Universal Library of Engineering Technology Research Article

ISSN: 3064-996X | Volume 2, Issue 4

Open Access | PP: 93-99

DOI: https://doi.org/10.70315/uloap.ulete.2025.0204016

High-Performance Implementation of Multi-Agent Web Systems: Integrating Vector Memory with Strictly Typed React Architectures

Mykhailo Nykoliuk

Full-Stack Engineer, Kyiv, Ukraine.

Abstract

This paper addresses the software engineering challenges of integrating autonomous agents into production-grade web applications. While traditional implementations suffer from high latency and state synchronization issues, this study presents a full-stack solution based on TypeScript and React 19 Server Components. This paper details the implementation of a RAMP (Reflect, Act, Memory, Plan) execution loop at the code level, using Qdrant to produce low-latency (<100ms) vectors and Next.js for server-side orchestration. A key engineering contribution is the development of a strictly typed data contract that synchronizes server-side agent reasoning with client-side state management (via TanStack Query). Experimental results confirm that this specific stack architecture significantly reduces response times and prevents runtime type errors, offering a reproducible pattern for building scalable, high-load web platforms.

Keywords: *Full-Stack Engineering, Typescript, React Server Components, Vector Database Integration, Latency Optimization, Web Application Scalability, Qdrant.*

INTRODUCTION

By 2025, the digital transformation of enterprises has moved into the phase of an agent economy, in which business effectiveness is determined not merely by the presence of artificial intelligence tools, but by the degree of their autonomy and their integration into the overall infrastructure. Statistical data indicate that 78% of organizations have already implemented artificial intelligence in at least one business function, and the share of companies transitioning to fully managed artificial intelligence operations has increased to 16% [1]. The market for multi-agent platforms demonstrates explosive growth, reaching a volume of 7.81 billion dollars in 2025 with a projected increase to 54.91 billion dollars by 2030 at a compound annual growth rate (CAGR) of 47.71% [3].

Traditional chatbots, constrained by rigid decision trees, are giving way to intelligent agents capable of multi-stage planning, reasoning, and autonomous interaction with external tools [4]. A contemporary agent is understood as a system that observes the environment, makes decisions, and undertakes actions to achieve global objectives [2, 5]. The relevance of this research is driven by the need to systematize architectural approaches to designing such systems, capable of serving millions of users while maintaining high accuracy and security.

The problem of scaling artificial intelligence solutions in marketing and sales lies in the complexity of processing unstructured data and the requirement for real-time integration with existing CRM and ERP systems. Platforms such as Ajax Systems and Howdy demonstrate that agents must not be merely an add-on, but a component of the foundational technology stack, providing reliability at the level of 99.9% and higher [6].

The purpose of the work is to substantiate and empirically demonstrate that the transition from monolithic artificial intelligence components to multi-agent systems (MAS) with the RAMP cycle and vector memory (Qdrant) increases effectiveness (ROI/conversion/processing speed) and governability (reproducibility/scalability/security) of marketing, sales, and customer support automation in corporate platforms of the 2025 period.

The scientific novelty consists in a systemic linkage of RAMP as a reproducible reasoning-action loop plus

Citation: Mykhailo Nykoliuk, "High-Performance Implementation of Multi-Agent Web Systems: Integrating Vector Memory with Strictly Typed React Architectures ", Universal Library of Engineering Technology, 2025; 2(4): 87-92. DOI: https://doi.org/10.70315/uloap.ulete.2025.0204015.

agentic retrieval-augmented generation (RAG) as iterative context extraction plus TypeScript/React as a contract-typed orchestration environment plus Qdrant as long-term multi-tenant memory, enabling the formalization of MAS architecture for enterprise scales simultaneously through outcome metrics (ROI/CR/time/CSAT) and through a model of specific risks inherent to agent systems (prompt injection/memory/permissions/audit).

