International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Adaptive API Support: A Human-in-the-Loop Agentic RAG Framework for Enterprise Financial Systems

Vishal Shah

Abstract: The world of enterprise developer support in the context of financial technology is characterized by a critical situation where the artificial intelligence usage rate is increasing, both by speeding up response generation and, at the same time, creating a lack of trust that requires verification by a person before production is put into place. The Agentic Knowledge Orchestrator resolves this paradox by offering a Retrieval-Augmented Generation framework, which conceptually incorporates domain expert validation as an inherent part and not as an exception-handling system. The system enhances AI output by integrating Human-in-the-Loop approval gates directly into enterprise collaboration systems, which convert unverified AI outputs into knowledge artifacts that are validated by subject matter experts before being shared with developer communities. The framework itself works on two-feedback learning strategies that detect expert corrections in validation stages and track patterns of end-user acceptance after the deployment, and will refine the knowledge base on the basis of the authoritative domain knowledge and practical utility indicators. Architectural constraints based on empirical foundations, based on benchmark evaluations, guide reranking strategies, response synthesis budgets that are optimized to financial services settings where accuracy, auditability, and compliance governance are the key design parameters. The architecture seals the recorded divide between recent adoption of AI tools and an ongoing lack of trust in the automated systems by developers by ensuring that automated support systems are tied to human expertise, and yet provide the scalability required of a global enterprise. Such a symbiotic combination of machine smarts and human intuition creates a precedent that can be repeated in areas that are knowledge-intensive and in which error is unavailable, but scale necessitates automation, especially in financial technology ecosystems in which API advice has a role in shaping security-critical deployments, regulatory and compliance adherence, and transaction-processing integrity.

Keywords: *Retrieval-Augmented Generation, Human-in-the-Loop Validation, Enterprise Collaboration Platforms, Financial Technology APIs, Dual-Feedback Learning*

1. Introduction

The intersection of artificial intelligence and enterprise collaboration platforms has radically changed developer support workflows, especially in the financial technology ecosystem, in which accuracy, compliance, and real-time validation are uncompromising demands. Enterprise communication tools have achieved new levels of scale, with one of the largest platforms having more than 320 million monthly active users in 2023, setting itself as a major surface upon which developer support experiences can be conducted, necessitating a reference to context and immediate interaction [1]. This massive adoption is symptomatic of a larger trend in which technical problem-solving is more and more done under integrated collaboration environments, as opposed to being done in isolated documentation repositories.

At the same time, AI-assisted tooling has been adopted by the developer community at an impressive pace. According to recent survey information, 76% of developers are currently using or intend to use AI tools, and 62% of developers have already adopted AI tools in their development process, which is an indication that AI-generated responses have become the first line of contact in solving technical problems [2]. The popularity of this application poses threats and opportunities to financial technology firms, in which API interactions may contain sensitive transactional logic, regulatory compliance needs, and security-sensitive operations. The convergence of universal AI usage and critical financial systems makes it necessary to create a framework that will utilize the scalability of AI but preserve the level of epistemic

Independent Researcher, USA

integrity required by financial services. This study proposes a Human-in-the-Loop Agentic Retrieval-Augmented Generation model called Adaptive API Support that is explicitly created to make sure that AI-driven guidance is not disseminated until it passes through the domain expert review, which will produce a knowledge ecosystem that may be relied on when developing financial technologies.

2. The Persistent Challenge of Documentation Accessibility in Financial Technology Environments

Although more advanced documentation and AI-driven search tools exist, developers still face a lot of resistance in finding solid technical advice, and this issue is specifically severe in fields of financial technology, where API specifications have to strike a balance between functional clarity and safety and legal aspects. Evidence provided by surveys shows that more than 61 percent of developers take over 30 minutes a day to search for solutions, and 90 percent look for documentation by using official API or SDK documentation, but again the overhead of search is extremely high, as it is fundamentally a mismatch between the static form of documentation structure and the dynamic nature of the implementation situation [3]. This time cost is a high amount of productivity loss between the enterprise development teams, which equates to topics of feature delivery and operational costs.

