Volume 11, Issue 4, April – 2026 International Journal of Innovative Science and Research Technology
ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

Contextiva: An Integrated Framework Based on Agentic Retrieval Augmented Generation and Model Context Protocol for AI-Assisted Software Development

S. Saraswathi[1]; Thamizharasu S.[2]; Adarsh P.[3]; Sourashish Roy[4]

[1,2,3,4]Department of Information Technology, Puducherry Technological University, Puducherry, India

Publication Date: 2026/04/24

Abstract: Modern AI-assisted software development tools improve productivity but face key challenges such as context loss, fragmented knowledge access, lack of standardized communication, and poor workflow integration. This paper presents Contextiva, a context-aware knowledge management and AI task orchestration platform that addresses these limitations through a unified architecture integrating Retrieval-Augmented Generation (RAG), Model Context Protocol (MCP), and intelligent web crawling.

Contextiva employs a multi-strategy Agentic RAG framework combining semantic, keyword, and hybrid retrieval with adaptive reranking to enhance accuracy and relevance. The MCP layer enables standardized communication between AI agents and system services, supporting real-time knowledge access and task management. Additionally, intelligent document processing techniques such as chunking, code extraction, and embedding generation enable scalable knowledge representation.

Experimental results demonstrate improved retrieval performance, contextual accuracy, and workflow efficiency compared to traditional approaches. Overall, Contextiva transforms AI from a passive assistant into an active, context-aware collaborator in software development.

Keywords: *Context-Aware Systems, Retrieval-Augmented Generation (RAG), Model Context Protocol (MCP), Agentic RAG, Knowledge Management, Semantic Search, AI Task Orchestration.*

How to Cite: S. Saraswathi; Thamizharasu S.; Adarsh P.; Sourashish Roy (2026), Contextiva: An Integrated Framework Based on Agentic Retrieval Augmented Generation and Model Context Protocol for AI-Assisted Software Development.
International Journal of Innovative Science and Research Technology, 11(4), 1685-1694.
https://doi.org/10.38124/ijisrt/26apr950

I. INTRODUCTION

The rapid advancement of artificial intelligence has led to the widespread adoption of AI-powered coding assistants in modern software development. Tools such as AI copilots and intelligent code generators have significantly enhanced developer productivity by assisting with code generation, debugging, and documentation. However, despite these advancements, current systems remain fundamentally limited in their ability to maintain context, integrate knowledge sources, and participate actively in the software development lifecycle.

Most AI systems operate in a stateless or session-based manner, where context is lost between interactions. This leads to repeated prompts, inconsistent outputs, and reduced efficiency. Additionally, knowledge required for development is often distributed across multiple sources such as documentation websites, repositories, and local files, resulting in fragmented access and reduced retrieval accuracy.

II. MOTIVATION

Existing AI-assisted development systems face persistent limitations that reduce their effectiveness in real-world software projects. AI models often lose project-specific context across sessions, knowledge remains fragmented across multiple platforms, and retrieval typically depends on single-strategy approaches that limit relevance and accuracy. At the same time, the absence of a standardized communication protocol between AI agents and external tools creates integration friction, while weak connections to project and task management systems keep AI interactions

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

isolated from actual development workflows. Because of these combined gaps, AI is constrained to ad hoc assistance rather than functioning as a reliable, collaborative development partner.

These challenges create a clear need for a unified system that can preserve context over time, centralize knowledge from diverse sources, and support adaptive retrieval methods that improve response quality across different query types. Such a system should also provide standardized communication for AI-agent interoperability and embed AI directly into project execution processes, not just standalone chat interactions. This motivates the development of Contextiva, a platform designed to connect knowledge retrieval, AI reasoning, and actionable task execution within a single integrated environment.

III. RELATED WORKS

Recent studies in AI-assisted software engineering show that large language models are increasingly capable of supporting specification-to-code transformation and data-driven reasoning, yet they still depend heavily on context quality and external grounding. The work on AI-enabled executable code generation from architectural specifications demonstrates the practical potential of converting high-level design artifacts into implementation-ready outputs, but it also reveals that generation-focused pipelines can become brittle when project constraints, evolving documentation, or tool-state continuity are not explicitly integrated [1]. Similarly, effectiveness studies of LLM-based decision systems report measurable benefits in recommendation quality and analytical support, but these systems are usually evaluated in bounded domains and do not fully model continuous developer workflows, cross-session memory, or dynamic technical knowledge updates [11]. As a result, the literature establishes that model capability is no longer the primary bottleneck; instead, operational context, retrieval fidelity, and interoperability determine whether AI assistance can scale from isolated demonstrations to reliable engineering practice [1][11].

Research on the Model Context Protocol (MCP) provides an important interoperability foundation by standardizing how tools and context can be exposed to intelligent agents across environments. The scalable MCP framework for context-aware multi-agent coordination identifies protocol-level abstractions that reduce integration fragmentation and improve composability of agent operations, particularly in heterogeneous tool ecosystems [3]. More recent work on context-aware server collaboration further advances this direction by reducing central orchestration overhead and allowing coordinated server-side behaviour through shared contextual state, indicating that decentralized collaboration can improve latency and throughput without degrading coordination quality [2]. However, both lines of work are primarily focused on communication architecture and orchestration semantics rather than full-stack developer outcomes; they do not comprehensively solve retrieval strategy selection, quality-aware knowledge ingestion, or direct coupling between retrieved evidence and actionable lifecycle operations such as task updates and document evolution. Therefore, MCP literature strongly addresses transport and coordination but leaves open the challenge of integrating protocol benefits with retrieval robustness and day-to-day software delivery workflows [2-3].