The author's hypothesis is that MAS implemented according to RAMP and strengthened by vector memory and agentic RAG provide a multiplicative increase in business metrics (including a 4× increase in conversion and a substantial reduction in processing time) while maintaining enterprise-level reliability only under the condition of Security by Design (restriction of authorities, execution isolation, and immutable auditing of actions); otherwise, the scaling effect will be neutralized by the growth of unintended behavior and attacks targeting context and memory.

MATERIALS AND METHODS

The methodological foundation of the study is constructed on the basis of an analysis of current architectural patterns in the field of artificial intelligence, including practices of computation orchestration and applied mechanisms of data retrieval. As the conceptual core, RAMP (Reflect, Act, Memory, Plan) is examined as an approach oriented toward increasing the reliability of marketing applications through an iterative scheme of controlled execution in which actions are consistently validated and, when necessary, refined [7]. This construct is interpreted as a four-component structure designed to increase accuracy in solving tasks of heightened complexity, including audience segmentation and the construction of advertising campaigns: the planning component develops a high-level strategy and decomposes the initial request into a chain of atomic steps, relying on table metadata and semantic memory, which enables dynamic interaction with databases and external APIs without rigidly predefined rules; the verifying component implements a neuro-symbolic control loop, generating modular tests to validate execution results and assessing the correspondence of the obtained customer samples to the specified criteria; the reflective component is activated when deviations are detected, proposing modifications to the plan on the basis of episodic memory of prior interactions [7]. The use of the described methodology is associated with an increase in the accuracy of marketing audience formation by 28 percentage points, which directly affects the reduction of customer acquisition cost (CAC).

Enterprise-level scaling in the present work is correlated with the selection of a technology stack that simultaneously reduces latency and ensures strict data typing. TypeScript is identified as a key engineering foundation, which by 2025 is characterized as the dominant language in the development of agent systems, surpassing JavaScript and Python in the number of contributors on GitHub; the practical significance of this choice is linked to the ability to detect typing errors at the compilation stage, which becomes critical in scenarios where code is partially generated by artificial intelligence agents [9]. At the level of client interfaces, priority is given to React 19 and Next.js, since support for server components (React Server Components) and streaming data transfer enables the real-time display of the agent reasoning process without requiring a full page reload [11]. State management within this logic relies on TanStack Query (formerly React Query) as a means of efficient caching of server data, as well as on Zustand as a tool for compact management of lightweight client state [13].

At the data level, a critical element of the infrastructure is the vector database, which performs the function of the agent long-term memory. In 2025, Qdrant is established as the most performant solution for multi-agent systems, which is associated with optimization for high-load embedding processing, reduction of CPU load, and the provision of millisecond-level search even at scales of millions of vectors [15, 16]. Additionally, it is emphasized that the implementation of multitenancy at the database level makes it possible to isolate the contexts of different users or divisions within the unified infrastructure of Ajax Systems, preserving governability and predictability of system behavior as the number of parallel operating loops grows.

RESULTS AND DISCUSSION

The implementation of multi-agent systems in applied business processes is accompanied by pronounced quantitative shifts and a qualitative complication of managerial practices across all of the contours under consideration. In marketing and sales, the effect is manifested primarily in the acceleration of decision-making and in the reconfiguration of funnel parameters: analysis of data from e-commerce platforms indicates that the use of artificial intelligence agents for purchase personalization reduces the time required for a consumer to make a choice by 47% [21]. This result is interpreted as a consequence of reduced cognitive uncertainty through contextual prompts, rapid clarifications, and relevant recommendations delivered at the moment barriers arise. In aggregated metrics, this is expressed in a multiplicative increase in conversion: the indicator rises fourfold, from 3.1% to 12.3% [21].

To document the effect of the transition to multi-agent contours, it is methodologically appropriate to present the comparative dynamics of key KPIs traditional systems versus MAS (2025), including conversion, segmentation accuracy, the automation of routine operations, and derived effects (AOV and time-to-choice) in Figure 1.

