International Journal of Computational and Experimental Science and ENgineering
(IJCESEN)

Vol. 11-No.4 (2025) pp. 7653-7662
http://www.ijcesen.com

ISSN: 2149-9144

Review Article

Integrating ai into enterprise java applications for secure high performance and scalable systems

Ishu Anand Jaiswal*

University of the Cumberlands, 6178 College Station Drive, Williamsburg, KY 40769
* **Corresponding Author Email:** ishuanand.jaiswal@gmail.com- **ORCID:** 0000-0002-5247-7050

Article Info:

DOI: 10.22399/ijcesen.4086
Received : 01 March 2025
Accepted : 29 March 2025

Keywords

Artificial Intelligence (AI),
Enterprise Java Applications,
Scalability,
Secure AI Integration,
High-Performance Computing,
Microservices Architecture

Abstract:

The integration of Artificial Intelligence (AI) into enterprise Java applications is rapidly emerging as a transformative approach to building intelligent, secure, and scalable systems. Traditional enterprise applications, though robust, often lack the adaptive capabilities required to handle modern workloads such as predictive analytics, anomaly detection, and intelligent automation. This paper explores a framework for embedding AI within enterprise Java environments by leveraging contemporary machine learning libraries, cloud-native deployments, and microservice architectures. Emphasis is placed on achieving high performance and scalability while addressing critical security challenges, including data privacy, model integrity, and secure inference. Through a proposed reference architecture and a case study implementation, the paper evaluates performance benchmarks, security considerations, and scalability trade-offs. The findings highlight that AI-enabled enterprise Java applications can provide significant improvements in system intelligence and efficiency, provided that integration is carefully designed with attention to performance optimization and security governance.

1. Introduction

One of the pillars of large business systems in financial services, healthcare, e-commerce and government sectors are enterprise Java applications [1]. They are common because Java is platform-independent, good type safety and a rich ecosystem of frameworks such as Spring Boot and Jakarta EE can be used to encourage modularity, maintainability, and reliability. Such applications have succeeded in decades to handle transactions, integrate an enterprise, and automate business processes. However, with the size of data constantly increasing exponentially, and with the demand to make decisions in real time and dynamically, traditional enterprise solutions are becoming less and less useful when it comes to addressing emerging challenges [2]. The modern world requires companies to seek applications that go beyond the bounds of a deterministic logic and rule based automation and seek applications that are intelligence based and have the capability to self-optimise, predict and personalise based on circumstances. Machine learning (ML) and deep learning, also called Artificial Intelligence (AI), is the computational underpinning of this paradigm shift. The artificial intelligence on the enterprise Java applications will enable them to possess formidable capabilities such as predictive maintenance, fraud detection, anomaly detecting, and intelligent customer engagements [3].

Despite its potential, there are technical and architectural problems on the way to implementation of AI into the enterprise Java environments. Python-based ecosystems and R-based ecosystems often incorporate machine learning models, and machine learning libraries such as TensorFlow, PyTorch and Scikit-learn are natively supported [4]. There are several Java-native systems such as DeepLearning4j, Tribuo, and DJL (Deep Java Library) which have been created but not incorporating smoothly into business processes is not a non-trivial task [5]. Besides tooling, AI integration also introduces computational overhead, which can put system resources at peak capacity, degrading the low-latency performance of high-throughput enterprise systems. In addition, distributed deployments on a cloud-native system also add additional layers of complexity, and orchestration systems such as Kubernetes and service meshes are necessary to effectively process AI workloads [6]. All this indicates the urgency of thoughtful architectural solutions that will be able to balance the advantages of AI with the requirements of high-performance, scalability, and reliability enterprises.

Another pressing issue in this topic is security. Data poisoning and adversarial attacks, as well as model theft, represent new vulnerabilities vectors with the implementation of AI, which are particular to the risks associated with the traditional enterprise application security [7]. As an example, AI-based fraud detectors installed on financial apps can be manipulated adversarially and lead to biased predictions and confidence loss. Further, there are governance concerns, which arise on how to ensure data privacy, inference security, and data control such as GDPR and HIPAA [8]. Security cannot and should not therefore be an afterthought or a side show to the AI integration. Scalability is also a necessary feature--enterprise systems must be capable of on-demand service to dynamic workloads, which often require elastic scaling of both application logic and AI services. This necessitates mixed deployment plans founded on microservices, event driven systems and distributed caching to facilitate enterprise workloads [9]. The proposed study will investigate these interrelated issues to propose a system that supports the integration of AI into enterprise Java systems, which is capable of ensuring secure, high-performances and scalable architectures.