Within retrieval-augmented generation, the literature shows a clear progression from baseline semantic retrieval to more adaptive and agentic retrieval behaviour. Early document question-answering pipelines establish that vector-based retrieval can substantially improve answer grounding compared with model-only responses, but they also expose limitations around single-strategy search, fixed chunk assumptions, and insufficient handling of complex multi-intent queries [4]. Agentic RAG approaches improve this by introducing retrieval planning and iterative refinement, enabling stronger support for analytical and implementation-oriented prompts where one-pass retrieval is insufficient [5]. Dynamic tagging and query-driven retrieval systems extend precision by injecting metadata-aware filtering, though they may incur consistency and scalability trade-offs when tag quality and schema evolution are not carefully controlled [6]. Context-aware domain systems, such as healthcare-oriented RAG, demonstrate that relevance improves significantly when retrieval is conditioned on domain context, but they also illustrate transferability constraints and the risk of domain lock-in [7]. Enterprise-scale intelligent retrieval systems further emphasize high-performance indexing and integrated retrieval architecture for large information estates, yet operational complexity and adaptation overhead remain notable considerations [8]. Collectively, these works indicate that no single retrieval strategy is universally optimal and that production-grade AI support requires configurable retrieval stacks that can balance precision, latency, and contextual depth [4-8].

Knowledge acquisition research confirms that retrieval quality is tightly coupled to ingestion quality, especially in web-scale documentation environments where structure and freshness vary substantially. Progressive web-agent approaches for scraper generation demonstrate that adaptive understanding of page structure can improve extraction robustness compared with rigid rule-based scraping, particularly for evolving sites and semi-structured content [9]. Domain-specific AI curation systems also show that combining assisted exploration with structured data management can increase discoverability and reduce manual curation burden, but they often remain focused on specific disciplinary ecosystems and do not automatically generalize to broader software engineering documentation pipelines [10]. Foundational surveys of web scraping techniques provide broad methodological coverage of extraction paradigms, challenges, and application domains, reinforcing that acquisition pipelines must account for heterogeneity, noise, maintenance cost, and ethical or policy constraints [12]. Taken together, this body of work demonstrates that intelligent crawling and extraction are necessary prerequisites for effective downstream RAG, but ingestion research alone does not guarantee workflow integration unless the extracted

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

knowledge is continuously normalized, indexed, and made operationally accessible to agentic systems [9-12].

A synthesis across all twelve studies shows a consistent architectural gap between strong component-level advances and complete end-to-end systems for AI-assisted development. Code-generation and decision-support research validates model usefulness but under-specifies persistent engineering context [1][11]; MCP research solves protocol interoperability but not full retrieval-to-action coupling [2-3]; RAG research expands retrieval sophistication but still faces trade-offs in configurability, transferability, and orchestration integration [4][8]; and ingestion research improves source acquisition without fully closing the loop into protocol-standardized, task-aware, and continuously updated developer operations [9-12]. The strongest inference from the literature is that next-generation systems should combine multi-strategy retrieval, context-aware coordination, intelligent crawling, and workflow actionability in one coherent architecture, where evidence retrieval is not the end-product but a trigger for concrete engineering actions such as implementation guidance, task progression, and project-state evolution. This integrated view represents the most credible path from promising research prototypes to durable, high-impact AI development infrastructure.

IV. TECHNIQUES USED

Contextiva is an AI-powered knowledge management and project coordination platform that integrates advanced computational techniques to improve information retrieval and development workflows. The system combines artificial intelligence, semantic search, and scalable architecture to efficiently manage and process large volumes of data.

Key techniques such as Retrieval-Augmented Generation (RAG), semantic embeddings, and microservices architecture allow Contextiva to retrieve relevant knowledge, assist AI agents, and support structured development processes. These techniques form the foundation of the system and enable intelligent decision-making and efficient project coordination.

➢ *Agentic Retrieval-Augmented Generation (Agentic RAG)*

Retrieval-Augmented Generation (RAG) is a technique that allows AI systems to search through a knowledge base to find relevant information before generating answers. Agentic RAG extends this by making the AI "think" about the best way to search, rather than just executing a single search method blindly. In simple terms, traditional RAG works like this: you ask a question → the system searches through documents using one method → returns results. Agentic RAG works like this: you ask a question → the system analyzes what you're asking → decides which search methods would work best → tries multiple search approaches in parallel → checks if results are good enough → improves results if needed → then provides the answer.

Contextiva's Agentic RAG uses four different search methods at the same time—one looks for similar meanings, one looks for exact keywords, one considers surrounding context, and one re-ranks results by importance. Instead of forcing every query through all methods (which is slow and expensive), the system intelligently chooses which methods to use based on what you're asking. This makes it faster, more accurate, and more cost-effective. The AI agent can also notice when results aren't good enough and automatically try different search strategies or expand the query to find better answers.

