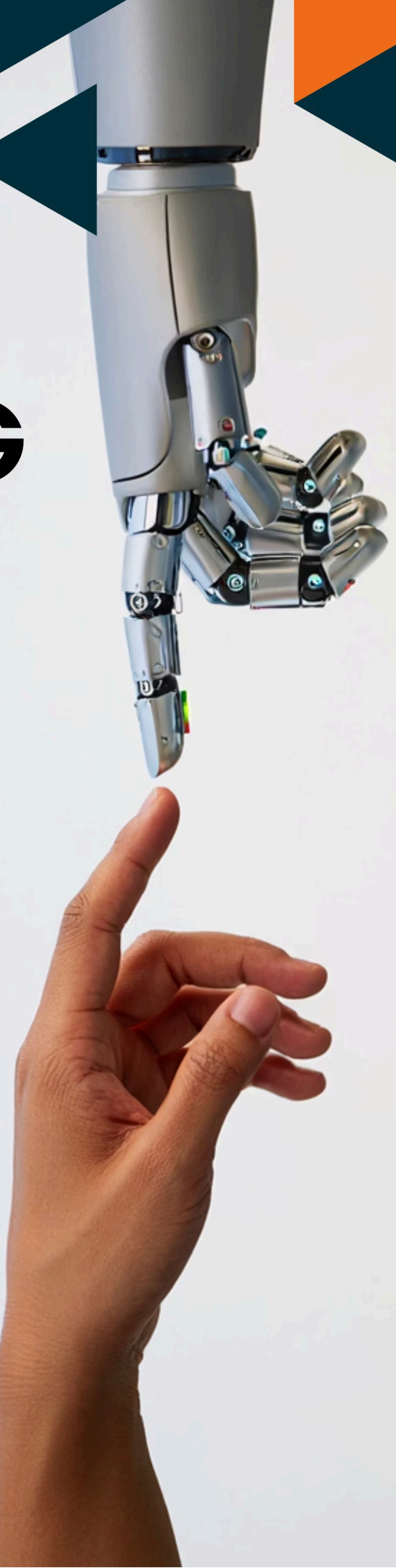
GSDC
Global Skill Development Council

PROMPT ENGINEERING

Interview Preparation Guide



www.gsdcouncil.org

Key Responsibilities in a Prompt Engineering Role

1

Designing and Refining Prompts

Designing and refining prompts for LLMs and generative AI tools.

2

Optimizing Outputs

Optimizing prompts to reduce hallucinations, bias, and irrelevant outputs.

3

Creating Templates

Creating reusable prompt templates and libraries.

4

Evaluating Performance

Evaluating prompt performance using quantitative and qualitative methods.

5

Team Collaboration

Collaborating with developers, data scientists, and product teams to integrate AI solutions.

Core Concepts to Master

Before an interview, ensure you have a deep understanding of these fundamentals:

Basics of LLMs

- How LLMs work: token prediction, probabilities, and pattern recognition.
- Limitations: hallucinations, context length limits, and lack of real-world awareness.
- Applications: chatbots, summarization, coding, data analysis, content generation.

Prompt Components

Understand each part of a prompt:

- Instruction: Clear task definition.
- Context: Background or audience information.
- Input Data: Text, code, tables, or structured data.
- Constraints: Style, tone, length, or prohibited content.
- Output Format: Structured response (lists, JSON, tables).

Prompt Techniques

- Zero-shot, one-shot, few-shot prompting.
- Chain-of-thought / step-by-step reasoning.
- Role-based prompting.
- Task decomposition.
- Retrieval-augmented prompting.

Prompt Patterns

- Instruction + Input
- Instruction + Example + Task
- Evaluation and scoring patterns
- Comparison / Analysis patterns

Debugging & Optimization

- Iterative testing of prompts.
- Reducing hallucinations.
- Managing style, tone, and consistency.
- Handling ambiguous or conflicting input.

Common Interview Questions: Fundamentals

Q: What is prompt engineering, in simple terms?

Prompt engineering is basically how you talk to an AI to get the best possible result. Instead of just asking a vague question, you structure instructions, context, and constraints so the model understands exactly what you want. In practice, it's less about tricks and more about clear thinking and communication.