Response times have improved with the introduction of AI-power your code and documentation synthesis, but without addressing the inherent lack of trust that mires automated systems. According to the latest industry research, developers are putting 24 percent of their workweek into maintenance and debugging; this number is inclusive of the legacy system remediation and verification of newly-generated code [1]. Crucially, the report also finds that 96% of developers do not fully trust AI-generated code, meaning AI outputs frequently require manual validation before being used in production [1]. Such a gap of trust is especially important in financial technology, where false API calls can start regulatory breaches, create security risks, or result in inconsistencies in transactions with substantive financial consequences.

The verification overload in the financial services context is more pronounced in practices that are often orchestrated with APIs that contain complex business logic that involves payment processing, fraud detection, regulatory reporting, and cross-border transaction processing. One incorrectly understood parameter in a payment initiation API may lead to wrong currency conversion, unsuccessful compliance verification, or incorrect funds transfer. Conventional documentation methods fail to reflect those fine-grained decision trees that experienced engineers develop through cumulative exposure to cases of production incidents and edge cases [4]. In addition, financial APIs change over time as regulatory frameworks, security patches, and business needs evolve, and introduce time delays in which even recently generated AI responses will refer to obsolete patterns or new constraints added.

The issue is not limited to the productivity of the individual developer but extends to organizational knowledge management. Financial institutions also have broad API portfolios that use internal microservices, partner integration, and external developer ecosystems. Every API embodies the cumulative domain insight regarding financial instruments, regulatory requirements, and business processes that are difficult to capture by fastening documentation. In cases where developers query AI systems that are trained over generic repositories, the reply is usually not in context to the institution itself, with reference to risk management policies, internal compliance standards, or architectural patterns that control production deployments. Such a contextual gap requires an architecture of support to directly bridge the gap between automated response generation and human experience so that the knowledge of the financial domain can inform AI results before affecting the choices that developers make with fiduciary and regulatory intent.

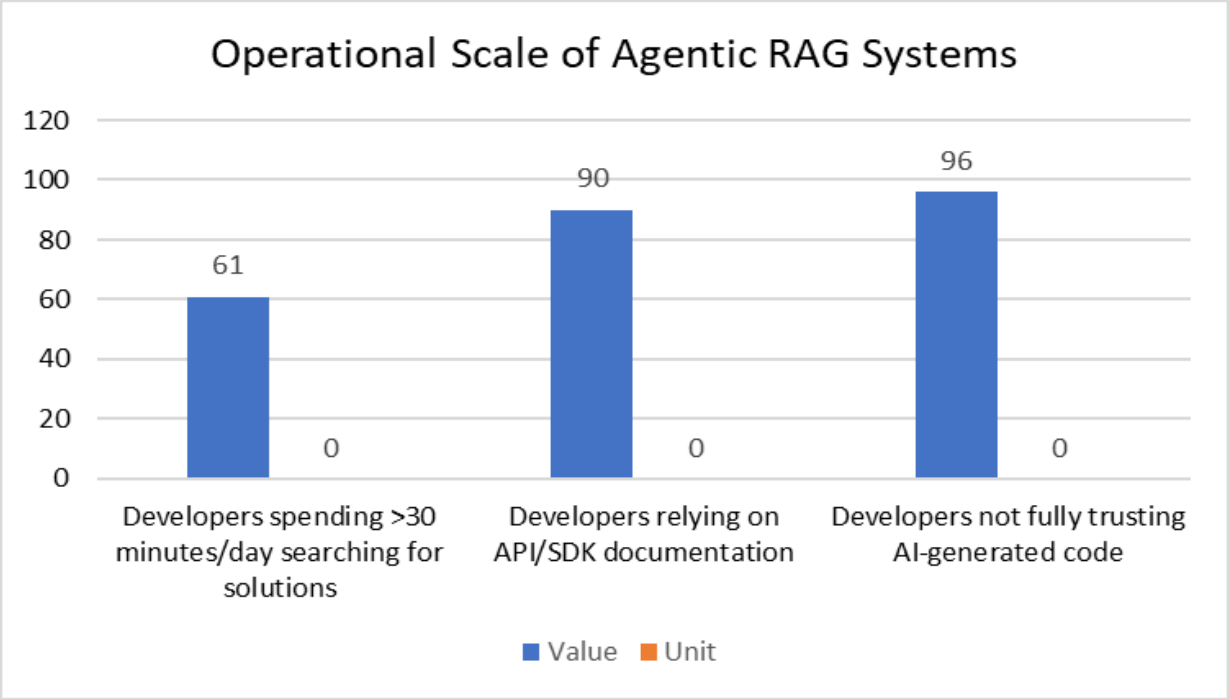

Fig. 1. Percentage of developers affected by documentation search overhead and lack of trust in AI-generated code [3, 4]

