Prompt Engineering for RAG:

GUVI | HCL
Skill Up. Level Up

Prompt Engineering for RAG

Prompt engineering in RAG involves crafting structured templates that combine user queries with retrieved context. Well-designed prompts help LLMs generate accurate, grounded, and context-aware responses based on relevant documents.

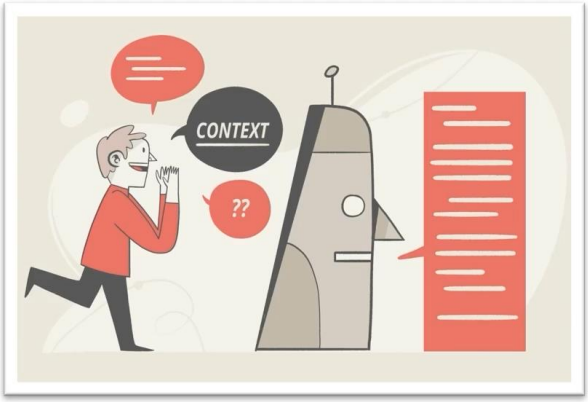
Prompt Engineering for RAG

GUVI | HCL
Skill Up. Level Up

Prompt Engineering in RAG refers to crafting effective prompts that guide the LLM to generate accurate, relevant, and grounded answers using retrieved context from a vector database.

In RAG, you're not just giving the model a question — you're giving it:

- ❑ A **question**
- ❑ Some **retrieved context**
- ❑ And an **instruction** on how to use that context

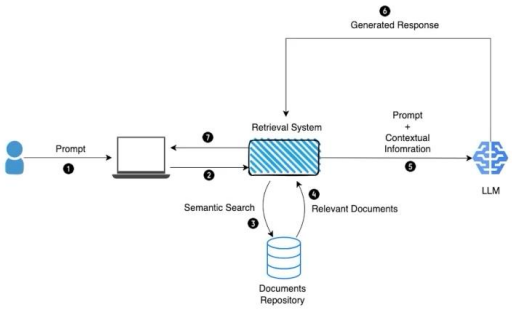


RAG Flow Recap

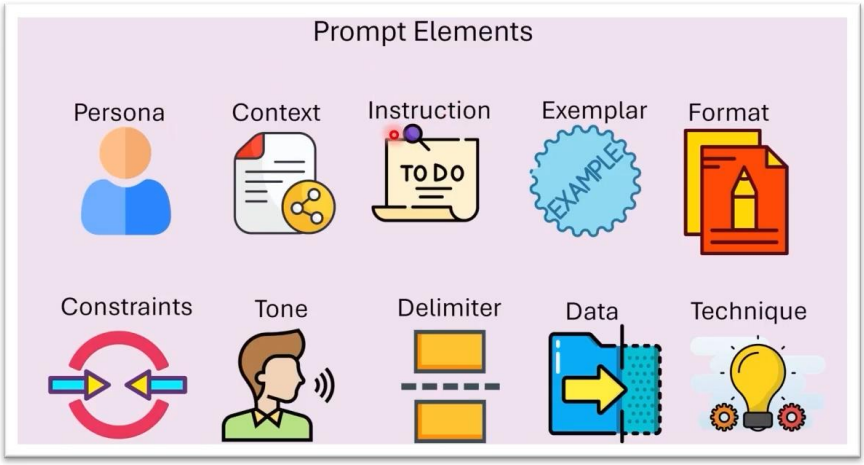
User Query → Retriever → Context → 🧠 Prompt → LLM → Answer

Prompt Engineering influences:

- ❑ How the model interprets the query
- ❑ How it uses the retrieved chunks
- ❑ How confident, concise, and factual the response is



Prompt Template Components



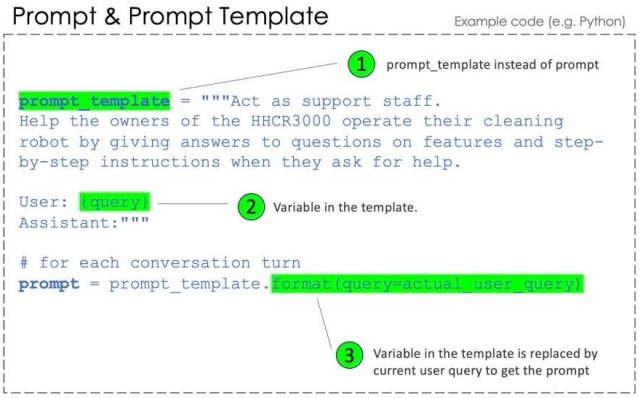
Example Prompt Template

You are an AI assistant. Use the following context to answer the question accurately and concisely.

Context:
{retrieved_context}

Question:
{user_query}

Answer in a helpful tone. If the answer is not found in the context, say "The information is not available."



Techniques in Prompt Engineering for RAG

Technique	Goal
Context Injection	Insert relevant info before question
Chain-of-Thought Prompting	Ask the model to explain its reasoning
Answer Formatting	Force structured replies (bullets, JSON, tables)
Fallback Handling	Handle cases where context is insufficient
Tone/Style Control	Adjust tone (e.g., formal, friendly, technical)

Best Practices

- ❑ Keep prompts short and consistent
- ❑ Include system role or persona if needed
- ❑ Limit context size to stay within token limits
- ❑ Escape hallucination by telling the model: "If unsure, say you don't know."
- ❑ Use few-shot prompting only if needed (more costly)