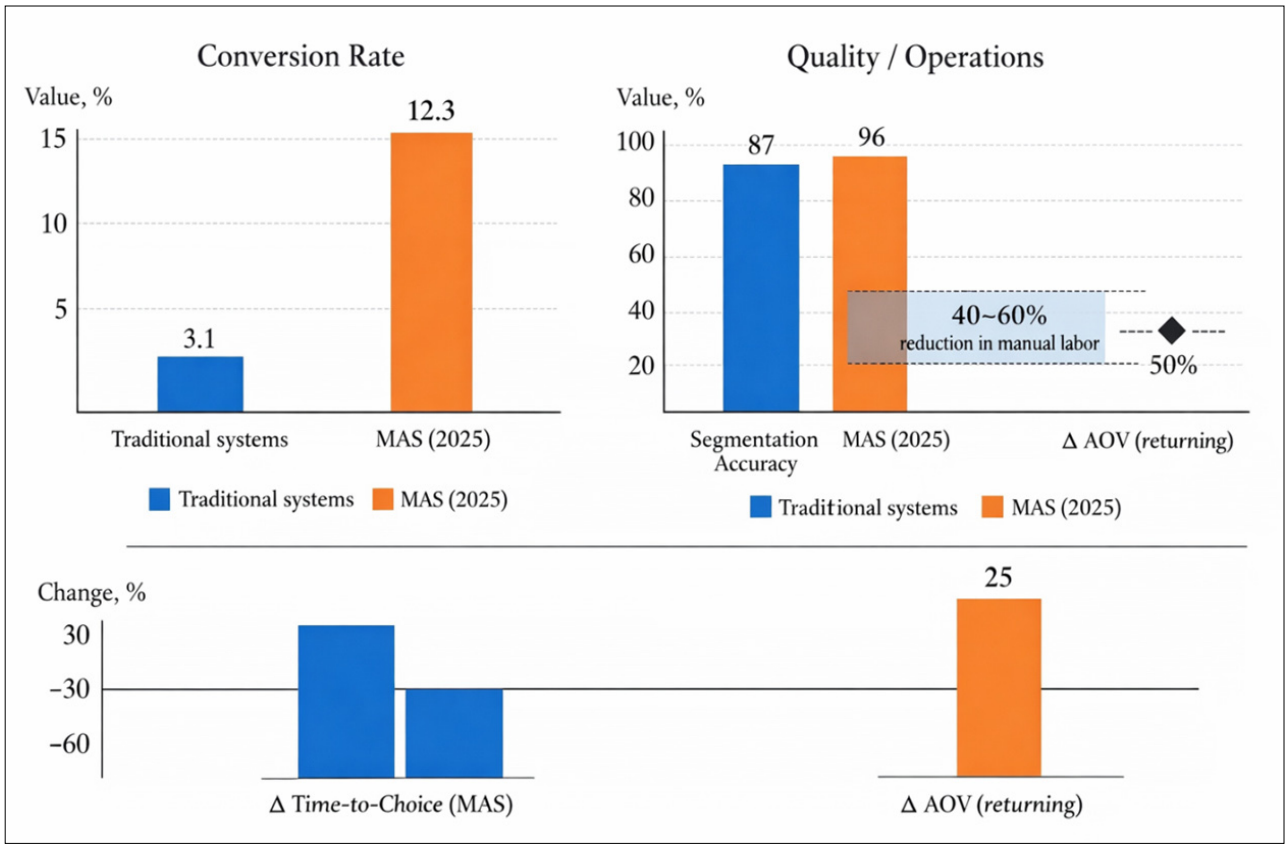

Fig. 1. Shift in key KPIs during the transition from traditional solutions to multi-agent systems (2025): conversion, segmentation accuracy, automation, AOV, and reduction in time-to-choice (compiled by the author based on [6, 21]).