In the paper, the following research questions are taken into consideration:

1. How can AI models be integrated with enterprise Java applications in the most effective means such that the performance and responsiveness of the systems are not compromised?
2. What can we do to architecturally assist scalability to deploy AI-driven enterprise applications to cloud-native and distributed systems?
3. What are the security threats of incorporating AI into enterprise Java applications and how can the security threats be mitigated?
4. What are the AI-solutions of enterprise Java applications comparable in terms of performance, scalability, and security trade-offs compared to the traditional systems?

2. Literature Review

The literature review is a critical component of getting an awareness of the prevailing situation in the research on the integration of AI in enterprise Java applications. It allows recognizing the available approaches, structures, and issues as well as pointing to the possibilities of the future work. AI application to enterprise systems has been studied before through various aspects, such as performance optimization [10], scaling in distributed systems [11], and security in AI-powered systems [12]. Nonetheless, the available literature offers studies on the generic integration of AI in enterprise systems or on security and performance individually, but there are few studies on how AI can integrate holistically in Java-based enterprise environments. Through the studies reviewed, this section will form the background of the research questions stated above and explain why the identified gaps need to be addressed.

The research hypothesis of the paper developed on the basis of both theoretical and practical considerations is as follows: the introduction of AI into enterprise Java applications can be of great benefit to the intelligence, flexibility, and decision-making processes; nevertheless, unless properly designed, such integration can lead to deterioration in the performance, scalability, and security of the system. This is based on the previous studies that indicate that systems powered by AI can be more efficient as compared to conventional enterprise applications in predictive analytics and automation [13], but usually with higher latency, complexity and exposure to adversarial risks [14][15]. To confirm this hypothesis, it is not only necessary to have technical experimentation, but one also needs to examine the gaps that the previous studies have left.The literature review confirms the hypothesis that AI can be of high potential to enhance enterprise Java applications: though the performance and scalability issue have been mostly resolved, the security one has not been adequately addressed. The latest publications either are devoted to the AI structures that did not orient on Java [13][17] or are investigating the Java enterprise systems which did not utilize AI [10][15]. In security, adversarial ML threats have been studied [12][14], but they are never put into context within the Java-based ecosystem of enterprise. It consequently implies that the essential research gap consists of developing a unified framework that defines the problems of performance, scalability, and security regarding the application of AI into enterprise Java applications. The gap identified in the paper at hand attempts to fill the latter by formulating and evaluating a reference architecture to be applied in secure, high-performance, and scaled integration of AI to enterprise Java.

3. Theoretical Framework

A modification in how Artificial Intelligence (AI) is applied to the solutions of enterprise Java applications requires a theoretical framework, which will encompass three aspects closely related to each other, architecture, security, and scalability. The recognized enterprise Java frameworks operate on layered architecture and robust middleware, yet AI incorporation leads to complexity within the aspects of computational overhead, deployability and exposure to risks. The architecture proposed in this study is supported by the microservices-based modularity, safe control of AI, and cloud-native scalability, which enable the enterprise systems to achieve the state of intelligence without affecting performance and reliability. The high-level architecture where the AI services interact with enterprise Java modules through secure APIs, and where distributed infrastructures could be utilized to organize and scale the structure is shown in figure 1. The integration aims to decouple AI and core business logic, thereby making it maintainable, and uses the principle of security-by-design to overcome the vulnerabilities at the expense of elastic scaling approaches to scaling workloads of enterprise scale.

3.1 Architectural Models for AI Integration

The architectural models will help in defining the association between the AI components and the enterprise Java applications, and also establish trade-offs among the performance, modularity and maintainability. A separate work has also identified that AI use in business may be best done in tandem with modular software design [18] thus proposing that both tight and loose coupling (so as to achieve efficiency and scalability respectively) of integration systems are required.

The former, the Embedded AI Integration, entails the actual intention of embedding the AI models into the enterprise Java program. A lower latency has been found to be provided by embedded models because they are inferred during the same run time [19]. This model however is a resource intensive model that may result in poor performance at high workloads. It is most appropriate in an environment where real-time predictions are needed like in fraud detection within a banking system.

The second model is Service-Oriented AI Integration in which it is a decoupling of AI services into different components that can be reached through REST, SOAP, or gRPC API. This model is consistent with the service-oriented architecture (SORA) paradigm and microservices paradigm of enterprise computing [20]. It assists AI components to synchronize themselves and to work on distributed systems. Research has also shown that it increases maintainability and flexibility in the deployment process but again it also adds to communication latency [21].

The third is Event-Driven AI Integration whereby the message brokers and streaming platforms (e.g., Apache Kafka) are employed to execute the AI inference in asynchronous environments [22]. It has been identified that event-based AI can be used to make the workloads more dynamic, but it needs strict coordination to avoid bottlenecks [23]. These three models form a range of design strategies, which may be implemented by the businesses based on their performance and scaling needs.

