

SKILLNAVORA · AI PRODUCT MANAGEMENT & GENAI
LAUNCHPAD · SKILL 1 OF 7

Prompt & Context Engineering

Make every AI request precise and evidence-based: clear instructions, the right context, structured outputs, task decomposition, long-context compression, reading documents and scans, citing sources and organising knowledge.

Edition 2026 · skillnavora.in

How to use this guide

Each class has goals, concepts with Indian examples, key takeaways and a practice exercise, from basic to advanced. Finish with the final project to build something you can show in interviews.

	Class	Level
Class 1	From prompts to context	Basic
Class 2	Prompt patterns that work	Basic
Class 3	Structured outputs: JSON, tables and fixed sections	Intermediate
Class 4	Task decomposition and chaining	Intermediate
Class 5	Long context: compression, summaries and memory	Intermediate
Class 6	Documents, PDFs, scans and OCR	Intermediate
Class 7	Source attribution, fact-checking and knowledge graphs	Advanced
Class 8	Context engineering for products and agents	Advanced
Toolkit: templates, patterns cheat sheet, glossary, 10 interview questions, further reading		

From prompts to context

See why the information you give the AI matters more than clever wording.

Prompt vs context

Prompt engineering	Context engineering
The words of your request: task, role, format, tone, examples	Everything the model sees: system instructions, documents, data, chat history, memory, tool results
"Write a PRD for a bill-split app"	...plus your user-interview summary, metrics, constraints, PRD template and two example PRDs

The six building blocks of a strong request

1. **Goal:** what you need and why ("so my VP can decide by Friday").
2. **Audience:** who will read it.
3. **Context:** facts, documents and data the AI must use.
4. **Constraints:** length, scope, what to avoid, rules.
5. **Format:** sections, table, JSON, slide outline.
6. **Quality bar:** "cite the source for every number; say 'unknown' if not in the documents".

Before and after

Weak: Summarise this survey.

Strong: You are helping a PM at a food-delivery startup in Pune.

Goal: find the top 3 reasons users cancel orders, for a roadmap review.

Use ONLY the attached survey (412 responses). Quote 2 responses per reason.

Output: a table (reason, % of mentions, quotes), then 3 recommendations.

If something isn't in the data, write "not in data".

Evidence-based requests ("use only these sources, cite each claim, flag gaps") are the single best defence against hallucinations.

Key takeaways

- Context (documents, data, examples) usually matters more than wording.
- Goal, audience, context, constraints, format, quality bar.
- Ask for citations and 'not in data' when unsure.

Practice exercise: Rewrite three of your recent AI requests using the six building blocks, and compare the answers.

Prompt patterns that work

Use proven patterns: examples, step-by-step, roles, critique and iteration.

Pattern	When	Example
Few-shot examples	You want a specific style or format	Give 2 good user stories, then ask for 10 more like them
Step-by-step	Reasoning, maths, analysis	"First list assumptions, then calculate, then give the answer"
Role and audience	Tone and depth	"Explain to a non-technical CFO in 120 words"
Delimiters	Separate instructions from data	Put documents inside <doc>...</doc> tags
Critique and revise	Higher quality drafts	"Review your draft against this checklist, then improve it"
Ask me questions first	Ambiguous tasks	"Before writing, ask me up to 5 questions"
Negative constraints	Avoid common mistakes	"No jargon; don't invent metrics"

Iterate like a PM

Treat prompts like product: version them, test them on 5–10 examples, keep what works. Save good prompts as templates (Class 3 of the AI Skills & Templates course).

Key takeaways

- Show examples of what good looks like.
- Separate instructions from data with clear delimiters.
- Version and test prompts like product features.

Practice exercise: Create a few-shot prompt that turns raw customer complaints into well-formed user stories; test it on 10 complaints.

Structured outputs: JSON, tables and fixed sections

Get predictable output other tools and people can use directly.

Why structure

Structured output can be pasted into Sheets, parsed by code, compared across runs and graded automatically (evals).

Ways to get it

- **Describe the schema** in the prompt with an example.
- **Structured-output / JSON modes** in model APIs that enforce a schema.
- **Fixed document sections** for PRDs and reports: "Use exactly these headings".

```
Return ONLY JSON matching:  
{ "theme": string, "mentions": integer, "severity": "low" | "medium" | "high",  
  "quotes": [string, string], "source_rows": [integer] }  
One object per theme, in an array, sorted by mentions.
```

Checks

- Validate the JSON (does it parse? required fields present?).
- Spot-check values against the source.
- Keep field names stable across versions so dashboards don't break.

Key takeaways

- Structure makes outputs reusable and testable.
- Give a schema plus an example.
- Validate and spot-check structured output.

Practice exercise: Turn 30 app-store reviews into JSON themes using a schema, then load the JSON into a Google Sheet.