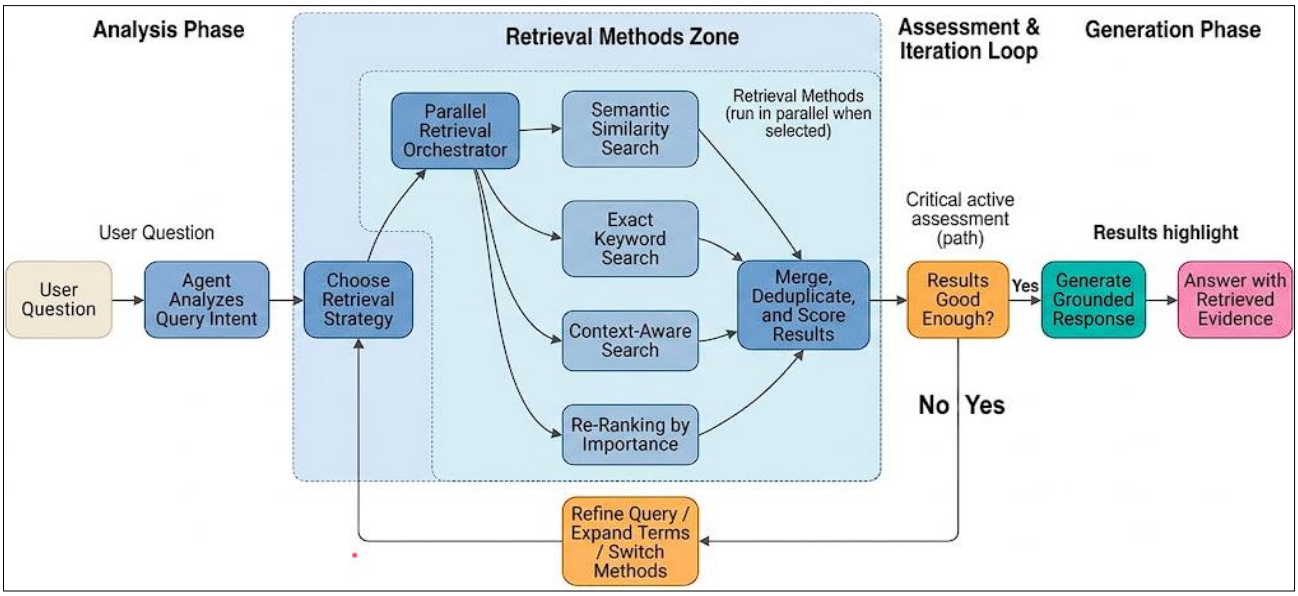

Fig 1 Concept of Contextiva's Agentic RAG

➢ *Model Context Protocol*

The Model Context Protocol (MCP) provides a standardized interface that enables AI assistants to interact with Contextiva's system services through structured communication. It integrates tools for task management and knowledge retrieval while using an HTTP-based microservices architecture to ensure lightweight deployment and easy integration with external system.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

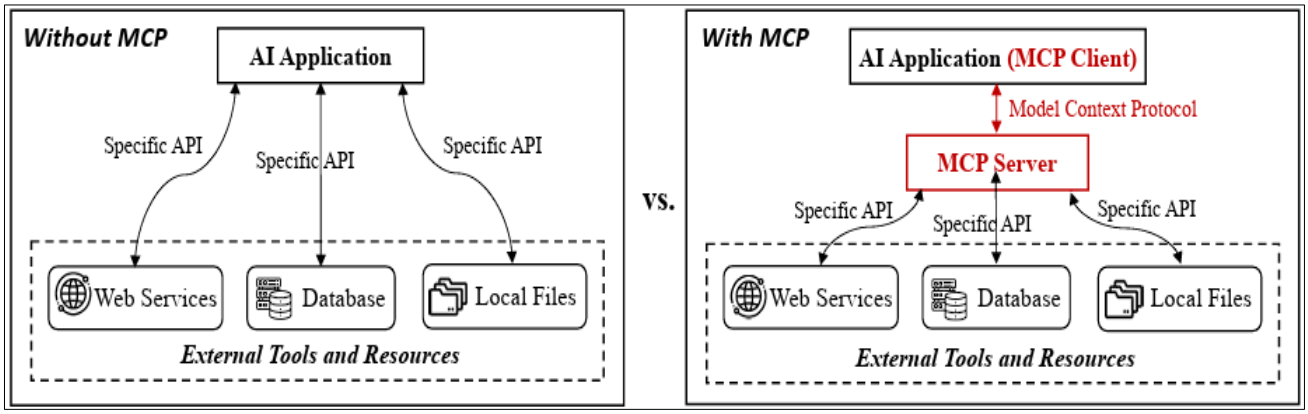

Fig 2 Comparison of AI Application with and without Model Context Protocol

➢ *Web Crawling*

An adaptive content ingestion pipeline that processes different types of URLs by first identifying whether the input is a sitemap, text file, or web page, and then applying an appropriate extraction strategy—batch processing for sitemaps, direct content extraction for text files, and recursive crawling with depth control for web pages. The extracted data is then standardized by converting it into markdown format and stored for further use, ensuring efficient, scalable, and consistent data collection for downstream applications.

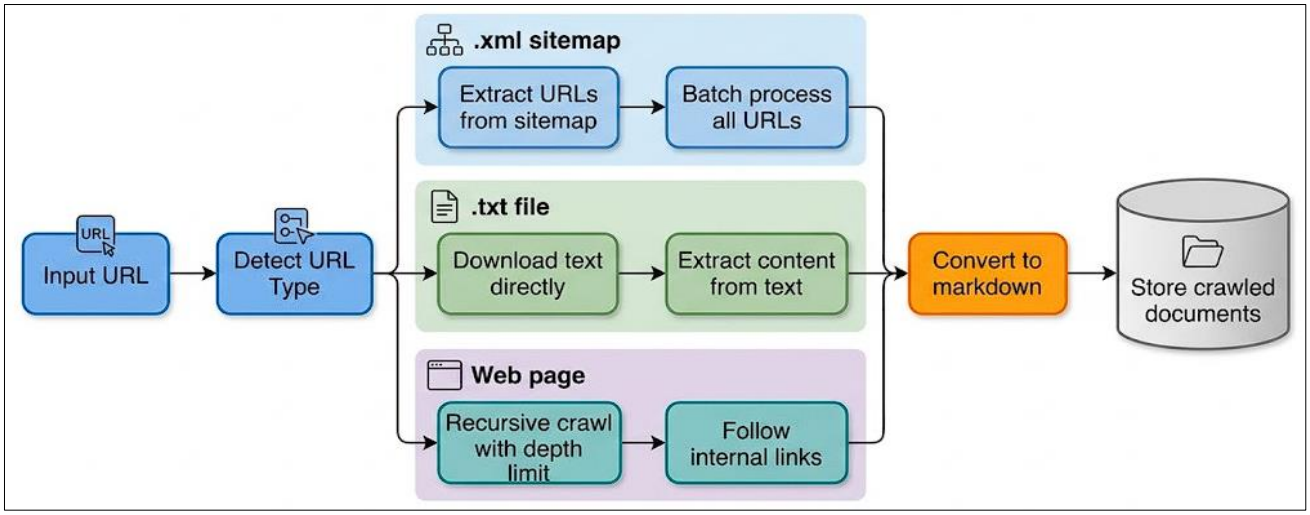

Fig 3 Concept of Contextiva's Adaptive Web Crawling

➢ *Smart Chunking*

The smart chunking algorithm divides large documents into smaller sections while preserving contextual meaning. The algorithm prefers splitting at paragraph or sentence boundaries before using word boundaries. This approach improves indexing efficiency and enhances semantic search performance.