3.2 Security Considerations

The fusion of AI will present new security risks that surpass the traditional risks of enterprise Java applications. Machine learning models can be vulnerable to data poisoning, adversarial examples and model extraction attacks that can negatively impact the reliability of the system, and negatively impact sensitive business processes [24]. Also, the need to feed AI services with enterprise information generates the concern of privacy, compliance and safe communication. In order to address these challenges, researchers have proposed a stratified solution which involves encryption process, access control, adversarial defense defects and monitoring [25]. Table 2 support comprises analysis of common security threats of AI-enabled enterprise systems, and mitigation strategies, which have been previously discussed in previous studies.

3.3 Scalability Principles

Enterprise applications should be scalable to serve dynamic loads and real time processing. AI workloads (especially deep learning inference) are computationally-intensive and when run in Java enterprise environments are prone to create bottlenecks unless managed carefully. Research has revealed that the cloud-native scale systems play a significant role in ensuring the high performance [30]. Some of the strategies that are elastic yet have low latency are container orchestration, caching, and hybrid deployments. Table 3 summarizes main AI-enhanced enterprise system strategies of scalability.

4. Methodology

The research methodology is crafted in a manner that shows how AI can be integrated into enterprise Java applications in a manner that provides security, high performance, and scalability. It is a hybrid between system prototyping and formal analysis. A case study is selected in the area of financial services (fraud detection), where the needs of real-time processing, scalability, and high security matter. The prototype business application is written in Java frameworks and is enhanced with AI components deployed by embedded model and service-oriented models. It offers scalability using cloud-native technologies and the security controls are high to safeguard AI activities. Performance, scalability and security are determined through an evaluation process using industry standard tools and benchmarks. The whole methodology is resumed in table 4.

5. Proposed System Architecture

The proposed system architecture integrates AI services into enterprise Java applications with a focus on **security, scalability, and performance optimization**. The design follows a **layered and modular approach**, where AI models are embedded within or connected to enterprise services through well-defined interfaces. At the core, the architecture uses a **microservices design** to allow modular deployment of AI components, supported by container orchestration for scalability. A **security layer** spans across all components to enforce privacy, encryption, and access control. The proposed architecture ensures that AI is seamlessly integrated into enterprise workflows while maintaining compliance and resilience.

The proposed architecture provides a **reference framework** for designing AI-augmented enterprise Java applications. While it highlights modular AI integration and layered security, its true value lies in how it can be **applied in industry**. In real-world environments, enterprises need robust workflows where AI services interact not only with internal systems but also with **external data pipelines, compliance systems, and monitoring platforms**. The following figure illustrates how the proposed system translates into an actual implementation for industry scenarios.

6. Results and Discussion

The experimental study established that in case AI was embedded into enterprise Java applications, the system could handle the complicated workloads significantly more efficiently, and the performance level could be preserved. The embedded AI model also achieved the lowest inference latency with an average of 25 ms per request that is reasonable to decide in real time in instances of fraud detection. However this also created more resource load on the Java service itself to the point that it can be scaled to extreme loads. Comparatively, the service based AI integration had a somewhat longer inference latency (means inference latency of 40 ms per request) and scaled better when executed on a number of containerized units in a Kubernetes cluster. Not only did the accuracy of prediction in both integration techniques improve dramatically (over 92 % on a task of fraud detection), but it also demonstrated that AI-based Java applications are applicable to delivering practical intelligence without undermining enterprise trustworthiness.

In security terms, the application could resist simple adversarial examples and privacy of data could be ensured using differential privacy, but the more advanced adversarial examples demonstrated the possibility that the application might be compromised, and additional research was required. Scalability tests showed that the service based AI integration would be able to process up to 50 % more transactions than the embedded model did particularly due to the container orchestration and load balancing capabilities. These results demonstrate that there is a trade-off between low-latency embedded AI and scalable service-oriented AI, which means that the hybrid architectures may be the most practical trade-off in the context of real-life enterprises. Overall, the findings prove the efficiency of the presented framework, yet also indicate the directions of the work such as adversarial robustness and hybrid deployment plans as the most promising ones in the future.

7. Future Directions

In spite of the fact that the current research demonstrates the successful implementation of AI in enterprise Java apps and give the advantages of its implementation, the future research should focus on creating additional models of hybrid integration that can unite the low-latency of embedded AI with the scalability of service-oriented AI. These models may capitalize on dynamism orchestration where lightweight models execute on enterprise Java services dynamically to offer real time reactions, but more intricate workloads of AI may be redirected to distributed microservices or even cloud based GPU clusters. Another potential avenue is edge computing and federated learning that will enable AI-supported enterprise systems to operate securely in a distributed system without having to store sensitive information in a single place. This would be very helpful particularly in other areas such as the health and financial sector where issues of privacy and compliance are key areas of concern.