Task decomposition and chaining

Break big jobs into steps the AI can do reliably.

Decompose

One giant prompt ("research the market, write a PRD and make slides") gives shallow results. Break it into steps, each with a clear input and output:

1 Extract facts from sources → 2 Synthesise insights → 3 Frame the problem → 4 Draft PRD sections → 5 Check against rubric → 6 Make slide outline

Rules for good steps

- Each step has one job and a checkable output.
- Pass only what the next step needs (keeps context small and focused).
- Add a review step (human or AI checklist) before anything goes to stakeholders.

Where to run chains

Manually in chat, in Claude or ChatGPT Projects, with Skills (packaged steps), or automated in n8n, Make or code (see Agent Orchestration & Monitoring).

Key takeaways

- Big tasks → small steps with checkable outputs.
- Pass only needed context between steps.
- Insert review steps before sharing.

Practice exercise: Decompose 'prepare a competitor teardown deck' into 6 steps with inputs and outputs for each.

Long context: compression, summaries and memory

Keep long projects accurate when chats and documents get large.

Problems with long context

- Important details get lost in very long chats ("context drift").
- Cost and latency grow with every token sent.
- Old, wrong information keeps influencing answers.

Techniques

Technique	How
Running summary	Every 20–30 messages, ask for a summary of decisions, facts and open questions; start a fresh chat with it
Project brief	A 1-page brief (goal, users, constraints, decisions) pinned in a Project
Retrieve, don't paste	Keep documents in a Project/second brain and let retrieval pick relevant parts
Compression with care	Ask the summary to keep numbers, names, dates and decisions exactly
Memory hygiene	Review stored memories; remove outdated facts

Summarise this conversation for a fresh chat. Keep exactly: decisions, numbers, names, dates, open questions, and the current task. Max 250 words.

Key takeaways

- Long chats drift; summarise and restart.
- Pin a project brief; retrieve documents instead of pasting.
- Compression must keep numbers, names and decisions exact.

Practice exercise: Take a long chat from your work and write a 250-word handover summary; start a new chat with it and check nothing important was lost.

Documents, PDFs, scans and OCR

Get reliable information out of PDFs, slides, spreadsheets and images.

Document ingestion

- Modern assistants read PDFs, Word, slides and images directly; very long files may be chunked.
- Tables in PDFs are error-prone: ask for the table as CSV and check totals.
- **OCR** (optical character recognition) reads text from scans and photos: invoices, handwritten notes, whiteboards. Vision models can also describe charts.

Extraction prompt

From the attached invoice images, extract: vendor, GSTIN, invoice no., date (DD-MM-YYYY), items (name, qty, rate, amount), tax, total. Return CSV. If a field is unreadable, write UNREADABLE (don't guess). Then check: do item amounts add up to the total?

Accuracy habits

- Add arithmetic and cross-checks (totals, dates in range).
- Spot-check 10% by hand; track error rate (that's an eval).
- Mask personal data (Aadhaar, PAN, phone) before sharing documents with AI tools.

Key takeaways

- AI reads PDFs, slides and images; tables need checking.
- Ask for 'UNREADABLE' instead of guesses.
- Cross-check totals and spot-check samples.

Practice exercise: Extract data from 5 receipts or invoices (photos) into CSV and measure how many fields were correct.

Source attribution, fact-checking and knowledge graphs

Make AI outputs traceable to evidence and organise knowledge by relationships.

Attribution

- Ask for citations per claim: document name + page/section, or a quote.
- Use tools with built-in citations (NotebookLM, Claude citations, Perplexity) for research.
- Verify: open the cited place and check it actually supports the claim (see the AI Evals course on faithfulness).

Fact-check prompt

For each claim in the draft below, label SUPPORTED / NOT FOUND / CONTRADICTED using only the sources provided, and quote the supporting sentence.

Knowledge graphs in plain words

A knowledge graph stores **things** (people, products, features, customers) and **relationships** between them ("Feature X requested by Customer Y", "Metric A depends on Metric B"). It helps answer connected questions: "Which enterprise customers asked for features that depend on the payments API?"

- Start simple: a table of entities and a table of relationships in Notion or Sheets.
- AI can extract entities and relationships from notes and tickets.
- Graph tools (e.g. Neo4j) and "GraphRAG" approaches combine graphs with retrieval for complex questions.

Key takeaways

- Every important claim should point to evidence.
- Use SUPPORTED / NOT FOUND / CONTRADICTED checks.
- Knowledge graphs capture relationships that plain search misses.

Practice exercise: Build a mini knowledge graph (entities + relationships table) from 10 customer tickets and ask AI 3 connected questions.

Context engineering for products and agents

Design what your AI feature or agent sees, and test it.

