

Retrieval-Augmented Generation (RAG)

RAG (Retrieval-Augmented Generation) is an AI architecture pattern that combines a retrieval system with a large language model (LLM) to generate accurate and context-aware answers.

Problem RAG Solves:

Large language models do not know private data, may hallucinate, and have knowledge cutoff dates.

RAG retrieves relevant information from your own data source, injects it into the prompt, and generates grounded answers.

High-Level Architecture:

User Query → Retriever (Vector Search) → Relevant Documents → LLM → Final Answer

How RAG Works:

1. Ingestion Phase (Offline):

- Load documents (PDF, DB, CSV, APIs)
- Split into chunks
- Convert chunks into embeddings
- Store embeddings in a vector database

2. Retrieval Phase (Runtime):

- Convert user query to embedding
- Search vector database
- Return top relevant chunks

3. Generation Phase:

- Provide retrieved chunks as context to LLM
- LLM generates grounded response

Why RAG Is Powerful:

- Reduces hallucination
- Uses private/company data
- Works with large document sets
- No fine-tuning required
- Easy to update (re-index data)

RAG vs Fine-Tuning:

RAG is dynamic, cheaper, easier to maintain, and works well with changing data.

Fine-tuning is static, expensive, and harder to update.

Real-World Use Cases:

- Banking knowledge assistants
- Legal document search
- Internal company chatbot
- Travel apps
- AI copilots

Advanced Production Patterns:

- Hybrid search (keyword + vector)
- Re-ranking models
- Metadata filtering

- Chunk optimization
- Embedding caching
- Streaming responses
- Guardrails and validation layers