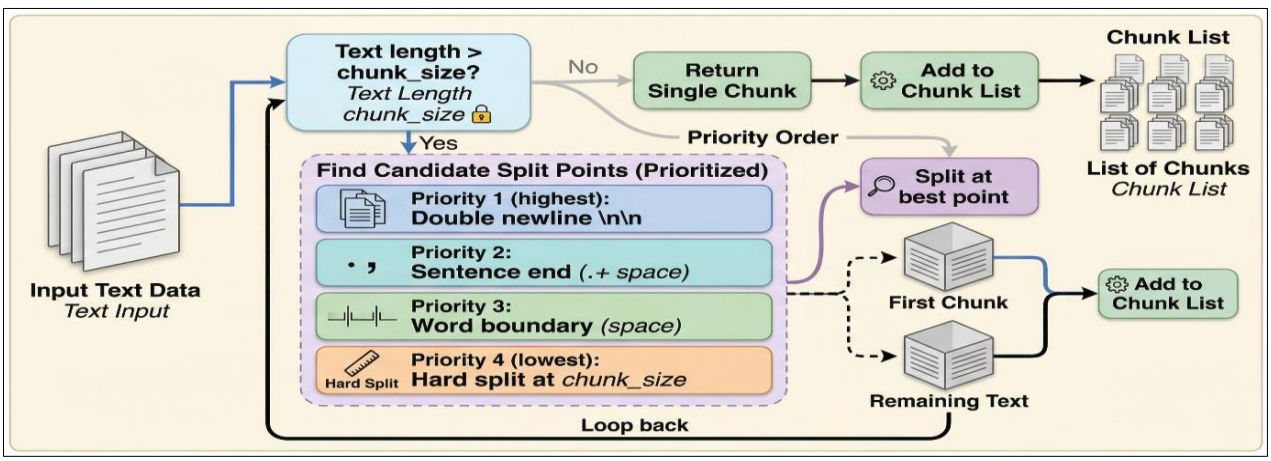

Fig 4 Concept of Contextiva's Smart Chunking of Documents/Web Pages

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

➢ *Code Extraction*

Code extraction is a multi-stage process that identifies, validates, and stores meaningful code snippets from processed documents. After chunking, the system scans various formats (text, HTML, Markdown, PDFs) to locate code blocks, filters out non-code content using validation rules like length checks and prose detection, and extracts surrounding context. Valid snippets are then deduplicated, enriched with AI-generated summaries, and stored with embeddings, making them searchable and useful for downstream applications.

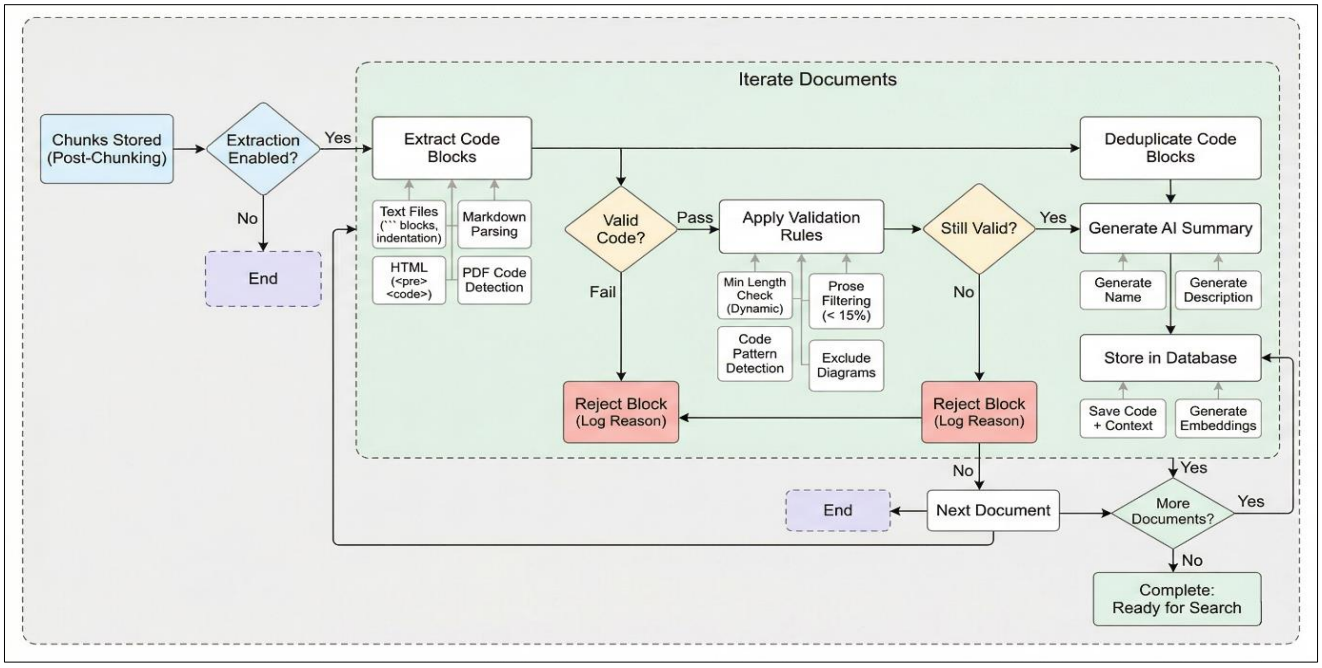

Fig 5 Concept of Contextiva's Code Extraction from Document/Web Pages

V. PROPOSED WORK

Contextiva is a context-aware knowledge management and AI task orchestration platform proposed to overcome the fundamental limitations of existing AI-assisted software development workflows. The system unifies intelligent web crawling, multi-strategy Agentic Retrieval-Augmented Generation (RAG), and Model Context Protocol (MCP) communication within a single cohesive microservices architecture.