3. Architectural Foundations of the Agentic Knowledge Orchestrator for Financial API Support

The system architecture is based on an Agentic Knowledge Orchestrator that links together various autonomous agents as part of a Retrieval-Augmented Generation pipeline tailored to the requirements of the financial technology space, where accuracy, auditability, and compliance governance are the key design constraints. The planner takes care of the entire life cycle during which query interpretation is done, candidate solution synthesis, expert validation, and learning, which is a continuous process based on performance metrics in the field. This architectural solution tries to overcome the inherent weakness of purely generative systems that generate much believable but factually inaccurate information, which is itself a failure mode, especially in financial scenarios where wrong advice on authentication procedures, transaction authorization code, or privacy configurations can pose endogenous risks.

The retrieval-augmented strategy has been empirically tested to be effective in the factual grounding of knowledge-intensive tasks. In the evaluation of RAG-based models, baseline testing showed Exact Match scores of 44.5 on Natural Questions and 52.2 on CuratedTrec benchmarks, which were higher than some parametric baseline models in the evaluation systems [5]. Such findings suggest that generation based on evidence retrieved is more accurate in facts than generation based on parametric knowledge of the model alone, and is directly applicable to the aspects of financial API support where the responses should be based on the up-to-date documentation, latest changes in policy, and organization-specific implementation needs, instead of general programming patterns.

This retrieval subsystem by the orchestrator should be limited to practical considerations based on the large-scale appraisal programs. More recent benchmark work has tested RAG systems on corpora with 10,960,554 webpages divided into 113,520,750 text segments, and retrieval operations are limited to the top 20 passages and generation results to 400 words to trade off latency, computational cost, and comprehensiveness of the results [6]. These numerical parameters directly influence the architectural choices of the orchestrator, such as the depth of retrieval, reranking plans, and response synthesis budgets in the context of financial technology deployments, where latency requirements are heavy because of the integration requirements of developer workflows.

The knowledge corpus in financial technology architectures comprises heterogeneous sources

such as formal API specifications, architectural decision logs of decisions made during the design, articles in the knowledge base of operational procedures, history of incidence reports of production failure modes, and transcripts of conversations that capture expert advice given in the course of past escalations. The orchestrator uses dense passage retrieval methods, which encode queries and knowledge segments into semantic vector spaces, to do similarity-based matching, which goes beyond lexical overlap and reflects semantic links between concepts. In the case of financial APIs, this feature is necessary since developers can specify the issues with business vocabulary when the corresponding documentation is written with technical terminology or the other way round.

The retrieval process follows a series of steps with the aim of maximising accuracy due to the top 20 pass limitation. Initial retrieval involves a broader net with effective approximate nearest neighbor search on the entire knowledge base, and candidate passages are found on the basis of semantic closeness. Another reranking step uses cross-attention models that encode the query and individual candidate passages together, and provide efficient relevance scores that consider fine-grained semantic relationships. This reranking step is especially useful in financial queries where similarity at the surface level can be misleading, e.g., in the differentiation between APIs to different payment methods that have similar parameter structures, but differ in significant authorization or settlement properties.

After the retrieval and reranking, the synthesis agent assembles candidate responses by retrieving the relevant information using passages and assembling coherent explanations that answer the query of the developer. The synthesis process preserves source attribution, which tracks which passages went into which statement in the generated response. This provenance tracking has various applications in the financial systems, such as allowing expert validators to establish claims to authoritative sources, support audit trails that record the origin of technical guidance, and allow focused knowledge base updates as experts detect gaps or errors in source materials. The 400-word generation budget requires synthesis agents to emphasize the most crucial information and provide actionable advice on the use of APIs, their required parameters, error handling techniques, and security options instead of giving detailed background information, which should be presented in corresponding documentation.