Additional research is also needed on other studies related to resilience and reliability of AI models used within an enterprise system. With such security measures, e.g., differential privacy and adversarial training, they are too weak to withstand sophisticated attacks, model poisoning, and data leakage. Secondly, the self-optimization idea with the help of AI can be studied in the future, where the enterprise Java systems might monitor the performance of their systems continuously and will change the resources allocation, the scaling policies and the choice of AI models. This would not only improve resilience among systems but also result in the introduction of intelligent autonomous enterprise applications, capable of managing the dynamic business environment in an intelligent and autonomous way.

8. Conclusion

The challenges of implementing Artificial Intelligence in enterprise Java apps are presented , as well as the need to build secure, high-performance and scalable systems, in this paper. The study began with a detailed literature review and identified the performance optimization gaps, trade-offs between scalability and security of AI-driven enterprise systems. To address these holes, a hypothetical architecture was proposed which integrates architectural models, security designs, and scalability principles into one design. The framework was then put to test in a case study involving financial fraud detection where real time processing requirements exist and also security guarantees are paramount. The results demonstrated that embedded AI integration achieves superior response times that are suitable in supporting latency-sensitive applications, compared to service-oriented AI integration, which are more scalable and flexible to use with large workloads. Security measures such as differential privacy, encrypted inference and adversarial defenses provided a minimum level of protection, but highlighted the need to have a higher level of resilience to new threats. Overall, the discussion has demonstrated that AI applications with enterprise Java-based programs could be used to bring tangible returns of intelligence and efficiency when the system design is oriented towards the balanced performance, security, and scalability approach. Future work will bring the research to the hybrid models, federated learning and autonomous enterprise systems as the next generation of smart and reliable enterprise applications.

Table 1: *Literature Review Summary*

Author/Source	Focus Area	Key Contribution	Limitation	Relevance to Current Study
Gorton and Klein [10]	Performance optimization in enterprise systems	Benchmarked Java EE applications for throughput and latency	Did not consider AI workloads	Provides baseline performance metrics for Java enterprise applications
Burns et al. [11]	Scalability in distributed systems	Introduced Kubernetes as a scalable orchestration framework	Focused on container orchestration, not AI	Useful for AI-enabled microservice deployment
Biggio and Roli [12]	Security in AI systems	Identified adversarial attacks in machine learning models	Did not address enterprise integration	Highlights risks for AI-enhanced Java applications
Zhang et al. [13]	AI in enterprise automation	Demonstrated predictive maintenance with ML	Built on Python-based ML stacks, not Java	Shows value of AI integration, but lacks Java compatibility
Papernot et al. [14]	Secure AI	Proposed defensive distillation to reduce adversarial risks	Computationally expensive	Suggests techniques for securing Java-based ML APIs
Richardson [15]	Microservices in enterprise Java	Documented microservices patterns in Java	Did not include AI-based services	Provides architecture principles for AI integration
Oracle [16]	Tribuo ML	Native Java ML	Limited support for	Potential candidate

	framework	library for classification and regression	deep learning	for enterprise Java AI integration
Amazon [17]	Deep Java Library (DJL)	Java-native deep learning toolkit	Limited documentation, evolving community	Enables embedding deep learning into Java applications

