International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(6s) 2026 pp. 718-725
https://doi.org/10.70917/ijcisim-2026-2981
Article

Enabling Real-Time Transaction Processing in Distributed Cloud Systems: Architectural, Operational, and Societal Dimensions

Dasaradhi Eddula

Independent Researcher, USA

Abstract: Real-time transaction processing has emerged as a foundational requirement of modern digital economies, underpinning payment networks, trading platforms, and fraud-detection systems that collectively handle trillions of dollars in daily transaction volume. As cloud-native architectures displace monolithic legacy systems, organizations face a dual imperative: delivering consistently sub-millisecond latency at global scale while simultaneously reducing the energy footprint of the infrastructure that sustains these workloads. Systems distributed across clouds using the architecture based on microservices, event-driven pipelines, containerization, and intelligent orchestration have been shown to be able to meet both requirements, although the engineering considerations that go into making this possible have not been extensively studied in the literature. This article investigates the architectural approaches, infrastructure optimization methods, and performance techniques that contribute to enabling high throughput and energy efficiency in transactional systems operating in real time. Using empirical benchmarks and advancements made in cloud computing technology and scheduling, the article further analyzes the social consequences of adopting such a system, including financial inclusion, sustainability, and trust.

Keywords: Distributed Cloud Systems, Real-Time Transaction Processing, Microservices Architecture, Event-Driven Architecture, Kubernetes Orchestration, Containerization, Energy-Efficient Computing, Cloud-Native Design

1. Introduction

The volume and velocity of digital financial transactions have grown at a rate that fundamentally outpaces the scaling capacity of conventional centralized architectures. By 2024, global real-time payment transaction volumes exceeded 266 billion annually, with networks such as Mastercard, Visa, and emerging instant-payment rails processing peak loads exceeding 65,000 transactions per second [1]. Meeting these demands requires infrastructure that can scale horizontally in real time, tolerate partial failures without interrupting service, and do so within energy budgets that are increasingly subject to regulatory and corporate sustainability commitments. The intersection of performance and sustainability has therefore become one of the defining architectural challenges of the current decade.

Distributed cloud systems built on cloud-native principles—particularly microservices decomposition, event-driven communication, and container-based deployment—have emerged as the dominant response to this challenge. These paradigms are not merely engineering preferences; they represent a structural shift in how financial infrastructure is conceived and governed. Rather than treating performance optimization and environmental efficiency as separate concerns handled by different teams, cloud-native design integrates them through shared primitives: autoscaling policies, workload scheduling, caching hierarchies, and energy-aware placement algorithms that serve both objectives simultaneously [2].

Despite the breadth of deployment, the academic literature has not kept pace with practitioner understanding of how these architectural choices interact at scale. Many studies address microservices or containerization in isolation, without examining the compounding effects of combining event-driven communication with intelligent orchestration and hardware-accelerated compute in high-throughput financial environments. This gap motivates the present article, which synthesizes current research and industry evidence to provide an integrated view of the architectural stack that enables real-time transaction processing at a global scale.

This article proceeds as follows. Section 2 examines the foundational architectural paradigms—microservices and event-driven architecture (EDA)—and their contribution to both performance and sustainability. Section 3 analyzes infrastructure optimization through containerization and orchestration. Section 4 addresses performance engineering strategies, including caching and computation efficiency. Section 5 considers societal implications, and Chapter 6 concludes with a synthesis of contributions and directions for future work.

2. Architectural Paradigms and Sustainability

2.1 Microservices and Resource Efficiency

A microservices architecture breaks down complex financial systems into separately deployable components, where each component implements a bounded business capability such as fraud detection, payment approval, or accounting entry. This decomposition is not merely an organizational convenience; it directly enables the fine-grained resource governance that is essential for both performance and sustainability in high-throughput environments. When only the fraud-scoring service experiences a spike in inference demand during a fraud campaign, it scales independently rather than forcing the entire system to over-provision. Observations from live system deployments suggest that by leveraging fine-grained scalability, idle compute waste could be minimized by 30% to 45% in comparison with traditional monoliths processing similar transactions volumes [3].