The platform enables AI coding assistants such as Cursor, Claude Code, Windsurf, and VS Code to connect to a centralized knowledge hub through standardized MCP tools, giving them persistent project context, accurate documentation retrieval, and the ability to participate in task management. Unlike existing systems that treat knowledge retrieval, agent communication, and project management as separate concerns, Contextiva integrates all three into a unified, scalable, and developer-focused platform.

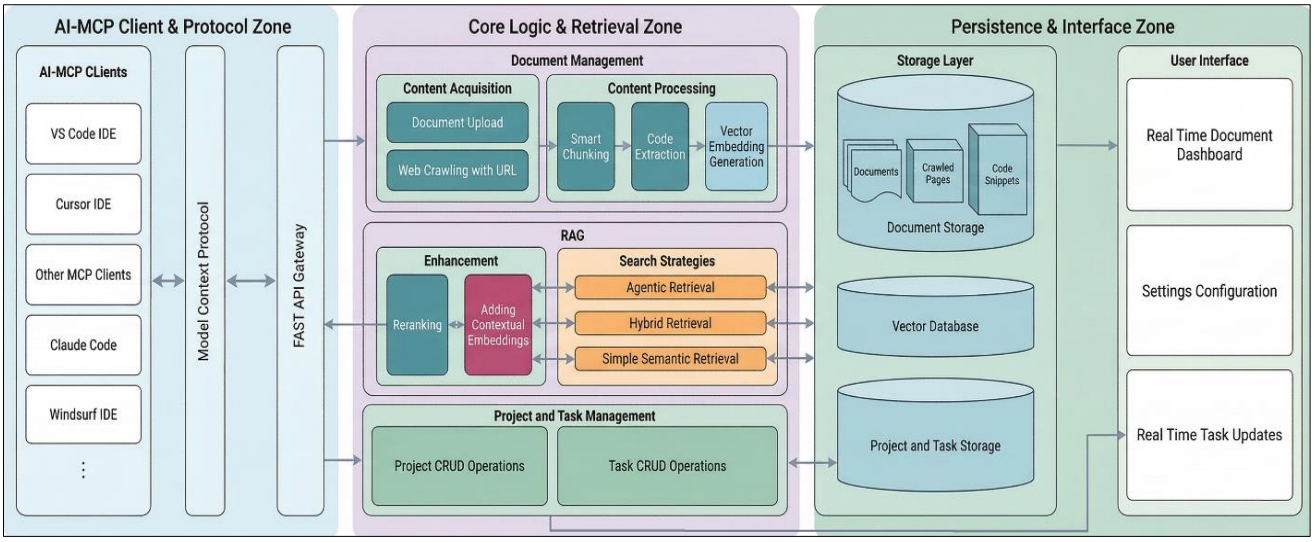

Fig 6 Architecture of Contextiva

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

In the *AI-MCP Client & Protocol Zone* (left section), the system interfaces with multiple development environments and clients such as VS Code IDE, Cursor IDE, Claude Code, Windsurf IDE, and other MCP-compatible clients. These clients communicate with the backend through a standardized *Model Context Protocol*, which ensures consistent formatting and handling of requests and responses. A *FastAPI Gateway* acts as the entry point, managing API requests, routing them to appropriate services, and enabling seamless integration between external clients and the system's internal components. This layer essentially abstracts the complexity of backend operations and provides a unified interface for developers and tools.

The central section, labelled the *Core Logic & Retrieval Zone*, represents the heart of the system where all major processing and intelligence-driven operations occur. It begins with *Document Management*, which is divided into two key stages: content acquisition and content processing. Content acquisition includes document uploads and web crawling via URLs, allowing the system to gather data from multiple sources. Once acquired, the content undergoes processing steps such as *smart chunking*, which splits documents into manageable segments, *code extraction*, which identifies and isolates relevant code snippets, and vector embedding generation, which converts textual data into numerical representations suitable for semantic search.

Within the same core zone, the *RAG (Retrieval-Augmented Generation)* module enhances the system's ability to retrieve and generate relevant information. It incorporates multiple search strategies, including *simple semantic retrieval, hybrid retrieval, and agentic retrieval*, each progressively increasing in sophistication and contextual awareness. Additionally, enhancement techniques such as *reranking* and *contextual embedding augmentation* are applied to improve the relevance and accuracy of retrieved results. This ensures that downstream tasks, such as answering queries or generating outputs, are based on highly refined and contextually appropriate information.

Also integrated into the core logic is the *Project and Task Management* component, which supports CRUD (Create, Read, Update, Delete) operations for both projects and tasks. This allows users to organize workflows, manage processing pipelines, and maintain structured records of ongoing activities. It highlights that the system is not only focused on data processing but also on operational management and productivity.

The *Persistence & Interface Zone* (right section) handles data storage and user interaction. The storage layer is divided into multiple repositories, including document storage (for raw documents, crawled pages, and extracted code snippets), a vector database (for storing embeddings used in semantic retrieval), and project and task storage (for managing workflow-related data). This separation ensures efficient data management and optimized retrieval performance. On the interface side, the system provides a real-time document dashboard, enabling users to monitor document processing activities, along with settings configuration for customization and real-time task updates for tracking progress dynamically.

VI. DATASETS USED

- *HumanEval Dataset*

The HumanEval Dataset is a widely used benchmark for evaluating the functional correctness of code generated by language models. Introduced by OpenAI, it consists of a set of hand-written programming problems, primarily in Python, each accompanied by a function signature, a docstring describing the task, and a suite of hidden unit tests. Instead of relying on traditional string-matching metrics, HumanEval assesses performance by executing the generated code against these tests, making it a robust measure of whether the model truly understands the problem and produces correct logic. This dataset is particularly important in research on code generation because it emphasizes correctness over superficial similarity, and it has become a standard benchmark for comparing large language models in programming tasks.