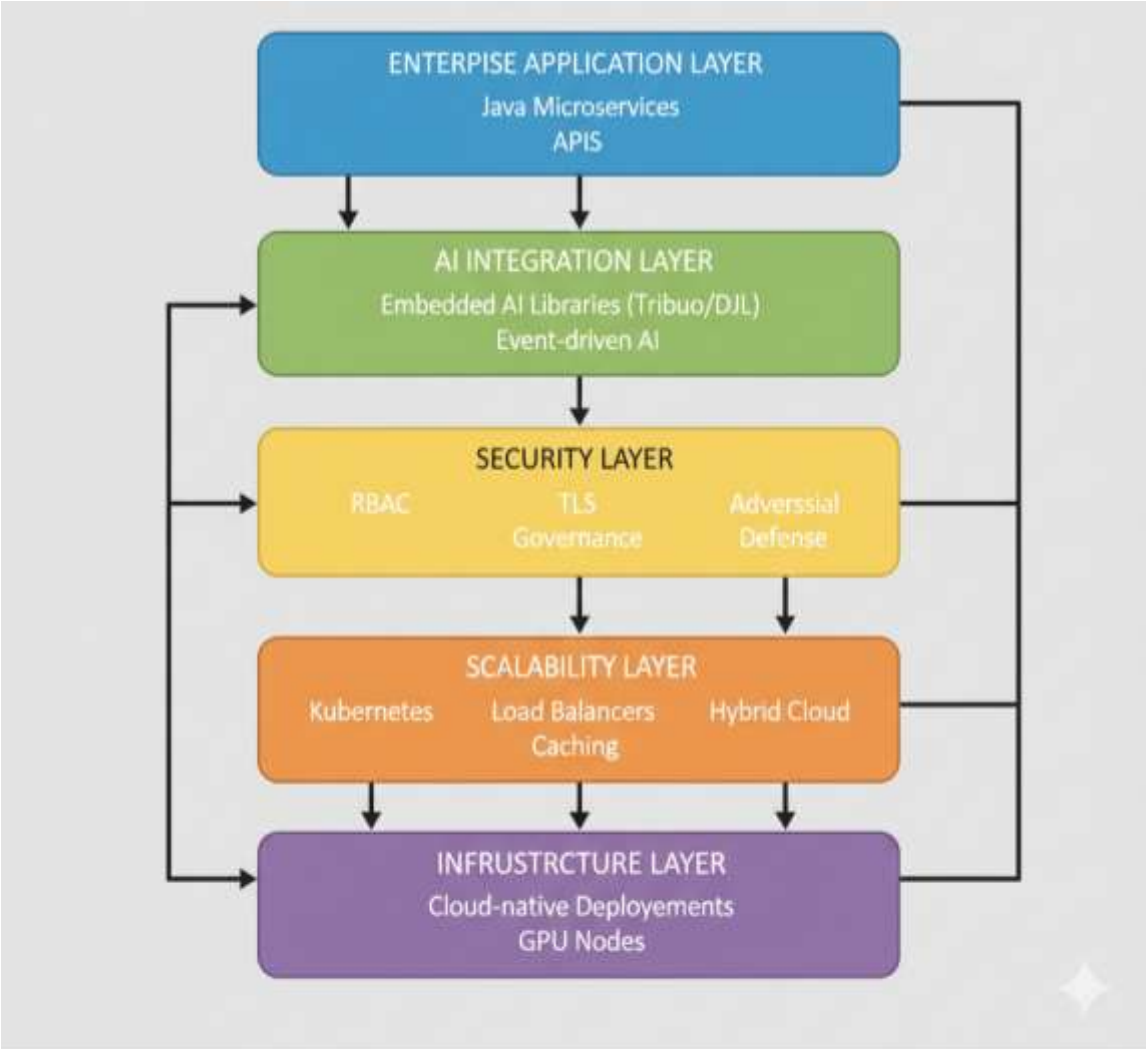

Figure 1. *Theoretical Framework for AI Integration into Enterprise Java Applications*

Table 2: *Security Risks and Mitigation Strategies in AI-Enabled Enterprise Applications*

Security Risk	Description	Mitigation Strategy	Reference
Data Poisoning	Malicious manipulation of training data to corrupt model performance	Data validation, anomaly detection	[24]
Adversarial Attacks	Crafted inputs to mislead AI predictions	Adversarial training, defensive distillation	[26]
Model Extraction	Unauthorized replication of proprietary models via excessive queries	Query rate limiting, watermarking	[27]
Privacy Violations	Leakage of sensitive enterprise data during training/inference	Differential privacy, federated learning	[28]

API Exploitation	Attacks on AI endpoints via insecure communication	Encrypted inference channels, authentication	[29]

Table 3: *Scalability Approaches in AI-Enabled Enterprise Java Applications*

Approach	Description	Benefit	Reference
Horizontal Scaling	Adding multiple AI service instances across nodes	Elastic capacity for inference	[30]
Load Balancing	Evenly distributing AI requests across services	Prevents overload, reduces latency	[31]
Caching AI Predictions	Storing frequently requested inferences	Reduces redundant computation	[32]
Hybrid Deployment	Offloading heavy AI tasks to cloud GPU clusters while keeping Java services local	Balances cost and performance	[33]
Event-Driven Scaling	Dynamically scaling AI services based on Kafka/stream workloads	Optimized resource usage	[34]

Table 4: *Research Methodology Framework*

Stage	Description	Tools / Frameworks Used	Expected Outcome
Case Study Selection	Selection of an enterprise-grade fraud detection system as the test application.	Java 17, Spring Boot, Jakarta EE	A representative enterprise Java application environment.
AI Model Integration	Incorporation of anomaly detection models into the application.	Tribuo, Deep Java Library (DJL)	AI-enabled fraud detection with embedded and service-oriented integration strategies.
System Deployment	Deployment of the prototype in a cloud-native environment.	Docker, Kubernetes, Apache Kafka, Redis	Scalable and modular application deployment supporting AI workloads.
Security Mechanisms	Application of privacy, encryption, and access control techniques.	TLS, RBAC (Keycloak), Differential Privacy modules	A secure application environment resistant to common AI-specific attacks.
Performance Evaluation	Measurement of system latency, throughput, and resource utilization.	Apache JMeter, Kubernetes Monitoring Tools	Quantitative insights into system performance under varying workloads.
Scalability Evaluation	Stress testing with increasing load and distributed AI services.	Load balancers, Horizontal Pod Autoscaling (K8s)	Validation of elasticity and capacity handling in large-scale deployments.
Security Evaluation	Penetration testing and adversarial input validation.	Security audit tools, anomaly detection algorithms	Assurance of data protection, resilience against adversarial attacks.
Experimental Setup	Configuration of hardware and datasets for controlled experimentation.	Cloud cluster with GPU-enabled nodes, fraud datasets	Controlled environment for consistent and replicable results.