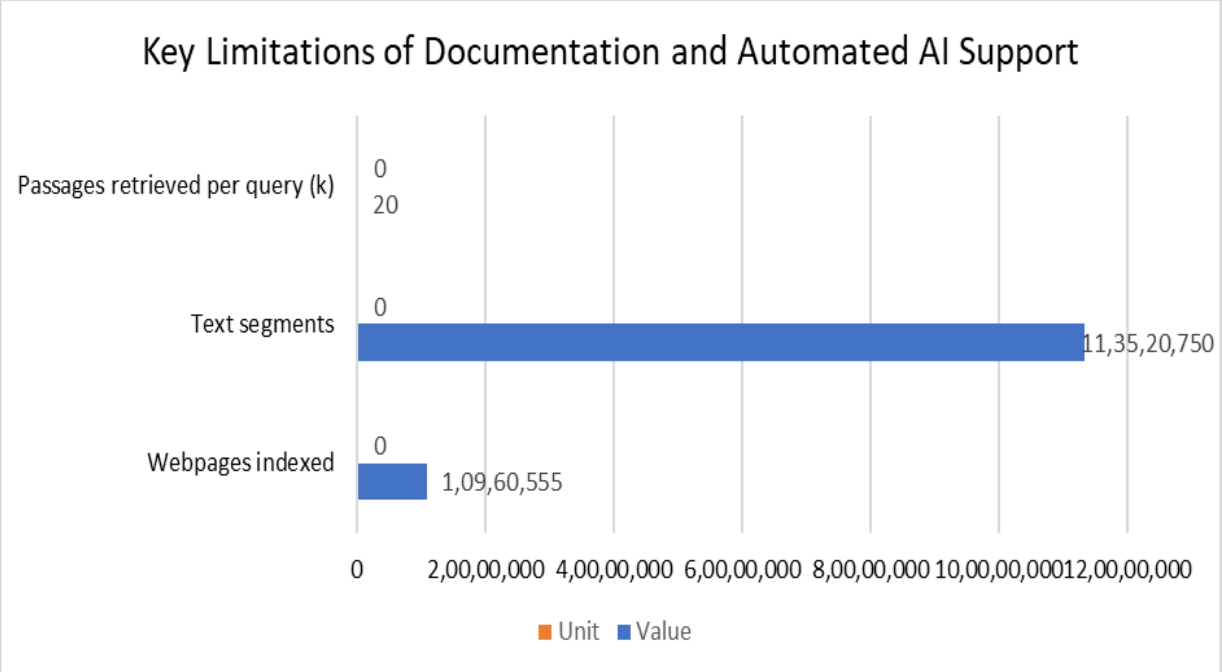

Fig. 2. Scale disparity between indexed corpora and per-query retrieval limits in agentic RAG systems [5, 6]

4. Human-in-the-Loop Validation Through Enterprise Collaboration Platform Integration

To ensure that unverified guidance never leaks out into the developer communities and to potentially spread through the codebases with the resulting errors, the framework proposes a mandatory Human-in-the-Loop approval gate directly embedded in the enterprise collaboration platforms, where API ownership teams already execute their daily business operations processes. RAG pipeline-generated candidate responses are automatically forwarded to specific subject matter experts to be verified and then published, and API support is no longer an automated broadcast model but a controlled knowledge dissemination process with evident accountability and auditability. Such an architectural decision acknowledges the fact that financial technology settings need clear approval chains of technical advice that can have an impact on security-implicit implementations, integrations that are important in compliance, or transaction-processing business logic.

The interconnection with enterprise collaboration systems solves a reported communication trend in which 44 percent of the developers use chat tools or email to communicate on APIs, and channels where critical clarifications and contextual insight may readily become diffuse or lost without organized review procedures [7]. The system places validation directly into the collaboration platform environment, eliminating the need to context switch to particular approval interfaces to validate, instead placing experts in the context of their working environment, thereby increasing the ease of the validation workflow and lowering response times. In geographically dispersed financial institutions, this integration allows follow-the-sun validation models to allow routing logic to route approval requests to geographically located experts in the appropriate time zone to keep the global development operations responsive.

The validation interface shows the experts the original developer query to make them have the complete context, the synthesized candidate response with inline source attributions that show which passages in the knowledge base informed each claim, and metadata such as retrieval confidence scores, semantic coherence measures, and references to related queries that were previously validated to ensure the expert is informed. The presentation format facilitates quick specialist assessment and gives enough background on which a decision can be made based on the accuracy and completeness of responses and the suitability of the responses to the query based on the implied use case in the query. In financial APIs, the validation criteria go beyond the technical correctness to cover the implications of regulatory compliance, alignment with best practices in security, and alignment with institutional policies on risk management.