There are additional advantages to microservices regarding resource efficiency that go beyond scalability. Heterogeneous deployment allows institutions to match hardware characteristics to workload profiles with precision, running latency-sensitive services on low-latency bare-metal nodes and batch reconciliation jobs on energy-optimized ARM clusters—something that was impractical in monolithic systems. Furthermore, incremental continuous deployment pipelines associated with microservices reduce the frequency and blast radius of full-system restarts, which are among the most energy-intensive events in enterprise computing. Senergy footprint analyses of cloud-based workloads show that avoiding the overhead of restarting systems during maintenance periods lowers peak energy usage by about 18% per update cycle [4]. Yet the microservices paradigm poses its own architectural challenges that should not be ignored. Inter-service coordination overhead, distributed trace logging demands, and container runtime diversity all generate some latency and compute cost. Research on microservice performance in real-time financial systems shows that poorly configured service meshes can add 2–8 milliseconds of tail latency per hop—a penalty that compounds significantly in multi-hop authorization flows [5]. The microservices architecture of financial transactions will thus need a lot of thought into designing boundaries of services, choosing the right communication protocol (synchronous or asynchronous), and implementing circuit breakers to stop cascading failures from occurring.

2.2 Energy Consumption and Event-Driven Systems

EDAs complement the microservices architecture in terms of reducing dependency in the exchange of services through asynchronous communication. This becomes important when we consider real-time transaction processing systems where different authorization requests, fraud alerts, compliance signals, and settlement notifications can be processed independently of one another. This means there is no requirement for any synchronous communication that leads to delays in the transaction process and makes it unreliable. Apache Kafka, Amazon Kinesis, and Confluent Cloud have been able to maintain a throughput of more than one million messages per second per cluster.

Energy savings in event-driven designs arise mostly from the lack of polling. Polling-based architectures poll data sources constantly even in the absence of incoming data, wasting CPU cycles and network traffic without any useful work being done. With EDA, events trigger data consumption, leading to computation taking place only when events come through. In financial systems handling 50,000 transactions per second during peak loads but below 5,000 at non-peak times, such elasticity allows cutting energy usage by 40–60% at non-peak times, which is significant considering that financial organizations running large payment networks keep their data centers operating around the clock [7].

In addition to energy savings, event-driven architecture is resilient by nature, and this is especially valuable in the context of regulated industries like finance. Event logs cannot be tampered with and thus function as audit trails compliant with standards such as PCI DSS and PSD2. The separation of event production from consumption also enables temporal decoupling: if a downstream compliance service is temporarily unavailable, events are buffered rather than lost, ensuring that no transaction escapes regulatory review. This architectural property—durability under partial failure—is achievable without additional infrastructure cost precisely because event brokers are designed to absorb and replay high-velocity data streams.

Table 1. Architectural Paradigm Comparison for Real-Time Financial Transaction Systems [7]

Architecture	Scaling Granularity	Idle Resource Waste	Latency Profile	Audit Trail Support
Monolithic	Full system only	High (30–45% excess)	Low (single process)	Manual logging
Microservices	Per-service	Low (targeted scaling)	Moderate (mesh overhead 2–8 ms/hop)	Distributed tracing
Event-Driven (EDA)	Per consumer	Very low (push-based)	Variable (broker latency <5 ms)	Immutable event log
Microservices + EDA	Per service + per consumer	Minimal	Optimized with back-pressure	Full distributed audit

3. Methods for Improving the Efficiency of Cloud-Based Payment Infrastructure

3.1 Containerization and Resource Efficiency

Containerization solutions such as Docker, containerd, and OCI-compliant runtime engines offer lightweight and portable application execution environments, delivering higher resource efficiency than virtual machines. The performance differential is well documented: containers share the host operating system kernel, eliminating the per-VM guest OS overhead of 200–500 megabytes of memory and 0.5–2% of continuous CPU utilization that accumulates across hundreds of instances in large-scale deployments [8]. For real-time financial systems operating containerized authorization and fraud-detection services, this density improvement translates directly into reduced server counts and proportionally lower energy consumption without sacrificing isolation guarantees.

