🧠 RAG: Generation Phase — The Full Picture

🔁 Recap: Where Are We Now?

You already:

- **Indexed**: Turned documents into embeddings, stored in a vector DB.
- **Retrieved**: Pulled top-k chunks based on similarity to user query.

Now you feed those retrieved chunks to a **Language Model (LLM)** to generate responses.

🧩 1. Input Structure: What the LLM Gets

🔹 Components:

- **User Query (Q)** — e.g. *"What are the symptoms of malaria?"*
- **Retrieved Context (C1, C2... Ck)** — chunks from vector store
- **Prompt Template (P)** — controls how Q + C are formatted before hitting the model

🔹 Common Prompt Format:

You are a helpful assistant. Use the context below to answer the question.

Context:

{context_chunk_1}

{context_chunk_2}

...

Question: {user_query}

Answer:

This **prompt engineering** is 🔑 to getting coherent, grounded outputs.

🛠️ 2. How Generation Happens

Option A: 🔗 LLM-as-a-Service

- You call OpenAI GPT-4, Claude, etc.

- Send prompt as input
- Get generated output

Option B: 🧠 Local Model

- Using models like LLaMA, Mistral, or Mixtral via Transformers
- Run on local GPU or CPU (slow but flexible)

Sample code (OpenAI GPT):

```
import openai

response = openai.ChatCompletion.create(
 model="gpt-4",
 messages=[
  {"role": "system", "content": "Use the provided context to answer the question."},
  {"role": "user", "content": f"Context:\n{retrieved_context}\n\nQuestion:\n{query}"}
 ],
 temperature=0.2
)

answer = response['choices'][0]['message']['content']
```

🧰 3. Generation Strategies

🧪 Temperature & Top-p

Control **creativity vs determinism**.

- temperature=0.0: deterministic
- temperature=1.0: more creative/random
- Use **0.2–0.5** for factual tasks

🧠 Max Tokens

Limit how long the model can generate.

- max_tokens=256 is common

💭 Stop Sequences

Prevent model from rambling endlessly.

- Add things like "\n\n", "###", "Answer:" to stop generation early

---

🤖 4. Avoiding Hallucinations (Critical!)

RAG = Retrieval-Augmented Generation. Emphasis on **retrieval**.

If your LLM isn't grounded in retrieved context, it will hallucinate.

Tips:

- Always format the context clearly in the prompt
- Keep retrieval chunks small & focused
- Use temperature=0
- Consider appending: **"Only answer from the context."**

🔄 5. Output Handling

A. Display

- Just show the output to user
- Optional: highlight which chunk(s) were used

B. Post-processing (Optional)

- Add citations to retrieved chunks
- Filter out incomplete sentences
- Add formatting for UI

🧪 6. Evaluation of Generation

You gotta know if your output is:

- **Factual**
- **Relevant**
- **Concise**
- **Cited properly (if needed)**

Metrics:

- BLEU, ROUGE — if ground-truth answers available
- Human eval — for real-life systems
- Faithfulness/Reliability scores (OpenAI's evals)

🔐 7. Security & Privacy (Don't Skip This)

If you're using external LLMs:

- Never send sensitive/private info in prompts
- Consider local models for private data apps

✅ Chain-of-Thought (CoT)

- Add "Let's think step by step" to prompt to improve reasoning

✅ Tool-Calling

- Let LLM trigger external tools (e.g., calculator, DB, search)

✅ Feedback Loops

- Evaluate the LLM output and retry if it's low quality

🔚 Summary: Generation = Controlled Creativity

Step	What it Does
Format Prompt	Puts query + context into a prompt
Call LLM	Sends it to GPT or local model

Step	What it Does
Tweak Params	Adjusts temperature, tokens, etc.
Post-process	Cleans up and displays output