- *CodeNet Dataset*

The Project CodeNet Dataset, developed by IBM, is one of the largest publicly available datasets for code intelligence research. It contains millions of code samples collected from competitive programming platforms, covering multiple programming languages and a wide variety of problem statements. Each submission is labelled with metadata such as problem IDs, programming language, and execution status (e.g., accepted, or incorrect), enabling tasks like code classification, error detection, and performance prediction. Unlike HumanEval, which is relatively small and focused on functional evaluation, CodeNet provides a large-scale, diverse corpus suitable for training and benchmarking machine learning models on real-world coding behaviour, making it highly valuable for both supervised learning and large-scale empirical studies.

VII. EVALUATION METRICS

- *Exact Match*

The Exact Match metric measures the percentage of predictions that match the ground truth exactly.

$$EM = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}(y_i = \hat{y}_i)$$

- *BLEU-4*

Bilingual Evaluation Understudy (BLEU) scores evaluate the quality of text which has been machine-translated.

$$BLEU_4 = BP \cdot exp\left(\sum_{n=1}^{4} w_n \log p_n\right)$$

- *ROUGE-L*

Recall-Oriented Understudy for Gisting Evaluation (ROUGE) focuses on recall. ROUGE-L specifically uses the Longest Common Subsequence (LCS).

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

$$ROUGE - L = \frac{(1+\beta^2) \cdot R_{lcs} \cdot P_{lcs}}{R_{lcs} + \beta^2 \cdot P_{lcs}}$$

- *CodeBLEU*

Specialized for programming languages, CodeBLEU incorporates syntactic and semantic features.

$$CodeBLEU = \alpha \cdot BLEU + \beta \cdot BLEU_{weight} + \gamma \cdot Match_{AST} + \delta \cdot Match_{DF}$$

- *BERTScore F1*

BERTScore leverages contextual embeddings to calculate similarity between tokens in the candidate and reference sentences.

$$F_{BERT} = 2 \cdot \left(\frac{P_{BERT} \cdot R_{BERT}}{P_{BERT} + R_{BERT}}\right)$$

- *Precision@5*

In this case, calculated for the top 5 results.

$$Precision@5 = \frac{[\text{Relevant Items} \cap \text{Top 5 Retrieved Items}]}{5}$$

- *Recall@5*

Calculated for the top 5 results relative to the total number of relevant items.

$$Recall@5 = \frac{[\text{Relevant Items} \cap \text{Top 5 Retrieved Items}]}{|\text{Total Relevant Items}|}$$

- *Mean Reciprocal Rank*

MRR assesses systems that return a ranked list of answers to queries.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

- *NDCG@5*

Normalized Discounted Cumulative Gain (NDCG) measures the quality of ranking based on the position of relevant documents.

$$NDCG@5 = \frac{\text{DCG@5}}{\text{IDCG@5}} = \frac{\sum_{i=1}^{5} \frac{rel_i}{log_2(i+1)}}{\sum_{i=1}^{5} \frac{rel^{ideal}{}_i}{log_2(i+1)}}$$

VIII. RESULTS

Across both datasets, the Existing baseline (Traditional RAG) provides stable retrieval behavior but shows limited adaptability when query intent varies across problem types. The Agentic RAG pipeline improves orchestration by adding query reasoning, strategy-aware retrieval, and code-focused context handling, which makes the system more responsive to task complexity instead of treating every prompt with a uniform retrieval pattern. This is especially visible in code-oriented quality behavior, where Agentic RAG tends to produce outputs that are structurally closer to executable coding expectations rather than only maximizing surface-level lexical overlap.

In the CodeNet setting, Agentic RAG demonstrates broader quality gains over the Existing baseline, indicating that agentic planning and context refinement are effective for larger and more diverse coding tasks. In the HumanEval setting, performance is more mixed: code-aware quality remains competitive or better, while some lexical-overlap indicators do not consistently improve. This suggests that Agentic RAG is better at semantic and structural alignment than strict token-level matching in short, canonical benchmark answers. Overall, the comparison indicates that Existing systems are reliable but rigid, while Agentic RAG is more flexible and workflow-aligned for practical coding assistance.

The comparison also reinforces key architectural limitations of Existing systems: fixed retrieval behavior, weaker intent adaptation, limited guidance controls, and less explicit coupling between retrieval results and developer workflow actions. Agentic RAG addresses these by enabling strategy-aware retrieval and richer context handling, but dataset-dependent variation shows that optimization still depends on task type and evaluation objective. Therefore, Agentic RAG should be interpreted as an adaptive improvement over Existing baselines, with strongest gains in code-structure-aware tasks and more nuanced outcomes on strict lexical benchmarks.

Table 1 Testing on HumanEval (Traditional vs Agentic)

Metrics	Traditional RAG	Agentic RAG
Exact Match	0	0
BLEU-4	0.004	0.0013
ROUGE-L	0.0528	0.0289
CodeBLEU	0.0681	0.0776
BertScore F1	0.7544	0.757
Precision@5	0.1	0.1
Recall@5	0.5	0.5
MRR	0.375	0.375
NDCG@5	0.4077	0.4077