Q: Why do companies care so much about prompt engineering?

Because the same AI model can give very different results depending on how it's prompted. Good prompts save time, reduce errors, and make AI outputs reliable enough for business use. It's often faster and cheaper than retraining or fine-tuning a model.

Q: What usually goes wrong when prompts don't work well?

Most of the time, the prompt is just too vague or assumes the AI "knows what I mean." Another common issue is asking too many things at once without prioritizing. When prompts fail, it's usually a communication problem, not a model problem.

Q: What's the biggest misconception about prompt engineering?

That it's just about clever wording. In reality, it's about system thinking, testing, and aligning AI behavior with real-world constraints. The best prompts often look simple.

Q: How do you test prompts before deploying them?

I test against common, edge, and adversarial inputs. I also compare outputs across multiple runs to check stability. If the prompt behaves predictably under stress, it's ready for deployment.

Q: Tell me about a time a prompt failed. What did you do?

The output was inconsistent across similar inputs. I realized the prompt assumed too much context, so I broke it into smaller steps and added explicit formatting rules. After iterating, the outputs became predictable and production ready.

Common Interview Questions: Techniques

Q: How do you usually start writing a prompt?

I start by clearly defining the goal of the output. Then I add context, specify the role I want the AI to take, and explain the format I expect. I think of it like briefing a teammate, not giving a command to a machine.

Q: Can you explain zero-shot and few-shot prompting with an example?

In zero-shot prompting, I might say, "Summarize this article in three bullet points," without any example. In few-shot prompting, I'd first show one or two example summaries so the model understands my style. Few-shot works better when tone or structure really matters.

Q: When would you use role-based prompting?

I use it whenever tone or expertise matters. For example, asking the model to respond "as a senior product manager" versus "as a beginner-friendly tutor" changes the depth and language significantly. It's a simple way to guide responses without over-explaining.

Q: What's chain-of-thought prompting, and do you actually use it?

Yes, especially for logic-heavy tasks. Chain-of-thought prompting encourages the model to think step by step before answering. It's useful for reasoning problems, but I'm careful with it in production to avoid unnecessary verbosity.

Common Interview Questions: Optimization & Debugging

Q: How do you improve a prompt that gives weak results?

I treat it as an iterative process. I look at what went wrong—was it unclear, missing context, or too open-ended? Then I refine one element at a time instead of rewriting everything.

Q: How do you handle hallucinations in AI responses?

First, I try to ground the prompt with provided data or references. I also explicitly tell the model not to guess and to say "I don't know" when information is missing. In production, I usually combine prompts with retrieval systems to reduce this risk.

Q: What is retrieval-augmented generation (RAG), in your own words?

RAG is when the model doesn't rely only on its training data. Instead, it pulls relevant information from a trusted source and uses that to answer. It's one of the most effective ways to improve accuracy in real-world applications.

Q: How do you test whether a prompt is "good enough" for production?

I test it across multiple inputs and edge cases, not just one happy path. I also check for consistency and failure behavior. If it breaks gracefully and stays predictable, it's usually ready.

Common Interview Questions

Q: What is prompt injection, and why is it dangerous?

Prompt injection is when users manipulate inputs to override system instructions. It's dangerous because it can leak data or change system behavior. To prevent it, I use strict system prompts and input validation.

Q: How is prompt engineering different from fine-tuning?

Prompt engineering changes behavior through instructions, while fine-tuning changes the model itself. Prompting is faster, cheaper, and more flexible. Fine-tuning is better when behavior needs to be deeply embedded.

Q: What skills make someone good at prompt engineering?

Clear communication is the biggest one. You also need logical thinking, experimentation, and some domain knowledge. Honestly, the best prompt engineers think like good teachers or product managers.

Q: Where do you see prompt engineering going in the future?

I see prompt engineering evolving from a tactical skill into a core part of AI interaction and system design. Instead of manually writing prompts, professionals will design dynamic prompt systems that adapt based on user behavior, context, and data. In fields like healthcare, finance, and legal, prompt engineering will focus on accuracy, compliance, and risk control, while in marketing, product, and media it will drive personalization, creativity, and scale.