Respondents have various response choices based on their judgment. Acceptance as is, instantly publishes the revised answer to the requesting developer, and advances the answer to certified status in the knowledge base, adding the response as an authority weight in future instances of answering an identical inquiry. With approval, annotations enable experts to add elaborations, caveats, or other context to the response without nullifying the underlying advice, which is a typical case where AI-generated responses are technically correct, but contain significant implementation warnings that are of interest to production implementations. Modification requests allow specialists to adjust individual parts of the response, keeping other parts intact, which will produce accurate correction cues that will be fed into the system's learning mechanisms. Rejection with corrective information gives the best negative signal and results in audits of the knowledge base to find out why the retrieval or synthesis did not succeed, and allows the experts to give authoritative alternative answers that override the invalid candidate.

Every interaction of validation produces organized feedback stored in a lasting audit trail that is used to address regulatory output compulsory in financial services and supports a continuous learning pipeline in the system at the same time. The audit trail captures the identity of the expert, the time of validation, any amendments or annotations made, and the reasoning behind these, and forms a complete provenance trail throughout the query to the generation of an AI response, through expert validation, and eventually publication. This auditability is necessary for the financial institutions that are regulated by the regulatory bodies about the sufficiency of their developer support processes, especially where API abuse may be part of the failures to comply with the regulators or any security attack or operational loss, which need root cause analysis.

Service level objectives can be configured in the validation workflow to be both comprehensive and easy to develop and use. Normal queries can have four-hour validation goals suitable for non-blocking guidance queries, whereas urgent queries identified by developers are expedited with one-hour goals, which can involve on-call professionals in urgent situations. Distribution of validation latency is also tracked by the system, and alerts occur when the backlog builds up, allowing capacity planning and workload balancing between expert teams. Upon validation taking more time than allowed settings, the system will apply graceful degradation mechanisms like publishing a response with high-profile disclaimers to show that the system is still undergoing validation, thus leaving the developer with just the right precautions as it maintains the human verification gateway as a knowledge base promotion.

Validation Component	Function	Output Artifacts
Query Routing	Directs candidate responses to appropriate API ownership teams based on metadata and organizational mappings	Validation request with full context
Validation Interface	Presents original query, synthesized response, source attributions, confidence scores, and similar resolved queries	Expert decision with rationale
Response Publishing	Distributes validated guidance to requesting developers and updates the knowledge base authority weights	Published response with verified status
Audit Trail Generation	Records expert identity, timestamp, decision, modifications, and rationale for regulatory compliance	Complete provenance chain
Service Level Management	Monitors validation latency against configured targets and implements escalation or graceful degradation	Performance metrics and alerts

Table 1: Human-in-the-Loop Validation Workflow Components [7, 8]

5. Continuous Improvement Through Dual-Feedback Learning Mechanisms

The framework enhances the sustained quality improvement by two complementary feedback loops that work at varying temporal scales and knowledge sources, collectively, allowing the system to adapt to real-world patterns of both expert correction in validation and real-world patterns of developer acceptance that are seen after the system deployment. Such a dual-feedback design attempts to overcome the real-life constraints of single-loop learning systems, which are optimized on either the alignment to experts or field performance alone, and which acknowledge that both views offer important cues to knowledge refinement in financial technology settings where expert opinion and practical utility in practice need to meet.

The former feedback loop, known as the Human-in-the-Loop validation process, encodes expert knowledge in the first step of the process, and each interaction with an expert is considered a guided learning experience that reveals any discrepancies between AI-generated products and expert knowledge in the domain. Human feedback has been shown to be an effective means of model alignment as research has shown that models with 1.3 B parameters trained on human preference feedback were more preferred by human evaluators compared to models with 175 B parameters without human preference feedback, suggesting that human evaluators can be overcome by deliberate feedback enough to alter the choice between large and small models [8]. This result is specifically applicable to financial technology applications, in which computational budget can limit model choice but domain-specific feedback data is rich, based on natural validation processes.