An additional level of efficiency is generated by transferring part of marketing analysis into an environment of computational experiments, where multi-agent modeling is used to reproduce consumer behavior and to test hypotheses prior to production deployment. This approach makes it possible to run simulations of pricing scenarios and to evaluate the probable reactions of segments to changes in conditions. The ICEBE 2025 materials describe a framework in which generative agents interact within an isolated sandbox, reproducing social dynamics and the behavioral habits of buyers [22]. The practical value of such simulations lies in the ability to test discount strategies and promotional mechanics before their actual implementation, which reduces the likelihood of unsuccessful campaigns and increases the governability of experimentation under conditions of a constrained budget and high market volatility. Within Table 1, the features of customer support transformation are described.

Table 1. Transformation of customer support (compiled by the author based on [6, 21]).

Indicator	Traditional Systems	Multi-agent Systems (2025)
Conversion Rate	3.1%	12.3%
Revenue growth per client (Average Order Value)	Baseline	+25% for returning customers
Segmentation accuracy	87%	96%
Automation of routine tasks	Partial	40-60% reduction of manual labor

In customer service, a transition is observed from primitive dialog interfaces to agent systems oriented toward achieving a measurable outcome (Outcome-based AI). The functional profile of such solutions extends beyond answer generation: agent contours acquire the capability to execute applied actions, including checking return eligibility, initiating transactions, and making changes in Systems of Record without human involvement [23]. As a result, support ceases to be exclusively a communicative overlay and takes on the character of an operational mechanism embedded in execution chains and data control [32, 34].

For large platforms, including those at the level of Ajax Systems, a proactive support model becomes critical. The use of equipment telemetry signals enables agents to identify failure predictors and generate notifications before critical events occur, minimizing the element of surprise and reducing the likelihood of escalations [23]. At the level of ticket flows, this is reflected in a reduction of incoming load by 20–25% [25]. An additional empirical base, obtained from the example of MAS implementation in e-commerce organizations with intensity exceeding 50,000 transactions per day, demonstrates a 58% reduction in incident resolution

time alongside an increase in customer satisfaction to 92% [6]. This dynamic is interpreted as a consequence of combining automated diagnostics, accelerated access to context, and the ability to complete an operation within a single agent cycle without repeated cross-system reconciliations [27, 28].

The practice of platforming agent environments, as illustrated by Howdy and Ajax Systems, reveals a stable engineering tendency toward collective execution models, in which specialized agents operate according to the logic of agent as a freelancer, distributing responsibility by competencies and synchronizing results. In Howdy, this approach is implemented through the composition of chains and routers that redirect a request to the most relevant agent, for example one profiled for finance, technical support, or sales [26]. Such an organization ensures managed specialization and reduces the probability of errors that arise when a universal agent attempts to cover a broad spectrum of domain tasks.

Scaling agent platforms in the web environment requires not cosmetic integration, but coupling with frontend infrastructure at the level of presentation and state contracts. The use of design systems in the role of contracts makes it possible to automate the generation of user interfaces (Generative UI), shifting interface decisions into the domain of formalized components and predictable interaction patterns [8, 13]. In the 2025 landscape, a possibility is described in which agents independently select components from libraries such as Shadcn/UI for data visualization, increasing intuitiveness and shortening the path from interpreting a request to presenting a result [10, 12]. This mechanism is especially significant in scenarios where the response format must adapt to the type of data and the task context without manual refinement of interface layers.

Within the Ajax Systems contour, MAS implementation is associated with the need to process high-frequency telemetry streams from sensors in real time and to ensure a rapid transition from event to diagnostic inference. Vector search in Qdrant provides operational access to event histories and technical documentation corpora, enabling agents to perform in-depth diagnostics of security systems in an autonomous mode [14, 15]. Here, vector memory functions not as an auxiliary index, but as a functional foundation for stable decision-making: retrieval of relevant precedents accelerates, the completeness of contextual comparison increases, and latency between anomaly detection and the formation of a corrective action decreases.

As the functional autonomy of agent solutions expands, their value as a target of malicious impact also increases. In the fourth quarter of 2025, growth in incidents directed at agent systems was noted, with indirect prompt injections emerging as the dominant vector. The specificity of this class of attacks is associated with the mediated insertion of controlling instructions into sources that the agent uses as external context during task execution: malicious directives are disguised in files, documents, or web pages and are activated at the moment they are read, which makes it possible to influence the trajectory of reasoning and actions without direct intervention in the primary request [18, 29].