Context design for an AI feature

Layer	Decide
System instructions	Role, rules, refusals, tone, format
User and session	Profile, language, permissions, history summary
Knowledge	Which sources are retrieved, how fresh, how many chunks
Tools	Which tools are available and their descriptions
Examples	Few-shot examples of ideal answers

Common context failures

- Too much irrelevant context (the model gets distracted).
- Missing a key fact (answers become generic or wrong).
- Conflicting instructions between system prompt and documents.
- Untrusted content (web pages, emails) treated as instructions: prompt injection.

Test it

Use an eval set (AI Evals course): vary the context (fewer chunks, more examples, shorter instructions) and measure quality, cost and latency. Pick the smallest context that meets the quality bar.

Final project

Write the context design for an AI feature (e.g. a Hinglish support bot): system prompt, retrieval plan, tools, 3 few-shot examples, a structured output schema, and a 20-case eval comparing two context versions.

Key takeaways

- Design every layer the model sees.
- Less but relevant context beats more context.
- Measure context changes with evals.

Practice exercise: Complete the final project for an AI feature of your choice.

Templates

A. Evidence-based request

Goal (and why):
Audience:
Use only these sources:
Constraints (length, scope, avoid):
Output format (sections / table / JSON schema):
Quality bar (cite each claim; write "not in data" if missing):

B. Handover summary

Decisions: · Facts and numbers (exact): · Names and dates: · Open questions: · Current task:

C. Context design sheet

Layer	Content	Size	Owner
System instructions			
User/session			
Knowledge (retrieval)			
Tools			
Examples			

Patterns cheat sheet

Term / tool	Meaning / use
Few-shot	Give examples of ideal outputs
Delimiters	Wrap data in tags or quotes to separate it from instructions
Chain-of-thought / step-by-step	Ask for reasoning steps before the answer
Structured output	JSON/table/sections with a schema
Task decomposition	Split work into checkable steps
Prompt chaining	Output of one step feeds the next
Context compression	Summaries that keep decisions, numbers and names
RAG	Retrieve relevant passages before answering
OCR	Read text from scans and images
Attribution	Link each claim to a source
Knowledge graph	Entities + relationships for connected questions
Prompt injection	Untrusted content trying to act as instructions

Glossary

Context window	All text the model can see at once
System prompt	Standing instructions for the AI's role and rules
Schema	Definition of fields and types for structured output
Chunk	A piece of a document used for retrieval
Citation	Pointer from a claim to its source
GraphRAG	Retrieval that uses a knowledge graph

10 interview questions with answers

Q1. Prompt vs context engineering?

Prompting is the wording of the request; context engineering is designing all information the model sees: instructions, documents, data, memory, tools and examples.

Q2. How do you reduce hallucinations?

Provide sources, restrict answers to them, ask for citations and 'not in data', use retrieval, and verify with fact-check evals.

Q3. Why use structured output?

It makes outputs predictable, machine-readable, comparable and easy to grade; useful for dashboards, automation and evals.

Q4. What is task decomposition?

Breaking a complex task into smaller steps with clear inputs and outputs, often chained, with review steps.

Q5. How do you handle very long chats?

Summarise decisions and facts, start fresh with the summary, pin a project brief and retrieve documents instead of pasting everything.

Q6. How do you extract data reliably from PDFs or scans?

Ask for a fixed schema, 'UNREADABLE' instead of guessing, arithmetic cross-checks, and measure accuracy on a sample.

Q7. What is a knowledge graph?

A store of entities and their relationships that answers connected questions plain keyword or vector search may miss.

Q8. How do you decide how much context to give?

Test variants against an eval set and choose the smallest context that meets quality, cost and latency bars.

Q9. What is prompt injection?

Instructions hidden in untrusted content (web pages, emails, documents) that try to override yours; treat such content as data and limit tool permissions.

Q10. Give an evidence-based prompt for a PM task.

Goal, audience, 'use only the attached survey', quote two responses per theme, table output, and 'not in data' for gaps.

Further reading and tools

- Anthropic: prompt engineering overview: <https://docs.claude.com/en/docs/build-with-claude/prompt-engineering/overview>
- Anthropic: effective context engineering for AI agents: <https://www.anthropic.com/engineering/effective-context-engineering-for-ai-agents>
- OpenAI: structured outputs: <https://platform.openai.com/docs/guides/structured-outputs>
- Google: prompting guide (Gemini): <https://ai.google.dev/gemini-api/docs/prompting-strategies>
- Prompting Guide: <https://www.promptingguide.ai/>
- Neo4j: what is a knowledge graph: <https://neo4j.com/blog/what-is-knowledge-graph/>

© SkillNavora · skillnavora.in · Part of the AI Product Management and GenAI Launchpad programs. Practise with the free AI Tutor, AI PM Toolkit and Practice Lab on skillnavora.in.