In cases where professionals edit candidate responses, the system will conduct the contrastive analysis between the AI-generated response and the expert-corrected response in order to determine systematic patterns in the differences. The patterns might indicate the failure of retrieval when the relevant authoritative passages were not retrieved, synthesis errors when information was distorted or differently synthesized based on various sources, or a knowledge gap when no documentation could have answered the query topic. In the case of retrieval failures, the system updates the weights on

passage importance and can cause reindexing with better metadata to enhance the recall in the future. In case of synthesis errors, the system constructs training sets consisting of the query and retrieved passages with the expert-corrected answer, which is used to superficially fine-tune synthesis models. In the case of knowledge gaps, the system automatically creates documentation improvement requests that are sent to API ownership teams and specify particular topics where more explanatory content is needed.

Expert comments that leave the underlying response unaltered, but include explanations or disclaimers, are useful indications of the completeness of information and contextual needs. These annotations are analyzed by the system to determine recurring themes (i.e., security warnings, compliance reminders, or performance considerations) that can be systematically incorporated into future responses to similar query patterns. In the long-term, this collection of annotated examples can allow the system to actively use domain-specific context that pure retrieval out of the official documentation may overlook, and capture the tacit knowledge that trained financial technology engineers use when consulting their coworkers.

The second loop of feedback keeps track of what the end-users do once the solution has been delivered, and developer actions and the expressed feedback are used as a measure of response utility and accuracy in real-life application scenarios. The post deployment analysis is based on the benchmarks specifically developed to identify the hallucination and knowledge incompleteness as the critical failure modes of the AI-generated technical guidance. In the case of 4,409 pairs of questions and answers explicitly marked to differentiate between answers that include fabricated information and those that are correct yet incomplete, this becomes one of the benchmarks that allow systematic evaluation of RAG systems in real-world conditions [9]. To be financially supported, hallucination detection is done to ensure that the API, parameters, and code patterns mentioned exist in existing documentation, whereas completeness evaluation is performed to determine whether the responses respond to security requirements, error handling requirements, and compliance considerations of financial technology implementations.

The system monitors several behavioral cues as quality indicators of response. The solution acceptance rate is the success rate of developers who continue with their implementation after the response came without seeking follow-up clarification or alternative solutions, to show that the original guidance covered their informational needs. Patterns of the follow-up queries show whether the developers needed further support on the related issues, which may possibly indicate insufficient details in the initial responses or the inability to predict the natural follow-up questions. Direct quality cues are explicit satisfaction ratings obtained with the help of those kinds of lightweight feedback prompts, but the response rate of these prompts is generally low without meticulous user experience design that reduces friction.

Implicit signals are especially useful when it comes to scale quality problems. When a particular response is given to developers, who in turn move to the human support avenue, the system leaves the response to be reviewed by the developers because the response probably did not address the root problem. The developers referencing a given solution in later error reports or incident tickets are investigated by the system to determine whether the guidance has contributed to the problem, which can be an indication that there are some subtle errors or mischaracterized API behavior in the response. On the other hand, when developers distribute validated answers to other developers via internal communication methods or use them in code review conversations, the system interprets such behavior as robust positive indicators of utility and reliability.

The acceptance feedback pipeline combines these signals across all deployments of any given validated response and calculates the quality measure that is used to weight the retrieval and re-validation when the quality measure falls below the thresholds. It is a statistical method of quality management on a scale of large knowledge bases deployed over varying populations of developers.

At the organizational level, these feedback mechanisms are critical towards keeping the knowledge systems functioning as they do due to the scale of software development in vendors working together. Recent reports in the industry indicate that one of the major platforms boasts a user base of more than 150 million individuals working in over 420 million repositories, which indicates the magnitude to which automated

knowledge systems need to evolve and correct themselves to stay relevant in the fast-changing technical environments [10]. To financial technology companies, this size takes the form of internal API portfolios consisting of hundreds or thousands of services, each actively evolving through version releases, feature additions, and deprecations, producing a knowledge management problem that cannot be easily addressed using a purely manual curation strategy.

