ContextRAG Whitepaper v1

Give LLMs the Context They Need A Context Extension Platform for AI Systems

Abstract

Large Language Models have revolutionized how developers interact with software systems. However, these models operate without awareness of the specific projects, architectures, and context in which they are used. ContextRAG introduces a Context Extension Platform that augments LLM interactions with project■specific knowledge, developer conversations, and continuously ingested documentation. By combining Retrieval Augmented Generation (RAG) with an event■driven knowledge ingestion pipeline, ContextRAG enables developers to interact with AI systems that are aware of their project environments.

The Problem

Developers rely on AI assistants like ChatGPT, Claude, and Gemini, yet these systems lack awareness of the developer’s project context. Developers must manually copy documentation or code into prompts. Previous discussions are lost between sessions. Architectural knowledge is not retained. ContextRAG introduces a contextual retrieval layer that enriches prompts with relevant project information automatically.

Vision

ContextRAG introduces a Context Extension Layer for AI systems. Instead of interacting directly with LLMs, developers interact with a system that retrieves project context, combines it with the query, and then forwards the enriched prompt to the LLM.

Core Concept: Retrieval Augmented Generation

ContextRAG uses Retrieval Augmented Generation (RAG). A user query is converted into embeddings, similar vectors are retrieved from a vector database, and the resulting context is merged with the user prompt before sending it to the LLM.

System Architecture

Core components include a Custom Multi■LLM UI, Java RAG Orchestration Service, Embedding Model, Vector Database (Weaviate), LLM Providers, Kafka Event Pipeline, and Conversation Summarization Layer.

Knowledge Ingestion Pipeline

Project knowledge such as documentation, code snippets, and architecture diagrams is parsed, chunked, embedded, and stored in a vector database to enable contextual retrieval.

Event Driven Knowledge Pipeline

Kafka enables asynchronous ingestion of knowledge from repositories, logs, or documentation pipelines. Consumers process events to create embeddings and update the vector database.

Conversation Memory Loop

User interactions generate conversation events which are summarized, embedded, and stored as searchable knowledge. This allows the system to learn from developer interactions over time.

Multi■LLM Architecture

ContextRAG supports multiple LLM providers including OpenAI, Claude, Gemini, and local LLMs via Ollama. The UI allows dynamic model selection.

API■First Architecture

The RAG orchestration layer exposes APIs enabling integration with Web UI, IDE plugins, CLI tools, and automation agents.

Scalability Model

Stateless orchestration services, distributed vector databases, Kafka pipelines, and parallel ingestion consumers enable horizontal scalability.

Security Considerations

ContextRAG may process proprietary code. Security features include authentication, encrypted vector storage, role■based access control, and audit logging.

Deployment Models

Deployment options include local developer mode, team deployments with shared vector databases, and enterprise deployments with distributed ingestion pipelines.

Long Term Vision

ContextRAG could evolve into an AI engineering intelligence platform capable of analyzing observability data, product analytics, and architecture patterns.

Conclusion

ContextRAG extends LLM capabilities by providing a persistent contextual knowledge layer built around developer artifacts and conversations, enabling truly context■aware AI assistance.