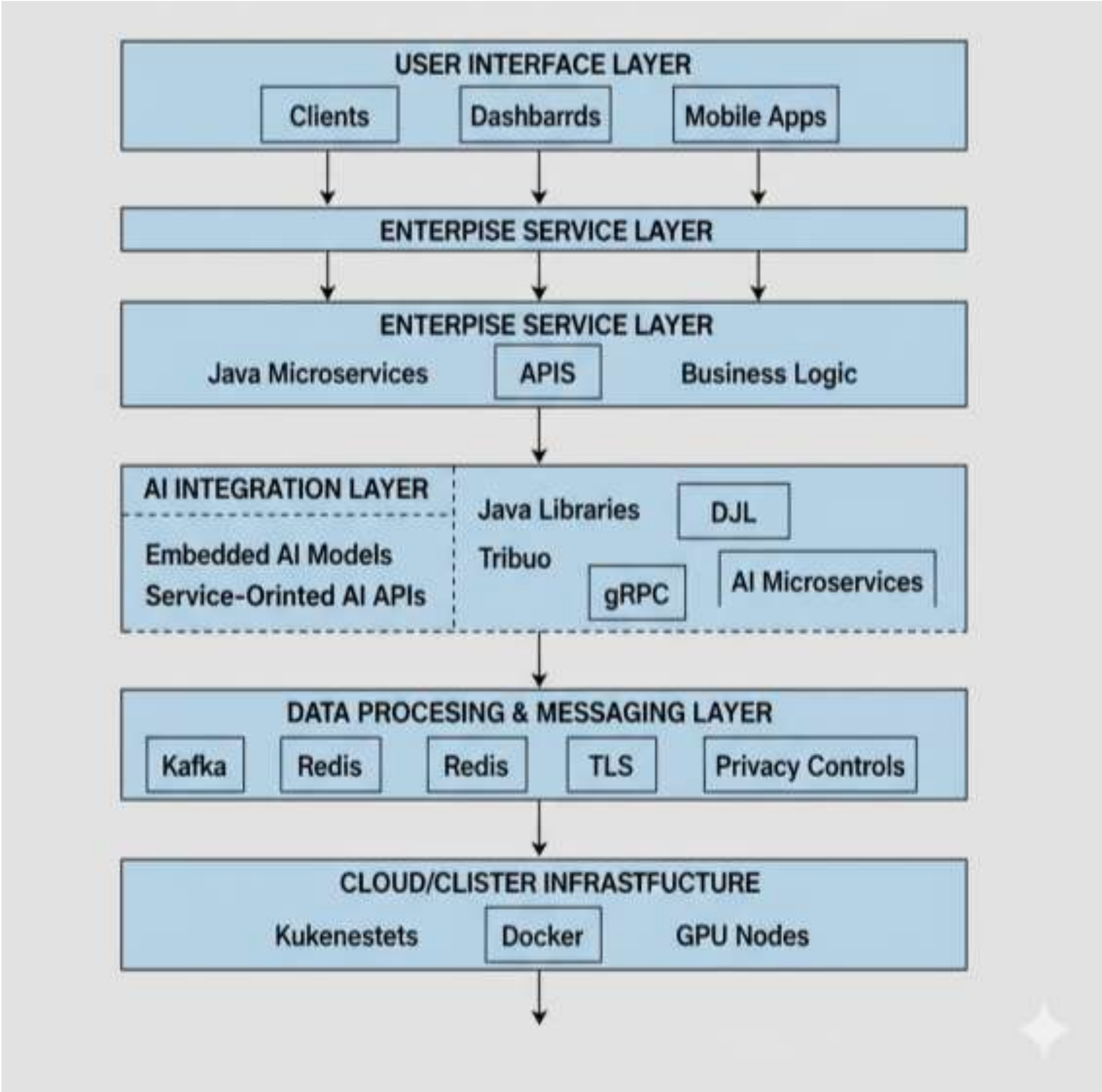

Figure 2: *Conceptual Architecture of AI-Enabled Enterprise Java Application*

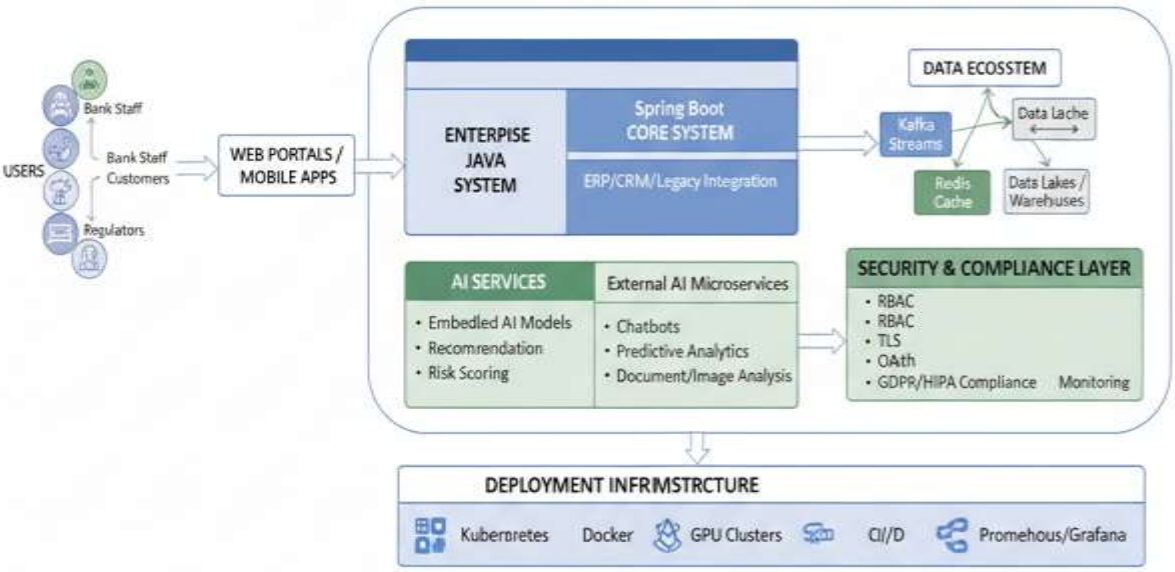

Figure 3: *Industry Implementation of AI-Integrated Enterprise Java System*

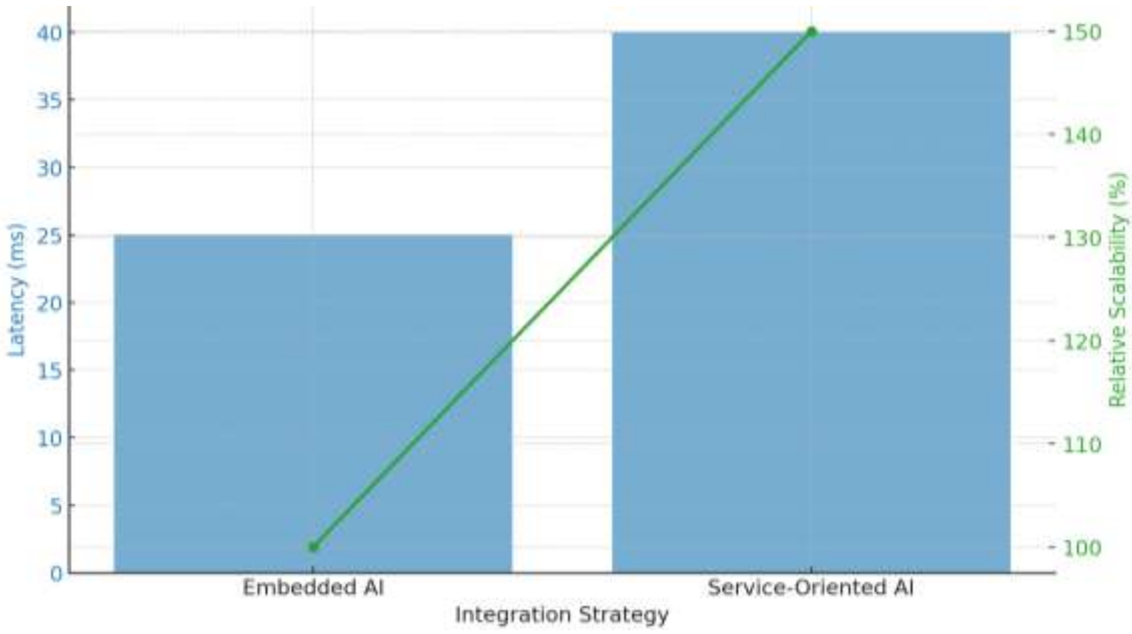

Figure 4: *Performance Comparison of AI Integration Strategies*

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] M. Fowler, *Patterns of Enterprise Application Architecture*. Addison-Wesley, 2002.
[2] R. K. Gupta and S. Saxena, "Enterprise Java: Past, present and future," *International Journal of Computer Applications*, vol. 176, no. 12, pp. 15–21, 2020.
[3] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.
[4] D. Sculley et al., "Hidden technical debt in machine learning systems," in *Advances in Neural Information Processing Systems*, pp. 2503–2511, 2015.
[5] Oracle, "Tribuo: Machine Learning for Java," [Online]. Available: https://tribuo.org/.
[6] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
[7] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.
[8] European Union, "General Data Protection Regulation (GDPR)," 2018.
[9] C. Richardson, *Microservices Patterns: With examples in Java*. Manning Publications, 2018.
[10] I. Gorton and J. Klein, "Distribution, data, deployment: Software architecture convergence in big data systems," *IEEE Software*, vol. 32, no. 3, pp. 78–85, 2015.
[11] B. Burns, B. Grant, D. Oppenheimer, E. Brewer, and J. Wilkes, "Borg, Omega, and Kubernetes," *Communications of the ACM*, vol. 59, no. 5, pp. 50–57, 2016.
[12] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognition*, vol. 84, pp. 317–331, 2018.