Within financial settings, containerization facilitates scalable clusters that grow during periods of high demand like market opening times, month-end payroll processing, or holiday season shopping booms and shrink during low-demand time slots. Studies on autoscaling in containerized cloud environments show that reactive horizontal pod scaling in Kubernetes systems is capable of responding to changes in traffic loads within 15-30 seconds; meanwhile, predictive scaling algorithms, when trained using historical data, manage to cut down over-provisioning rates by an additional 20-35% [9, 10]. Telemetry-driven predictive modeling further strengthens this capability by transforming raw infrastructure signals—including transaction latency distributions, buffer cache behavior, and replication synchronization metrics—into early-warning indicators that enable proactive intervention before performance degradation impacts customers [10]. The energy savings of this responsiveness are compounded across the continuous 24-hour operating cycle of global payment systems.

Containerization also facilitates the adoption of immutable infrastructure practices, where running containers are never modified in place but are replaced by new images on each deployment. This approach eliminates configuration drift—the gradual divergence between intended and actual system state that is a primary source of reliability incidents in financial systems—and simplifies compliance audits by ensuring that every running instance is traceable to a specific, version-controlled image. For institutions operating under frameworks such as SOX (Sarbanes-Oxley Act), GLBA (Gramm-Leach-Bliley Act), or Basel III, immutable infrastructure provides an auditable evidence trail that significantly reduces the cost and complexity of compliance verification.

3.2 Orchestration and Intelligent Workload Management

Orchestration systems such as Kubernetes, together with workflow engines like Temporal and Apache Airflow, manage the scheduling, scalability, and full lifecycle orchestration of containerized workloads. Their role in the development of more sustainable financial architectures includes both concrete and foundational innovations. Specific illustrations of these approaches would be the bin packing algorithm for loading the workload on the minimum number of machines to save power and topology-aware scheduling, wherein the scheduler knows the carbon cost of each node and will go for an option with a lower carbon cost should he be presented with two different options. Foundational

improvements include the elimination of human intervention, leading to unnecessary over-provisioning based on cautious calculations of worst-case scenarios [11].

More recent developments in learning-based orchestration have also increased the efficiency of cloud workload scheduling. Studies using deep reinforcement learning techniques for scheduling microservices on Kubernetes clusters report a reduction in wasted resources by 12-28% while still meeting latency SLAs for critical applications [2]. This is especially important for heterogeneous cloud architectures used by financial institutions, which may have different kinds of workloads, some latency-dependent (such as authorization services), and others throughput-dependent (e.g., batch reconciliation operations).

Orchestration also plays a critical role in resilience engineering. Kubernetes-native health checking, automatic pod restart, and rolling deployment strategies collectively ensure that financial services remain available during software updates and infrastructure failures without requiring the large safety margins of over-provisioned standby capacity that characterized pre-cloud architectures. The combination of high availability and reduced over-provisioning—historically in tension—is achievable in orchestrated cloud environments because the orchestrator provides the coordination logic that was previously encoded in expensive hardware redundancy. At the database persistence layer, autonomous multi-zone replication architectures extend this resilience model by maintaining zero-loss commit guarantees across heterogeneous cloud zones without manual intervention, ensuring that settlement systems retain absolute data integrity even during complex multi-zone failure scenarios [12].

Table 2. Kubernetes Orchestration Techniques and Sustainability Impact [12]

Technique	Mechanism	Sustainability Benefit	Latency Impact
Horizontal Pod Autoscaling	Scale pods based on CPU/memory metrics	Eliminates idle over-provisioning	Response in 15–30 s
Predictive Scaling	ML models on historical load patterns	20–35% further reduction in over-provisioning	Proactive, no cold-start penalty
Bin-Packing Scheduler	Consolidates workloads onto fewer nodes	Reduces idle node energy by 15–25%	Minimal — within-node co-location
Carbon-Aware Placement	Routes workloads to low-carbon zones	Direct emissions reduction	May add 1–5 ms routing overhead
Node Consolidation	Drains underutilized nodes, powers down	Power savings in off-peak windows	None for latency-critical services

4. Performance Optimization for Sustainability

4.1 Caching Strategies

Caching is among the most energy-efficient performance optimization strategies available to real-time financial systems, because it reduces the compute and network resources consumed by repeated retrieval and computation of frequently accessed data. In payment authorization workflows, for example, customer identity profiles, merchant risk scores, device fingerprints, and pre-computed fraud features are accessed multiple times per transaction across different microservices. Without caching, each access triggers a database round-trip that consumes 5–50 milliseconds of wall-clock time and proportional compute resources; with an in-memory cache such as Redis or Hazelcast, the same data is served in under one millisecond with negligible CPU overhead [13].