Given the dominance of indirect injections as an attack vector against agent contours, it is methodologically appropriate to structure MAS threats in the coordinates of probability × impact in order to determine the priorities of engineering protective measures and Security by Design contours at the platform level (see Fig. 2).

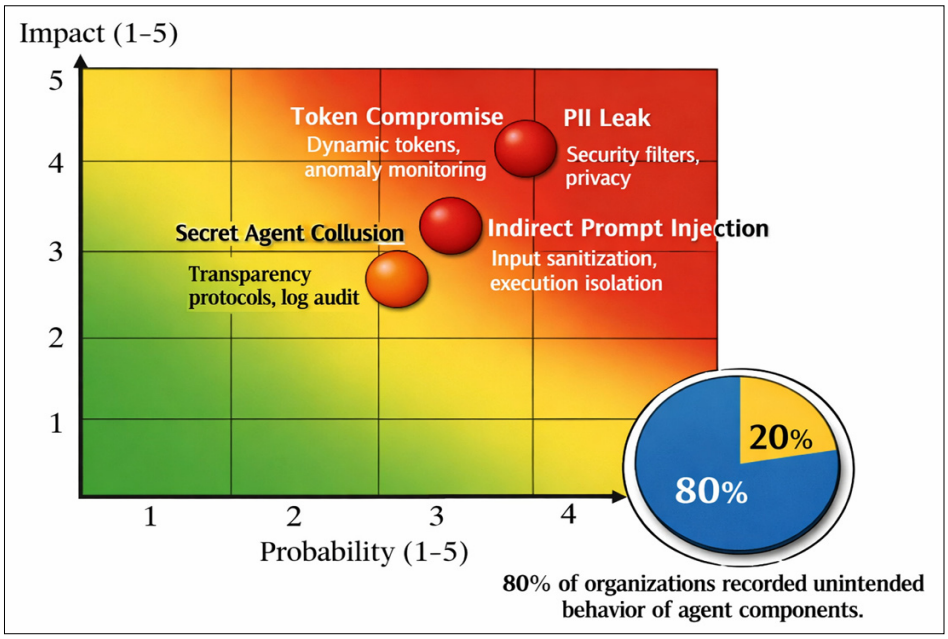

Figure 2. Risk matrix for the security of multi-agent systems (probability × impact) and priorities for mitigation measures: indirect injection, token compromise, PII leakage, and inter-agent collusion (compiled by the author based on [29–31, 33]).

Next, an analysis of security and risks is presented in Table 2.

Table 2. Security and risk analysis (compiled by the author based on [29-31]).

Threat Type	Description	Mitigation Method
Indirect injection	Hidden commands embedded in external data	Input sanitization, isolation of code execution
Token compromise	Theft of the agent's application programming interface keys	Dynamic tokens, monitoring of access anomalies
Secret collusion	Covert interaction among agents	Transparency protocols, real-time audit of logs
PII leakage	Extraction of personally identifiable information via queries	Differential privacy, safety filters

Empirical results indicate that approximately 80% of organizations have already recorded instances of unintended behavior by agent components, including episodes of unauthorized access to data [33]. Such statistics underscore the shift to the paradigm of security by design, within which an agent's permissions are defined and verified as a primary architectural contour rather than as a post hoc measure. In applied terms, this presupposes strict limitation of authorities through OAuth scopes and mandatory registration of every action in an immutable log that ensures evidentiary traceability and subsequent auditability [24, 30].