Over time, the role will expand into areas like AI governance, workflow automation, and human-AI collaboration, where prompts act as decision frameworks rather than simple instructions. Essentially, prompt engineering will become a cross-functional discipline, similar to UX or system architecture, influencing how AI behaves across every industry rather than just how it responds to a single query.

Technical Q&A: Core Concepts

Q: Explain the difference between zero-shot and few-shot prompting.

Zero-shot prompting: The AI is asked to perform a task without any examples. It relies entirely on its pre-trained knowledge. This is best for straightforward or general tasks.

Example: "Translate the following sentence into French."

Few-shot prompting: The AI is given a few examples of the task along with the instructions. This helps the model understand the expected format, style, or reasoning process, improving accuracy for complex tasks.

Example: Showing 3 examples of question-answer pairs before asking a new question.

Q: How do you reduce hallucinations in LLM outputs?

- Provide source material so the model can base its output on facts.
- Use explicit instructions like "Only use the information provided."
- Ask the AI to respond with "I don't know" if the answer is uncertain.
- Limit the scope of the task to reduce open-ended interpretation.
- Use retrieval-augmented prompting by providing verified external content.

Q: What are the limitations of LLMs, and how do prompts mitigate them?

Limitations of LLMs:

- May produce hallucinations (confident but incorrect responses).
- Limited memory/context window; cannot access real-time data.
- Sensitive to wording; slight changes can affect output.

Mitigation through prompts:

- Clear instructions reduce ambiguity.
- Context and examples improve understanding.
- Constraints and output formats ensure usable, controlled responses.

Technical Q&A: Writing & Refinement

Q: Write a prompt to summarize a technical article for a non-technical audience.

```
You are a technical writer.

Your task is to summarize the following technical article in simple language suitable
for a non-technical audience.

Context:
The readers have no prior knowledge of the topic.

Input:
[Insert technical article here]

Constraints:
- Limit to 5 bullet points.
- Avoid jargon and explain complex terms simply.
- Keep each bullet under 25 words.

Output Format:
- Bullet points summarizing key insights.
```

Q: Refine a poorly performing prompt for clarity and consistency.

Original prompt (vague): "Explain AI."

Refined prompt:

```
You are an AI educator.
Explain artificial intelligence in 3 short paragraphs suitable for high-school students.
Focus on applications in everyday life and avoid technical jargon.
```

Why it works: Adds audience, format, and focus; reduces ambiguity and improves readability.

Technical Q&A: Output Formatting & Safety

Q: Design a prompt that generates JSON-formatted product reviews.

```
You are an e-commerce content generator.

Generate product reviews based on the following information.

Input:
Product name, features, user experience notes.

Constraints:
- Generate 3 reviews per product.
- Each review must be 2-3 sentences.
- Include sentiment (positive, neutral, negative).

Output Format:
{
  "product_name": "string",
  "reviews": [
    {
      "review_text": "string",
      "sentiment": "string"
    }
  ]
}
```

Q: How would you design prompts for a chatbot that handles sensitive user data?

- Clearly define roles and rules to prevent sharing sensitive data.
- Include constraints like "Do not output personal identifiers."
- Add context for safe behavior: "Respond professionally, do not ask for sensitive info."
- Test prompts with edge cases to ensure safe responses.

Technical Q&A: Testing & Domain Adaptation

Q: How would you test and evaluate prompt performance for a multi-step reasoning task?

- Break the task into smaller steps and create prompts for each.
- Evaluate outputs for accuracy, completeness, and consistency.
- Use step-by-step reasoning prompts to ensure logical processing.
- Iteratively refine prompts based on observed errors or hallucinations.

Q: How would you adapt prompts for a domain with specialized terminology (e.g., medicine, law, finance)?

- Assign the role of an expert in the field. Example: "You are a licensed medical professional."
- Provide context and definitions for any technical terms.
- Include constraints to ensure precise, domain-appropriate language.
- Use few-shot examples to teach the AI expected style and format.