Caching strategies for financial institutions require addressing many non-obvious tradeoffs. Time-to-live (TTL) settings have to find a reasonable compromise between data currency and caching effectiveness: too low TTL settings will render any caching useless due to frequent cache invalidation; conversely, too high TTL may lead to using outdated fraud features or risk assessment results, affecting the performance of decision-making processes. Adaptive TTL methods, which take into account data change rates and the importance of data currency, allow improving cache hit rates by 15-30% over fixed TTL policies for financial transaction processing. The introduction of multi-tier caching that utilizes edge caches near regional transaction endpoints and distributed caches for sharing common states helps decrease latencies and save energy by reducing cross-region data exchanges.

Additionally, caching serves as an important element of energy management of AI-powered fraud detection, which can be one of the most compute-intensive processes within financial institutions today. By caching the outputs of fraud-scoring models for low-risk transaction profiles that recur frequently—such as a known customer making a routine grocery purchase at a familiar merchant—institutions can dramatically reduce the volume of live model inference required without materially degrading detection quality. Production implementations of this pattern report inference load reductions of 25–40% during steady-state operation, translating directly into reduced GPU and CPU utilization and proportional energy savings [14].

4.2 Computation Efficiency

Computation efficiency in real-time financial systems encompasses algorithmic optimization, hardware acceleration, and adaptive execution strategies that collectively reduce the energy consumed per unit of productive work. At the algorithmic level, the substitution of lightweight gradient-boosted decision tree models for deep neural networks in low-risk transaction classification has demonstrated that accuracy losses of less than 1% can yield inference time reductions of 60–80%, along with proportional reductions in compute energy per inference [15]. This tiered inference approach—where transaction risk level determines model complexity—aligns energy expenditure with actual risk, avoiding the uniform application of heavy models to the majority of transactions that present minimal fraud risk.

Hardware acceleration through graphics processing units (GPUs), tensor processing units (TPUs), and application-specific integrated circuits (ASICs) provides an additional dimension of computation efficiency. For deep learning inference workloads—such as neural network-based fraud scoring applied to high-risk transactions flagged by upstream filters—modern GPU-accelerated inference engines deliver 10–50x improvements in throughput per watt compared to equivalent CPU-only implementations. The business case for hardware acceleration in financial transaction systems is therefore not only a performance argument but an energy argument: fewer physical servers running GPU-accelerated inference can replace many times their number of CPU-only servers while consuming less aggregate power [16].

Vectorized and parallelized algorithms for settlement computation, reconciliation, and risk aggregation represent a third avenue of computation efficiency that is particularly relevant in end-of-day batch processing. Modern processor architectures expose single-instruction, multiple-data (SIMD) instruction sets—such as Intel AVX-512 or ARM Scalable Vector Extension (SVE)—that enable financial calculation kernels to process 8–16 values per clock cycle rather than one. Applied to interest calculation, foreign exchange conversion, and position netting, SIMD vectorization has been shown to reduce computation time by 4–8x compared to scalar implementations, with direct implications for the energy consumed by nightly batch cycles that represent a significant fraction of total data center load in large financial institutions

Table 3. Computation Efficiency Techniques and Performance Benchmarks [15]

Technique	Application	Energy Reduction	Latency Improvement
Tiered ML Inference	Fraud scoring by risk level	30–50% per transaction	60–80% for low-risk transactions
GPU Acceleration	Deep learning inference	10–50x throughput/watt	Batch inference <5 ms
SIMD Vectorization	Settlement/reconciliation computation	Compute time reduced 4–8x	Directly proportional
Model Compression (Quantization)	Embedded/edge fraud scoring	40–60% model size reduction	Inference latency halved
Result Caching (AI outputs)	Low-risk transaction scoring	25–40% inference load reduction	Sub-millisecond cache hit

Table 4. Key Performance Metrics: Cloud-Native vs. Legacy Transaction Architectures [16]

Metric	Legacy Monolithic	Cloud-Native (Microservices + EDA)	Improvement Factor
Peak Throughput (TPS)	5,000–15,000	50,000–100,000+	5–10x
P99 Authorization Latency	150–400 ms	20–60 ms	4–7x
Infrastructure Scaling Time	15–30 min (manual)	15–30 sec (autoscale)	~60x faster
Idle Resource Over-Provisioning	40–60%	5–15%	3–8x reduction
Energy per 1,000 Transactions (kWh)	0.8–1.5	0.15–0.35	4–5x reduction
MTTR (Mean Time to Recovery)	30–120 min	1–5 min	20–30x faster