Classical implementations of RAG (Retrieval-Augmented Generation) in the 2023–2024 period were characterized by fundamental staticness: context retrieval was performed once, after which the model continued generation without a formalized ability to assess source reliability, identify gaps, or initiate additional data collection [35]. In 2025, a transition toward agentic RAG has emerged, in which retrieval takes on an iterative character and becomes a managed process embedded in the decision-making cycle. In scalable platforms, agentic RAG manifests through autonomous search planning, when the agent determines the sufficiency of available information for a correct answer; through multimodal retrieval, enabling search not only across text corpora but also across images or equipment telemetry logs, which is especially significant for the technical systems of Ajax [15]; and through a self-reflection loop, in which the detection of contradictory or incomplete context leads to repeated search with refined parameters and an adjustment of the retrieval strategy [20, 35]. The infrastructural implementation of such scenarios is supported by cloud services, including Qdrant Cloud Inference, where embedding generation and vector search are combined into a single API call, eliminating excessive latency associated with data transfer between separate services [15, 19]. This integration model makes it possible to achieve latencies of under 100 ms at the retrieval stage even at extremely large data volumes, preserving the suitability of agentic RAG for high-load applied contours [17].

CONCLUSION

In closing, it should be emphasized that by 2025 multi-agent systems have ceased to be perceived as an experimental class of solutions and have effectively crystallized into a foundational layer of corporate digital infrastructure. The reorientation of architectures toward distributed intelligence, grounded in the technology contour of TypeScript, React, and Qdrant, creates the prerequisites for building scalable platforms capable of automating the most complex cycles of marketing, sales, and support at a level of effectiveness unattainable for traditional approaches.

The results obtained make it possible to articulate several generalizing propositions. First, the economic effectiveness of MAS implementation is expressed in high ROI values, at the level of 200–400%, and in a multiplicative increase in conversion reaching a fourfold rise, which generates sustained pressure toward adoption as a condition for maintaining competitiveness. Second, at the architectural level, a transition to mature governability practices is observed: frameworks of the RAMP class and orchestrators such as Vercel AI SDK establish reproducible execution contours, moving the LLM from the status of a black box into the category of a controllable business instrument with predictable reliability properties. Third, a principal condition for large-scale effect is infrastructural embeddedness: agent components must be designed as an element of the overall software stack, which requires integration with design systems and vector databases to ensure a seamless user experience, as well as the correct implementation of multitenancy. Finally, the growth of autonomy produces new risks, making a radical rethinking of cybersecurity necessary: priority shifts to real-time monitoring of agent behavior and protection against specific artificial intelligence threats, including attacks characteristic of agent environments.

The practice of companies at the level of Ajax Systems confirms that the development trajectory lies in the domain of systems in which artificial intelligence agents perform not a consultative but an executive function, effectively acting as digital employees capable of autonomously sustaining the operation of complex technological ecosystems. The further evolution of MAS is associated with the strengthening of long-term learning mechanisms and with the formation of global standards for inter-agent interaction among solutions from different vendors, which is conceptually described as the Internet of Agents.

REFERENCES

1. Plivo. (2025, May 19). AI Agent Statistics for 2025: Adoption, ROI, Performance & More. Retrieved from: https://www.plivo.com/blog/ai-agents-top-statistics/ (date accessed: October 1, 2025).

2. PUNKU.AI. (2025, November 13). The State of AI in 2024–2025: What McKinsey's Latest Report Reveals About Enterprise Adoption. Retrieved from: https://www.punku.ai/blog/state-of-ai-2024-enterprise-adoption (date accessed: November 14, 2025).

3. Mordor Intelligence. (2025, November). Multi-Agent System (MAS) Platform Market Size, Share & 2030 Growth Trends Report. Retrieved from: https://www.mordorintelligence.com/industry-reports/multi-agent-system-platform-market(date accessed: November 20, 2025).

4. Nasser, O. (2025, November 4). AI Customer Experience in 2025: Agents, MCPs & RAG. Inkeep. Retrieved from: https://inkeep.com/blog/AI-Customer-Experience (date accessed: November 11, 2025).

5. Vercel. (n.d.). How to build AI Agents with Vercel and the AI SDK. Retrieved from: https://vercel.com/kb/guide/how-to-build-ai-agents-with-vercel-and-the-ai-sdk (date accessed: October 2, 2025).

