

Framework-Free RAG Implementation with Azure OpenAI



Document Sources

PDF, DOCX, TXT files Internal knowledge base Department documentation



Document Processing

Text extraction Sentence-aware chunking Metadata generation



Vector Store

ChromaDB with persistence Sentence transformer embeddings Semantic indexing



Semantic Search

Query contextualization Vector similarity search Relevant chunk retrieval



Azure OpenAl

GPT-4 Contextual response generation Prompt engineering



Conversational Memory

Session management Chat history tracking Follow-up handling



User Interface

Web-based chat interface | Session support



X Technology Stack

Azure OpenAl GPT-4, GPT-3.5-turbo

ChromaDB Vector database **Sentence Transformers** all-MiniLM-L6-v2

Document Processors PyPDF2, python-docx

Python Core implementation



RAG Workflow Steps



Document IngestionProcess department documents (PDF, DOCX, TXT) and extract text content with metadata



Text ChunkingSplit documents into sentence-aware chunks (~500 chars) for optimal retrieval



Embedding GenerationGenerate vector embeddings using sentence-transformers model



Vector Storage

Store embeddings in ChromaDB with persistent storage and metadata



Query ProcessingContextualize user queries using conversation history



Semantic Retrieval

Find most relevant document chunks using vector similarity



Response Generation

Use Azure OpenAl to generate contextual responses with retrieved information



Memory Update

Update conversation history for follow-up questions and context