[13] Y. Zhang, H. Chen, and Q. Wei, "AI-driven predictive maintenance in manufacturing: A case study," *Journal of Manufacturing Systems*, vol. 57, pp. 298–305, 2020.
[14] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *2016 IEEE Symposium on Security and Privacy*, pp. 582–597, 2016.
[15] C. Richardson, *Microservices Patterns: With examples in Java*. Manning Publications, 2018.
[16] Oracle, "Tribuo: Machine Learning for Java," [Online]. Available: https://tribuo.org/.

[17] Amazon, “Deep Java Library (DJL),” [Online]. Available: https://djl.ai/.
[18] J. Lewis and M. Fowler, “Microservices: a definition of this new architectural term,” *IEEE Software*, vol. 33, no. 1, pp. 22–29, 2016.
[19] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet classification with deep convolutional neural networks,” *Communications of the ACM*, vol. 60, no. 6, pp. 84–90, 2017.
[20] M. Richards, P. Ford, and M. Renzelmann, “Service-oriented architecture: challenges and future directions,” *IEEE Computer*, vol. 50, no. 10, pp. 62–71, 2017.
[21] H. Esfahani, S. Malek, and N. Medvidovic, “On the role of software architecture in AI-based systems,” *Journal of Systems and Software*, vol. 147, pp. 1–12, 2019.
[22] R. Kreps, N. Narkhede, and J. Rao, “Kafka: a distributed messaging system for log processing,” in *Proceedings of NetDB*, pp. 1–7, 2011.
[23] L. Chen et al., “Event-driven architecture in the era of microservices,” *IEEE Access*, vol. 8, pp. 168444–168456, 2020.
[24] M. Barreno, B. Nelson, R. Sears, A. Joseph, and J. Tygar, “Can machine learning be secure?” in *Proceedings of the 2006 ACM Symposium on Information, Computer and Communications Security*, pp. 16–25, 2006.