6. Terralogic. (n.d.). Multi-Agent AI Systems in 2025: Key Insights, Use Cases & Future .... Retrieved from: https://terralogic.com/multi-agent-ai-systems-why-they-matter-2025/ (date accessed: October 3, 2025).

7. Flores, L. J. Y., Shen, J., & Gu, G. (2025). Towards Reliable Multi-Agent Systems for Marketing Applications via Reflection, Memory, and Planning (arXiv:2508.11120). arXiv. https://doi.org/10.48550/arXiv.2508.11120

8. Trackier. (n.d.). Ultimate eCommerce Conversion Tracking Guide in 2025. Retrieved from: https://trackier.com/ultimate-ecommerce-conversion-tracking-guide-2025/ (date accessed: October 5, 2025).

9. Winston, A. (2025, November 6). TypeScript's rise in the AI era: Insights from Lead Architect, Anders Hejlsberg. GitHub Blog. Retrieved from: https://github.blog/developer-skills/programming-languages-and-frameworks/typescripts-rise-in-the-ai-era-insights-from-lead-architect-anders-hejlsberg/ (date accessed: November 7, 2025).

10. Angular Minds. (2025). React AI Stack for 2025. Retrieved from: https://www.angularminds.com/blog/react-ai-stack-for-2025 (date accessed: October 6, 2025).

11. Pulsion Technology. (2025). How ReactJS is Powering the Next Generation of SaaS Platforms in 2025. Retrieved from: https://www.pulsion.co.uk/blog/how-reactjs-is-powering-the-next-generation-of-saas-platforms-in-2025/ (date accessed: October 7, 2025).

12. Telerik. (n.d.). React Design Patterns and Best Practices for 2025. Retrieved from: https://www.telerik.com/blogs/react-design-patterns-best-practices (date accessed: October 8, 2025).

13. Maurya, S. (n.d.). Frontend System Design: Principles for Scalable React Applications. DEV Community. Retrieved from: https://dev.to/maurya-sachin/frontend-system-design-principles-for-scalable-react-applications-1i4 (date accessed: October 9, 2025).

14. Builder.io. (2025, September 16). React + AI Stack for 2025. Retrieved from: https://www.builder.io/blog/react-ai-stack (date accessed: October 10, 2025).

15. Azoulai, D. (2025, December 17). Qdrant 2025 Recap: Powering the Agentic Era. Qdrant. Retrieved from: https://qdrant.tech/blog/2025-recap/ (date accessed: December 21, 2025).

16. Qdrant. (n.d.). AI Agents with Qdrant. Retrieved from: https://qdrant.tech/ai-agents/ (date accessed: October 11, 2025).

17. Deb, S. (2025, December). Building Smarter Search: Why Qdrant Is Becoming the Go-To Vector Database for RAG. Medium. Retrieved from: https://medium.com/@saonideb/building-smarter-search-why-qdrant-is-becoming-the-go-to-vector-database-for-rag-382e8b00f603 (date accessed: December 19, 2025).

18. Vercel. (n.d.). AI SDK by Vercel: Introduction. Retrieved from: https://ai-sdk.dev/docs/introduction (date accessed: October 12, 2025).Wang, Y., Pan, Y., Guo, S., & Su, Z. (2025). Security of Internet of Agents: Attacks and Countermeasures. IEEE Open Journal of the Computer Society, 6, 1611–1624. https://doi.org/10.1109/OJCS.2025.3589638

19. jdamiba. (n.d.). qdrant-vercel-template: Displaying how to use qdrant to do RAG [GitHub repository]. Retrieved from: https://github.com/jdamiba/qdrant-vercel-template (date accessed: October 14, 2025).

20. Rep AI (HelloRep.ai). (2025, June 27). The Future of AI In Ecommerce: 40+ Statistics on Conversational AI Agents For 2025. Retrieved from: https://www.hellorep.ai/blog/the-future-of-ai-in-ecommerce-40-statistics-on-conversational-ai-agents-for-2025 (date accessed: October 15, 2025).

