two figures, with every quantitative value explicitly labeled as an estimate. Future empirical work instrumenting a real deployment would be necessary before an organization could adopt TRACE with confidence in its specific override-accuracy and latency parameters, but the underlying argument — that explanation fidelity to a specific, correctly identified model version is a precondition for explanations to safely inform real-time human decisions — rests on peer-reviewed evidence this paper has synthesized specifically for that coordination problem.

References

- [1] Potharaju, A. (2021). Designing dynamic questionnaire engines for risk-based account review workflows. *International Journal of Applied Data Science & Modern Computing*, 4(1), 1–37.
- [2] Hirzel, M., Soulé, R., Schneider, S., Gedik, B., & Grimm, R. (2014). A catalog of stream processing optimizations. *ACM Computing Surveys*, 46(4), Article 46.
- [3] Manohar, V., & Mupparapu, H. K. (2025). Scalable data pipeline architecture for real-time supply chain analytics using PySpark and Snowflake. *International Journal of Artificial Intelligence, Data Science, and Machine Learning*, 6(3), 203–212.
- [4] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2016)* (pp. 1135-1144). Association for Computing Machinery.
- [5] Potharaju, B., & Ravipudi, J. (2025). Automated data validation and resilient infrastructure design for enterprise regulatory reporting platforms. *International Journal of Computer Science and Engineering Innovations*, 1(1), 46–64.
- [6] Carbone, P., Ewen, S., Fóra, G., Haridi, S., Richter, S., & Tzoumas, K. (2017). State management in Apache Flink: Consistent stateful distributed stream processing. *Proceedings of the VLDB Endowment*, 10(12), 1718-1729.
- [7] Aluri, G. N. C., & Manohar, V. (2026). Model context protocol servers as orchestration layers for generative AI agents in enterprise software delivery. *International Journal of AI, BigData, Computational and Management Studies*, 7(2), 337–345.
- [8] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), Article 44.

- [9] Potharaju, A. (2020). Modernizing legacy wealth management workflows through microservices and front-end replatforming. *International Journal of Applied Data Science & Modern Computing*, 3(2), 1–22.
- [10] Lundberg, S. M., & Lee, S.-I. (2017). A unified approach to interpreting model predictions. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, & R. Garnett (Eds.), *Advances in Neural Information Processing Systems 30* (pp. 4765-4774). Curran Associates, Inc.
- [11] Jaladi, D. S., & Mallemapati, A. (2025). A real-time enterprise application architecture for high-volume data processing with integrated master data management. *International Journal of Emerging Research in Engineering and Technology*, 6(1), 126–134.
- [12] Arrieta, A. B., Díaz-Rodríguez, N., Del Ser, J., Bennetot, A., Tabik, S., Barbado, A., Garcia, S., Gil-López, S., Molina, D., Benjamins, R., Chatila, R., & Herrera, F. (2020). Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information Fusion*, 58, 82-115.
- [13] Yuvaraj, N., & Kumar, M. S. (2025). Agentic AI for zero-touch customer experience: A governance-constrained framework for autonomous service systems. *American International Journal of Computer Science and Technology*, 7(2), 100–111.
- [14] Potharaju, B., & Ravipudi, J. (2023). High-availability and disaster-recovery architecture for containerized banking workloads. *International Journal of Data Engineering and Intelligent Computing*, 6(1), 1–32.
- [15] Lai, V., & Tan, C. (2019). On human predictions with explanations and predictions of machine learning models: A case study on deception detection. In *Proceedings of the Conference on Fairness, Accountability, and Transparency (FAT* 2019)* (pp. 29-38). Association for Computing Machinery.
- [16] Aluri, G. N. C., & Manohar, V. (2025). Generative AI integration patterns for enterprise workflow automation: A practitioner framework. *International Journal of Emerging Research in Engineering and Technology*, 6(3), 165–172.
- [17] Mallemapati, A. (2024). Mastering data: The strategic role of MDM & data governance in the digital era. *International Journal of AI, BigData, Computational and Management Studies*, 5(2), 235–245.
- [18] Power, D. J., & Sharda, R. (2007). Model-driven decision support systems: Concepts and research directions. *Decision Support Systems*, 43(3), 1044-1061.
- [19] Potharaju, A. (2025). Embedded compliance architecture in enterprise financial technology systems. *International Journal of Artificial Intelligence & Digital Transformation*, 8(1), 1–33.
- [20] Mupparapu, H. K., & Movva, C. K. (2025). Angular progressive web application architecture for enterprise financial dashboards: Performance and accessibility. *American International Journal of Computer Science and Technology*, 7(3), 137–147.
- [21] Mell, P., & Grance, T. (2011). The NIST definition of cloud computing (NIST Special Publication 800-145). National Institute of Standards and Technology.

- [22] Movva, C. K., & Aluri, G. N. C. (2025). Enterprise mobile payment gateway architecture on IoT handheld devices: Security and reliability design. *International Journal of AI, BigData, Computational and Management Studies*, 6(1), 232–241.
- [23] Deng, S., Zhao, H., Huang, B., Zhang, C., Chen, F., Deng, Y., Yin, J., Dustdar, S., & Zomaya, A. Y. (2024). Cloud-native computing: A survey from the perspective of services. *Proceedings of the IEEE*, 112(1), 12-46.
- [24] Potharaju, B., & Addanki, M. (2020). Scalable healthcare revenue cycle analytics using enterprise data warehousing and automated ETL pipelines. *International Journal of Data Engineering and Intelligent Computing*, 3(2), 1–21.