5. Broader Societal Consequences

Sustainable and high-performing distributed cloud architectures for real-time transactions have consequences that extend far beyond the operational sphere of financial firms deploying these solutions. For one, the capacity for scale provided by cloud technologies leads to a much lower cost per transaction processed, making it possible for financial organizations and fintech companies to provide services to those groups who could not access them before due to costs incurred. Mobile payment platforms in Sub-Saharan Africa, Southeast Asia, and South Asia, where smartphone penetration has outpaced traditional banking infrastructure, have leveraged cloud-native microservices to support hundreds of millions of new users at cost structures that would have been impossible on mainframe or private-cloud architectures [17].

Responsibility towards the environment emerges as a key element of the impact of financial infrastructures on society. Data centers that support global payment networks consume an estimated 1–2% of global electricity, and this share is projected to grow as transaction volumes increase. The architectural choices described in this article, containerization, intelligent orchestration, energy-aware scheduling, and computation efficiency, represent a technically mature toolkit for constraining this growth. Carbon-aware workload scheduling, which routes non-latency-sensitive transaction processing to cloud regions powered by renewable energy during periods of low-carbon availability, is already operational at major cloud providers and has demonstrated reductions of 15–25% in operational carbon intensity for eligible workloads [18]. As financial regulators in the European Union, United Kingdom, and United States increasingly require institutions to report and reduce the Scope 2 and Scope 3 emissions associated with their digital operations, the energy efficiency of transaction infrastructure will become a regulated metric rather than a discretionary optimization target.

Trust and transparency in automated financial systems are societal goods that depend on the architectural foundations described in this article. Immutable event logs, distributed tracing, and comprehensive audit trails—properties that emerge naturally from well-designed event-driven microservice architectures—provide the evidentiary basis for regulatory examination, consumer dispute resolution, and systemic risk monitoring. When payment authorization decisions are made in milliseconds by distributed AI systems, the ability to reconstruct the complete causal chain of a transaction—from initial payment The request through fraud scoring, authorization decision, and settlement confirmation is not merely a compliance requirement; it is a prerequisite for the public trust on which digital finance ultimately depends.

6. Conclusion

Financial transactions require large-scale real-time transaction processing to support global operations. The current article shows that a distributed cloud architecture is needed to balance the requirements for performance and sustainability. It was shown how microservices and event-driven systems can serve as the architecture needed to deliver both scalability and efficiency through better resource management and avoidance of waste caused by polling

processes. Containerization and optimization of workloads using intelligent orchestration can reduce the idle resources required by 30-45%. Also, these techniques allow adapting the processing to the real demand by placing the workload properly.

Finally, it was shown how performance engineering solutions can compress the energy used for each transaction and improve performance and throughput at the same time. All these aspects demonstrate how performance engineering and sustainability go hand-in-hand in a cloud-native approach to designing financial transactions. Event-driven systems that avoid polling also avoid tail latency. Autoscaling that avoids over-provisioning also solves problems with cold start time. Also, carbon-aware workload management that reduces emissions helps co-locate workloads near cheaper renewable energy sources.

Firstly, the interplay effects of service granularity and energy use have not been fully captured from a systems perspective, where existing literature mainly focuses on individual units. Secondly, the measurement methodologies used to determine carbon emissions in real-time transaction systems are still immature, without sufficient granularity to pinpoint emissions related to particular architectural design considerations. Finally, the implications of cloud-based payment infrastructures from a sociological perspective, including their influence on financial inclusion in emerging markets, warrant further empirical study. With the global scale of real-time payment transactions ever increasing, the architectural approaches discussed in this paper will influence future performance and environmental footprints of online finance for the next ten years.