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

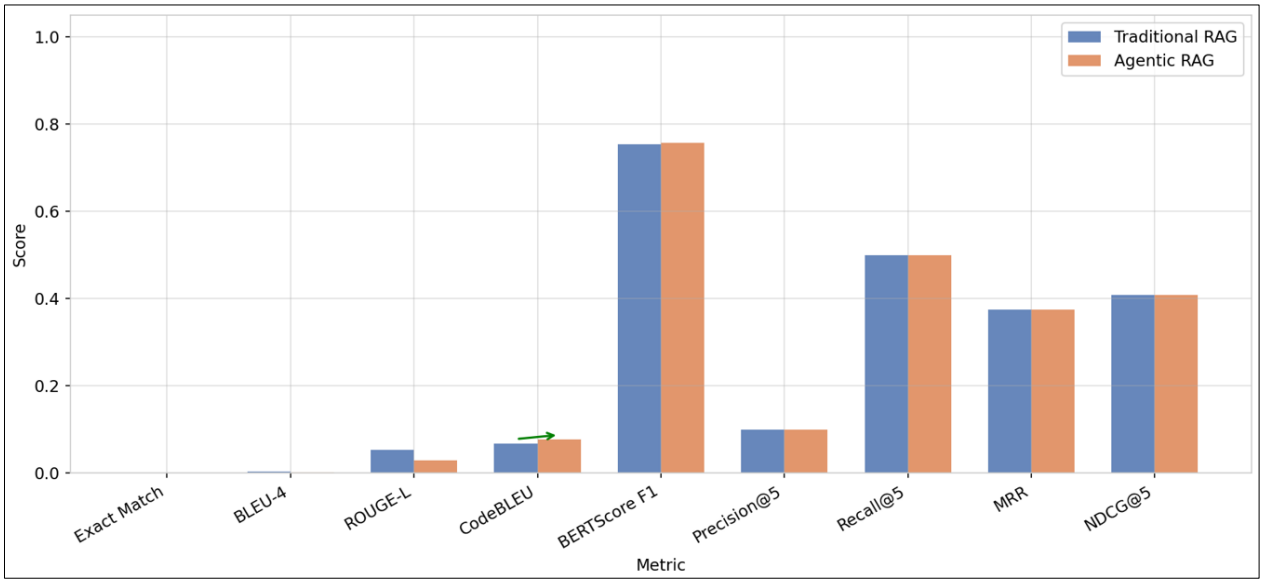

Fig 7 BarPlot for Traditional RAG vs Agentic RAG on HumanEval

Table 2 Testing on CodeNet (Traditional vs Agentic)

Metrics	Traditional RAG	Agentic RAG
Exact Match	0	0.0099
BLEU-4	0.0058	0.0091
ROUGE-L	0.0401	0.0571
CodeBLEU	0.044	0.052
BertScore F1	0.7551	0.7584
Precision@5	0.1	0.12
Recall@5	0.5	0.5143
MRR	0.375	0.3939
NDCG@5	0.4077	0.4334

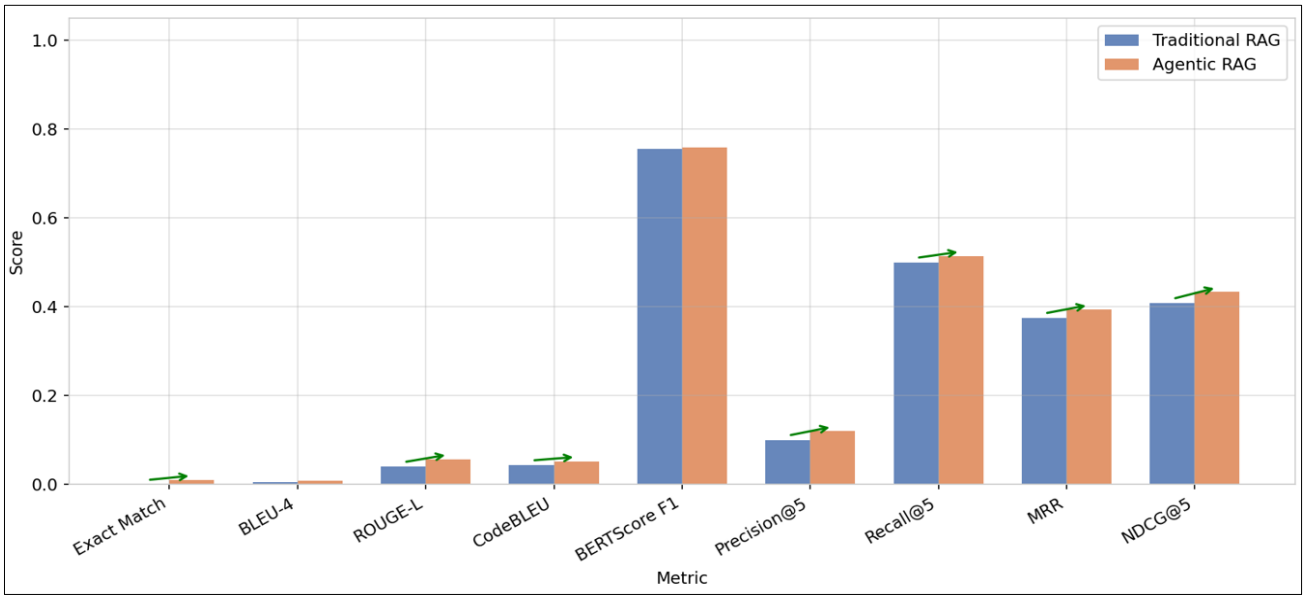

Fig 8 BarPlot for Traditional RAG vs Agentic RAG on CodeNet

IX. CONCLUSION

Thus, the project demonstrates that an integrated AI platform can significantly improve how teams capture, organize, and act on knowledge by unifying document processing, semantic retrieval, task orchestration, and agent-driven assistance in one coherent workflow. Instead of treating research, planning, and execution as disconnected activities, the system links them through shared context, enabling faster iteration, better traceability, and more consistent decision-making across technical and non-technical users. The current implementation proves the

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

practical value of combining retrieval-augmented intelligence with project management primitives, especially in environments where information changes rapidly and teams need both speed and reliability. By emphasizing modular architecture, clear service boundaries, and extensible tooling patterns, the platform has established a strong foundation for continuous evolution while already delivering measurable gains in developer productivity, knowledge reuse, and operational transparency

Future enhancements should focus on advancing both intelligence quality and production robustness to move from a strong beta platform toward enterprise-grade maturity. On the intelligence side, key priorities include hybrid retrieval pipelines that combine vector similarity with keyword and metadata ranking, richer reranking and evaluation frameworks for answer quality, stronger citation grounding, and expanded multimodal capabilities for diagrams, tables, code-heavy documents, and audio-derived transcripts. On the product side, improvements such as fine-grained role-based access control, collaborative review workflows, smarter task automation policies, and deeper analytics for usage and outcome tracking will strengthen organizational adoption. At the infrastructure layer, enhanced observability, autoscaling strategies, failure isolation, and resilience testing will be critical for dependable large-scale deployment. Together, these enhancements can transform the platform into a comprehensive AI-native operating layer for knowledge, execution, and continuous delivery.