Feedback Loop	Signal Source	Data Captured	Learning Mechanism
Expert Validation Feedback	Human-in-the-Loop approval phase	Modifications, annotations, rejections with corrections	Contrastive analysis between AI-generated and expert-corrected versions
Acceptance Feedback	Post-deployment developer behavior	Solution acceptance rates, follow-up query patterns, satisfaction ratings, escalation events	Statistical aggregation across response deployments
Retrieval Failure Correction	Expert modifications revealing missing passages	Queries where relevant documentation was not surfaced	Passage importance reweighting and enhanced metadata indexing
Synthesis Error Correction	Expert modifications to the response structure or the content combination	Training examples pairing queries and passages with corrected responses	Supervised fine-tuning of generation models
Knowledge Gap Detection	Expert rejections or documentation enhancement requests	Topics lacking adequate authoritative coverage	Automated requests to API ownership teams
Hallucination Detection	Benchmark evaluation and developer escalations	Responses containing fabricated information are verified against documentation	Quality metric computation and response flagging

Table 2: Dual-Feedback Learning Mechanisms [9, 10]

Conclusion

The Agentic Knowledge Orchestrator creates an AI-based developer support architecture, which is production-ready in financial technology systems by combining empirically justified retrieval strategies with binding expert validation procedures and ongoing dual-feedback learning systems. By structurally integrating human expertise as a process of knowledge refinement instead of verification being a corrective mechanism, this framework eliminates the inherent conflict between the adoption rates of AI and the lack of trust in it. Integration with enterprise collaboration platforms makes validation a natural extension of a workflow, and allows subject matter experts to control knowledge dissemination and remain as responsive to modern support systems as developers require. Dual-feedback learning takes advantage of the two corrections by the experts in approval steps and the actual acceptance of the field deployments, which form a self-enhancing knowledge ecosystem that improves in response to the knowledge of the experts in the domain and the utility in practice at the same time. The architectural weaknesses obtained through large-scale benchmark testing guarantee that the retrieval, reranking, and synthesis operations strike a tradeoff between accuracy and latency demands in line with interactive developer processes. In financial technology companies with a large API portfolio where guidance errors have regulatory, security, and operational risks, this structure gives the governance infrastructure they need to roll out AI-based support at scale without undermining the quality of standards of correctness that mission-critical systems demand. The symbiotic association between automatic creation of responses and human judgment produces a sustainable model of knowledge administration that advances with the enlargement of organizations without damaging epistemic purity through continued professional control, building a replicating pattern that is applicable to other systems of knowledge-intensive practices and various domains where a compromise between precision and scale is required.

References

[1] Satya Nadella, “Microsoft Fiscal Year 2024 First Quarter Earnings Conference Call,” Microsoft, 2023. [Online]. Available: https://www.microsoft.com/en-us/investor/events/fy-2024/earnings-fy-2024-q1

[2] Stack Overflow, “2024 Developer Survey – AI”. [Online]. Available: https://survey.stackoverflow.co/2024/ai
[3] Stack Overflow, “2024 Developer Survey,” 2024. [Online]. Available: https://survey.stackoverflow.co/2024/
[4] Kevinbrowne, "Verification debt is the AI era’s technical debt," 2025. [Online]. Available: https://www.kevinbrowne.ca/verification-debt-is-the-ai-eras-technical-debt/
[5] Patrick Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” NeurIPS, 2020. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/6b493230205f780e1bc26945df7481e5-Paper.pdf
[6] Yue Wang et al., “Laboratory for Analytic Sciences in TREC 2024 Retrieval Augmented Generation Track,” NIST. [Online]. Available: https://trec.nist.gov/pubs/trec33/papers/ncsu-las.rag.pdf
[7] Postman, “2024 State of the API Report,” 2024. [Online]. Available: https://voyager.postman.com/doc/postman-state-of-the-api-report-2024.pdf
[8] Long Ouyang et al., “Training Language Models to Follow Instructions with Human Feedback,” arXiv:2203.02155, 2022. [Online]. Available: https://arxiv.org/abs/2203.02155
[9] Xiao Yang et al., “CRAG – Comprehensive RAG Benchmark,” 38th Conference on Neural Information Processing Systems (NeurIPS 2024) Track on Datasets and Benchmarks, 2024. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/1435d2d0fca85a84d83ddcb754f58c29-Paper-Datasets_and_Benchmarks_Track.pdf
[10] Dora, “DORA State of AI-Assisted Software Development,” 2025. [Online]. Available: https://dora.dev/research/shared/dora-report-2025/