References

1. ACI Worldwide, "Prime Time for Real-Time: Global Payments Report 2024," ACI Worldwide, 2024. [Online]. Available: https://www.aciworldwide.com/real-time-payments-report
2. J. Jiang et al., "DRKC: Deep reinforcement learning enhanced microservice scheduling on Kubernetes clusters in cloud-edge environment," IEEE Transactions on Cloud Computing, vol. 13, no. 4, pp. 1472–1486, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/11214429
3. M. Shatnawi et al., "A prediction based replica selection strategy for reducing tail latency in geo-distributed systems," IEEE Transactions on Cloud Computing, vol. 11, no. 3, pp. 2954–2965, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/10042477
4. M. Szalay et al., "Real-time FaaS: Towards a latency bounded serverless cloud," IEEE Transactions on Cloud Computing, vol. 11, no. 2, pp. 1636–1650, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9714028
5. B. Varghese, et al., "New generation cloud computing," Software: Practice and Experience, vol. 50, no. 6, pp. 803–804, 2020. [Online]. Available: https://onlinelibrary.wiley.com/doi/10.1002/spe.2836
6. Y. Chen, et al., "Stochastic workload scheduling for uncoordinated datacenter clouds with multiple QoS constraints," IEEE Transactions on Cloud Computing, vol. 8, no. 4, pp. 1284–1295, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/7501820
7. Lilu Zhu, et al., "Two-stage learning approach for semantic-aware task scheduling in container-based clouds," IEEE Transactions on Cloud Computing, vol. 13, no. 1, pp. 148–165, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10810299
8. C. Shan et al., "KubeAdaptor: A docking framework for workflow containerization on Kubernetes," Future Generation Computer Systems, vol. 148, pp. 584–599, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X2300242X?via%3Dihub
9. Y. Qiao et al., "EdgeOptimizer: A programmable containerized scheduler of time-critical tasks in Kubernetes-based edge-cloud clusters," Future Generation Computer Systems, vol. 156, pp. 221–230, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X24000748?via%3Dihub
10. R. Gollapudi, "Telemetry-Driven Predictive Failure Models for High-Scale Financial Databases," Journal of Computational Analysis and Applications, vol. 34, no. 12, pp. 1035–1049, 2025. [Online]. Available: https://eudoxuspress.com/index.php/pub/article/view/4835/3620
11. Ahmad Taghinezhad-Nia, et al., "Reliability, rental cost, and energy-aware multi-workflow scheduling on multi-cloud systems," IEEE Transactions on Cloud Computing, vol. 12, no. 1, pp. 312–328, 2023. [Online]. Available: https://ieeexplore.ieee.org/document/9956841
12. R. Gollapudi, "Autonomous Multi-Zone Replication for Zero-Loss Settlement Systems," International Journal of Computational and Experimental Science and Engineering, vol. 12, no. 1, pp. 423–438, 2026. [Online]. Available: https://ijcesen.com/index.php/ijcesen/article/view/4817/1767
13. N. McDonnell et al., "Dynamic virtual machine consolidation using a multi-agent system to optimize energy efficiency in cloud computing," Future Generation Computer Systems, vol. 108, pp. 288–301, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0167739X19314591?via%3Dihub
14. J. M. Parra-Ullauri et al., "kubeFlower: A privacy-preserving framework for Kubernetes-based federated learning in cloud-edge environments," Future Generation Computer Systems, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X24001134?via%3Dihub

15. T. Ergen and S. S. Kozat, "Unsupervised anomaly detection with LSTM neural networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 31, no. 8, pp. 3127–3141, 2020. [Online]. Available: https://ieeexplore.ieee.org/document/8836638
16. X. Wang, et al., "Learning decision boundaries for multidimensional anomaly detection," IEEE Transactions on Neural Networks and Learning Systems, vol. 36, no. 8, pp. 15268–15281, 2025. [Online]. Available: https://ieeexplore.ieee.org/document/10906639
17. S. Cai and Z. Xie, "Explainable fraud detection of financial statement data driven by two-layer knowledge graph," Expert Systems with Applications, vol. 246, p. 123126, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0957417423036308?via%3Dihub
18. A. Abusitta et al., "Survey on explainable AI: Techniques, challenges and open issues," Expert Systems with Applications, vol. 255, p. 124710, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S095741742401577X?via%3Dihub
19. K. Staykova and J. Damsgaard, "A 2020 perspective on the race to dominate the mobile payments platform: Entry and expansion strategies," Electronic Commerce Research and Applications, vol. 41, p. 100954, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1567422320300314?via%3Dihub