REFERENCES

[1]. Padmavathi, Sreenivasachar, Raghavendra., “An AI Enabled Solution for Generating Executable Code from Architectural Specification,” *Internation Journal of Engineering Research & Technology (IJERT).*, Volume 14, Issue 06, June 2025.

[2]. Jayanti, M.A., & Han, X.Y., "Enhancing Model Context Protocol (MCP) with Context-Aware Server Collaboration", *ArXiv, abs/2601.11595,* 2026

[3]. Parwani, K, Das S, Vijay D.K., “Model Context Protocol (MCP): A Scalable Framework for Context-Aware Multi-Agent Coordination,” *International Journal of Multidisciplinary Research.,* Volume 1, Issue 4, September 2025.

[4]. K. Muludi, K. M. Fitria., "Retrieval-Augmented Generation Approach: Document Question Answering using Large Language Model," *International Journal of Advanced Computer Science and Applications (IJACSA).,* Volume 15, Issue 3, 2024.

[5]. P. Singh, A. Jamdar, P Kaul., "Agentic Retrieval Augmented Generation: Advancing AI-Driven Information Retrieval and Processing," *International Journal of Computer Trends and Technology*., Volume 73, Issue 1, January 2025.

[6]. Surya Kuchibhotla, Sri Kuchibhotla., "Retrieval-Augmented Generation System for Dynamic Document Tagging and Query-Driven Retrieval," *International Journal of Science and Research Archive (IJSRA).*, Volume 16, Issue 1, July 2025

[7]. N. Kolhare, R. Dhapse, Om Patil, B. Kore., "Proposing a RAG-Based System for Context-Aware Healthcare Monitoring," *International Journal of Engineering Research & Technology (IJERT).*, Volume 14, Issue 4, May 2025.

[8]. Padmavathi, Sreenivasachar, Raghavendra., "IRIS: Intelligent Retrieval Information System," *International Journal of Engineering Research & Technology (IJERT).*, Volume 14, Issue 6, June 2025.

[9]. W. Huang, Z. Gu, C. Pen., "AUTOSCRAPER: A Progressive Understanding Web Agent for Web Scrapper Generation," *In Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*., pages 2371–2389, Miami, Florida, USA. Association for Computational Linguistics , November 2024.

[10]. M. J. Elliot, M. Luciano, J. Fortes., "AI-Assisted Exploration, Curation, and Extension of Biodiversity Data Using iChatBio," *Biodiversity Information Science and Standards*, 9. e181968, December 2025.

[11]. B. Jiang, Q. Fan, J. Zhou., "Effectiveness Evaluation and Application of Large Language Model in Data-Driven Teaching Decision-Making," *International Journal of Evaluation and Research in Education (IJERE).*, Volume 14, Issue 3, June 2025.

[12]. L. Chaimaa, S. Swetha, E. Myriam, L. Imen., "Web Scrapping Techniques and Applications: A Literature Review", *SCRS Conference Proceedings on Intelligent Systems.*, 2021

[13]. Sanikommu N. R., “Model Context Protocol: Enhancing LLM Performance for Observability and Analytics”, *European Journal of Computer Science and Information Technology*,13(29),112-126, May 2025

[14]. Yu, H., Gan, A., Zhang, K., Tong, S., Liu, Q., Liu, Z.., “Evaluation of Retrieval-Augmented Generation: A Survey”. *Communications in Computer and Information Science*, vol 2301, 2025.

[15]. Puri, Ruchir & Kung, David & Janssen, Geert & Zhang, Wei & Domeniconi, Giacomo & Zolotov, Vladmir & Dolby, Julian & Chen, Jie & Choudhury, Mihir & Decker, Lindsey & Thost, Veronika & Buratti, Luca & Pujar, Saurabh & Finkler, Ulrich. “Project CodeNet: A Large-Scale AI for Code Dataset for Learning a Diversity of Coding Tasks”. *10.48550/arXiv.2105.12655*, 2021.

[16]. Chen, Mark et al. “Evaluating Large Language Models Trained on Code.” *ArXiv abs/2107.03374*., 2021.

[17]. Gan, Aoran et al. “Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey.” *ArXiv abs/2504.14891*., 2025.

[18]. Chanhee Park, Hyeonseok Moon, Chanjun Park, and Heuiseok Lim., “MIRAGE: A Metric-Intensive Benchmark for Retrieval-Augmented Generation Evaluation”. *In Findings of the Association for Computational Linguistics: NAACL*, 2025.

[19]. Abdallah, Abdelrahman et al. “RerankArena: A Unified Platform for Evaluating Retrieval, Reranking and RAG with Human and LLM Feedback.” *Proceedings of the 34th ACM International Conference on Information and Knowledge Management.*, 2025.

ISSN No: -2456-2165 https://doi.org/10.38124/ijisrt/26apr950

[20]. Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert., "RAGAs: Automated Evaluation of Retrieval Augmented Generation". *In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, 2024.

AUTHOR'S BIOGRAPHY

Dr. S. Saraswathi earned her Ph.D. from Anna University, with a specialization in Natural Language Processing. She currently serves as a professor in the Department of Information Technology, Puducherry Technological University and has an extensive publication record, with numerous research papers in esteemed refereed journals and international conferences. Beyond her research, Dr. Saraswathi is dedicated to mentoring students and has guided numerous graduate and doctoral candidates, helping them to achieve academic and professional success.

Thamizharasu S. is a B.Tech (IT) student at Puducherry Technological University, currently focused on the fields of Data Engineering, Artificial Intelligence and Machine Learning. He has proficiency in Python, AWS, Snowflake, Docker and Kubernetes.

Adarsh P. is a B.Tech (IT) student at Puducherry Technological University, currently focused on the fields of Articial Intelligence and LLMs. He has proficiency in Python, C, C++ and MySQL.

Sourasish Roy is a B.Tech (IT) student at Puducherry Technological University, currently focused on the fields of Full Stack Web Development ,Articial Intelligence and Machine Learning. He has proficiency in Python, AWS, Docker and MERN.