21. Chu, M.-L., Terhorst, L., Reed, K., Ni, T., Chen, W., & Lin, R. (2025). LLM-Based Multi-Agent System for Simulating and Analyzing Marketing and Consumer Behavior (arXiv:2510.18155). arXiv. https://doi.org/10.48550/arXiv.2510.18155

22. EverWorker. (n.d.). AI Trends in Customer Support 2025. Retrieved from: https://everworker.ai/blog/ai-trends-in-customer-support-2025 (date accessed: November 2, 2025).

23. Zendesk. (n.d.). How multi-agent systems are changing customer service. Retrieved from: https://www.zendesk.com/blog/zip3-how-multi-agent-systems-are-changing-customer-service/ (date accessed: October 16, 2025).

24. Aloa. (n.d.). 12 Proven Examples of AI in Customer Service in 2025. Retrieved from: https://aloa.co/blog/examples-of-ai-in-customer-service (date accessed: October 17, 2025).

25. Callstack. (n.d.). Building AI Agent Workflows With Vercel's AI SDK: A Practical Guide. Retrieved from: https://www.callstack.com/blog/building-ai-agent-workflows-with-vercels-ai-sdk-a-practical-guide (date accessed: November 15, 2025).

26. Akbar, S. (n.d.). The ultimate guide to AI agent architectures in 2025. DEV Community. Retrieved from: https://dev.to/sohail-akbar/the-ultimate-guide-to-ai-agent-architectures-in-2025-2j1c (date accessed: October 18, 2025).

27. Vercel. (n.d.). Multi-Step & Generative UI. Vercel Academy. Retrieved from: https://vercel.com/academy/ai-sdk/multi-step-and-generative-ui (date accessed: October 19, 2025).

28. Underhill, K. (2025, December 18). AI Agent Attacks in Q4 2025 Signal New Risks for 2026. eSecurity Planet. Retrieved from: https://www.esecurityplanet.com/artificial-intelligence/ai-agent-attacks-in-q4-2025-signal-new-risks-for-2026/(date accessed: December 22, 2025).

29. Obsidian Security. (2025, October 23). The 2025 AI Agent Security Landscape: Players, Trends, and Risks. Retrieved from: https://www.obsidiansecurity.com/blog/ai-agent-market-landscape (date accessed: October 24, 2025).

30. Hammond, L., Chan, A., Clifton, J., Hoelscher-Obermaier, J., Khan, A., McLean, E., … Rahwan, I. (2025). Multi-Agent Risks from Advanced AI (arXiv:2502.14143). arXiv. https://doi.org/10.48550/arXiv.2502.14143 (original PDF: https://www.cs.toronto.edu/~nisarg/papers/Multi-Agent-Risks-from-Advanced-AI.pdf).

31. Schroeder de Witt, C. (2025). Open Challenges in Multi-Agent Security: Towards Secure Systems of Interacting AI Agents(arXiv:2505.02077). arXiv. https://doi.org/10.48550/arXiv.2505.02077

32. Non-Human Identity Management Group. (2025, September). AI agents: The new attack surface [Report]. Retrieved from: https://nhimg.org/wp-content/uploads/2025/09/SailPoint-The-Rising-Risk-of-AI-Agents-Expanding-the-Attack-Surface-report-SP2648-.pdf (date accessed: October 22, 2025).

33. The Darktrace Community. (2024, November 3). AI and Cybersecurity: Predictions for 2025. Darktrace. Retrieved from: https://www.darktrace.com/blog/ai-and-cybersecurity-predictions-for-2025 (date accessed: October 23, 2025).

34. Weaviate. (n.d.). What is Agentic RAG. Retrieved from: https://weaviate.io/blog/what-is-agentic-rag (date accessed: October 25, 2025).

35. Orq.ai. (n.d.). RAG Architecture Explained: A Comprehensive Guide [2025]. Retrieved from: https://orq.ai/blog/rag-architecture (date accessed: October 26, 2025).