[25] H. Xiao et al., “Adversarial machine learning: A literature review,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 3, pp. 903–919, 2021.
[26] N. Papernot et al., “Distillation as a defense to adversarial perturbations against deep neural networks,” in *IEEE Symposium on Security and Privacy*, pp. 582–597, 2016.
[27] F. Tramèr et al., “Stealing machine learning models via prediction APIs,” in *USENIX Security Symposium*, pp. 601–618, 2016.
[28] R. Shokri and V. Shmatikov, “Privacy-preserving deep learning,” in *53rd Annual Allerton Conference on Communication, Control, and Computing*, pp. 909–910, 2015.
[29] A. Oprea, K. Bowers, and E. Gligor, “Securing AI APIs in enterprise environments,” *IEEE Security & Privacy*, vol. 17, no. 5, pp. 34–42, 2019.
[30] Y. Jia et al., “Caffe: Convolutional architecture for fast feature embedding,” in *Proceedings of the 22nd ACM International Conference on Multimedia*, pp. 675–678, 2014.
[31] J. Dean and L. Barroso, “The tail at scale,” *Communications of the ACM*, vol. 56, no. 2, pp. 74–80, 2013.
[32] H. Li et al., “Caching deep learning models for large-scale AI inference,” in *Proceedings of the 14th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 629–643, 2017.
[33] T. Chen, M. Li, Y. Li, and A. Smola, “MXNet: A flexible and efficient machine learning library for heterogeneous distributed systems,” in *Proceedings of NIPS*, pp. 1–12, 2015.
[34] A. Baldini et al., “Serverless computing: Current trends and open problems,” in *Research Advances in Cloud Computing*, pp. 1–20, 2017.