

Key concepts for LLMs and AI agents: the shared vocabulary your first project needs

A one-hour session for mixed project teams before writing the first prompt or line of code

FRAMING

Agree on ten concepts before kickoff, or re-litigate them in change-requests later

SITUATION

Our team is preparing its first AI project. Managers own scope, risk and budget; engineers own the build. Both sides have read vendor material and feel ready.

COMPLICATION

The same words mean different things across the table. 'Fine-tuning' is rarely the right first move; 'token' is a billing unit to one person and a model internal to another. These gaps become change-requests once the build starts.

RESOLUTION

Spend one hour now aligning on ten concepts — each tied to a project decision — so scope, cost and acceptance criteria rest on shared ground before a single line of code is written.

One concept per slide, each tied to the project decision it drives

1	How an LLM works: tokens and training shape what the model holds and when costs land	Slides 4–5 cover the context window and training versus inference — the two facts that govern prompt design and per-request budgeting.
2	How you shape its behaviour: retrieval, prompting and hallucination controls are your levers	Slides 6–8 cover RAG, fine-tuning versus prompting, and hallucination mitigation — the decisions that determine data pipeline design and quality controls.
3	How you extend it into an agent: function calls and loops turn answers into actions	Slides 9–10 cover function calling and the agent plan-act-observe loop — the decisions that open your internal APIs and set automation boundaries.
4	How you ship it responsibly: evaluation, guardrails and economics decide what goes live	Slides 11–12 cover test-set evaluation, guardrails, cost and latency — the gate criteria every AI feature must clear before launch.

The context window caps what the model reads at once, so it caps your document and prompt design

~4 chars

A token is roughly four characters of English text

Models read tokens, not words. One token is about four characters in English. A 1,000-word document is roughly 1,300 tokens. This determines how much text fits in a single request.

8K–1M

Context windows span 8K to 1M tokens across vendors

GPT-4o: 128K; Claude: 200K; Gemini: 1M tokens. The window holds your system prompt, retrieved docs and question. Larger windows enable bigger documents and longer histories.

Overflow forces a choice: chunk, summarise or retrieve

When content exceeds the window, choose: compress via summarisation, split into chunks, or retrieve only relevant passages. This choice is the first architectural decision in your data pipeline.

**Training is a sunk cost you rarely pay;
inference is the per-request bill you always pay**

One-off

**Training is a one-off cost
already paid by the vendor**

Frontier models are trained on trillions of tokens at a cost of tens of millions of dollars (Stanford HAI AI Index 2024). That cost is built into vendor pricing — your team does not re-run training on a typical project.

\$/token

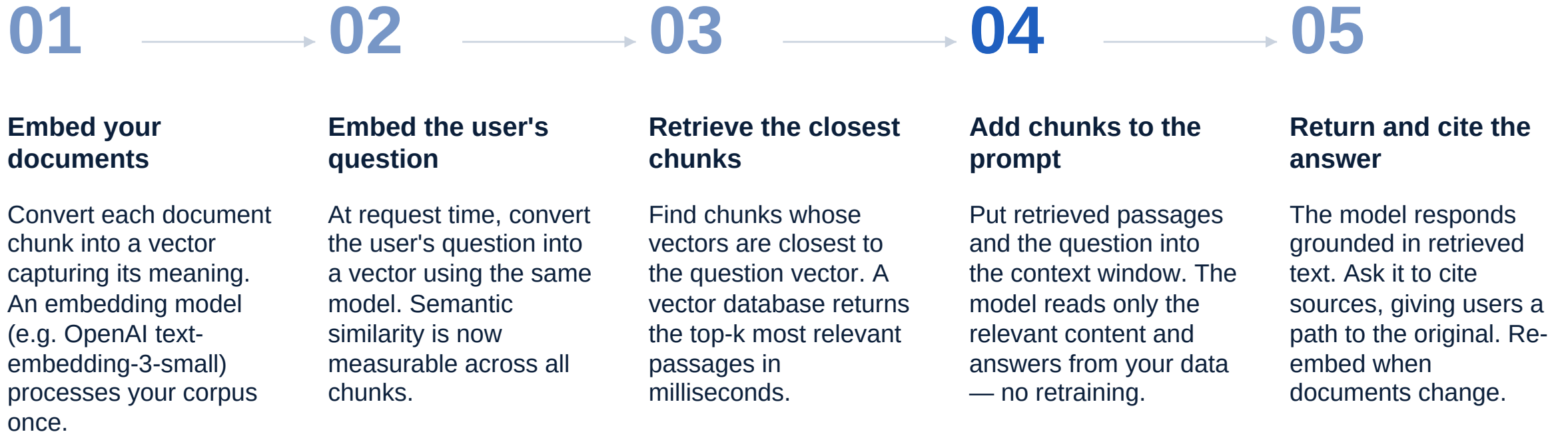
**Inference bills on every API
call you make**

You pay per token — input and output priced separately — on every user request. GPT-4o and Claude both operate this way (OpenAI, Anthropic pricing pages; vendor-dependent). Volume drives cost.

**Budget for inference, not
training**

Retrieving your data at request time (RAG) is faster, cheaper and easier to update than fine-tuning. Model your inference cost: tokens per request times volume times per-token price.

Retrieval, not re-training, is the default way to put your data in front of the model



Data pipeline

Application

Application

Application

Application

Source: Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks (2020) ; OpenAI platform documentation — models and pricing (2024) ; Hugging Face fine-tuning and embeddings documentation (2024)

Try prompting first; fine-tune only when prompting plateaus on a stable task

	RECOMMENDED Prompting Instructions in text only	RAG Retrieval adds live data to each prompt	Fine-tuning Model weights updated on your examples
What it changes	Instructions and examples in text	Adds relevant docs at request time	Model weights retrained on labelled pairs
Setup time	Hours to days	Days to weeks (pipeline build)	Weeks (labelling + training run)
Update cost when data changes	Edit a text file; instant	Re-embed changed documents	Re-run training; high cost
When to use	Always try first	Data changes; facts must stay current	Prompting plateaued; stable high-volume task

Verdict: Start with prompting; add RAG when you need live data; fine-tune only on a stable proven task

Source: OpenAI platform documentation — models and pricing (2024) ; Hugging Face fine-tuning and embeddings documentation (2024)

Treat confident wrong answers as a product risk and mitigate with retrieval, structure and checks

Ground answers in retrieved sources

RAG constrains the model to facts in the retrieved text. Ask it to cite passages. If no relevant source is found, the system should say so rather than improvise. Grounding is the most effective single mitigation.

Constrain the output with schemas

Structured outputs (JSON schema, enum choices) remove space for invented values. Supply allowed labels for classification; supply field names and types for extraction. The model cannot hallucinate what it cannot choose.

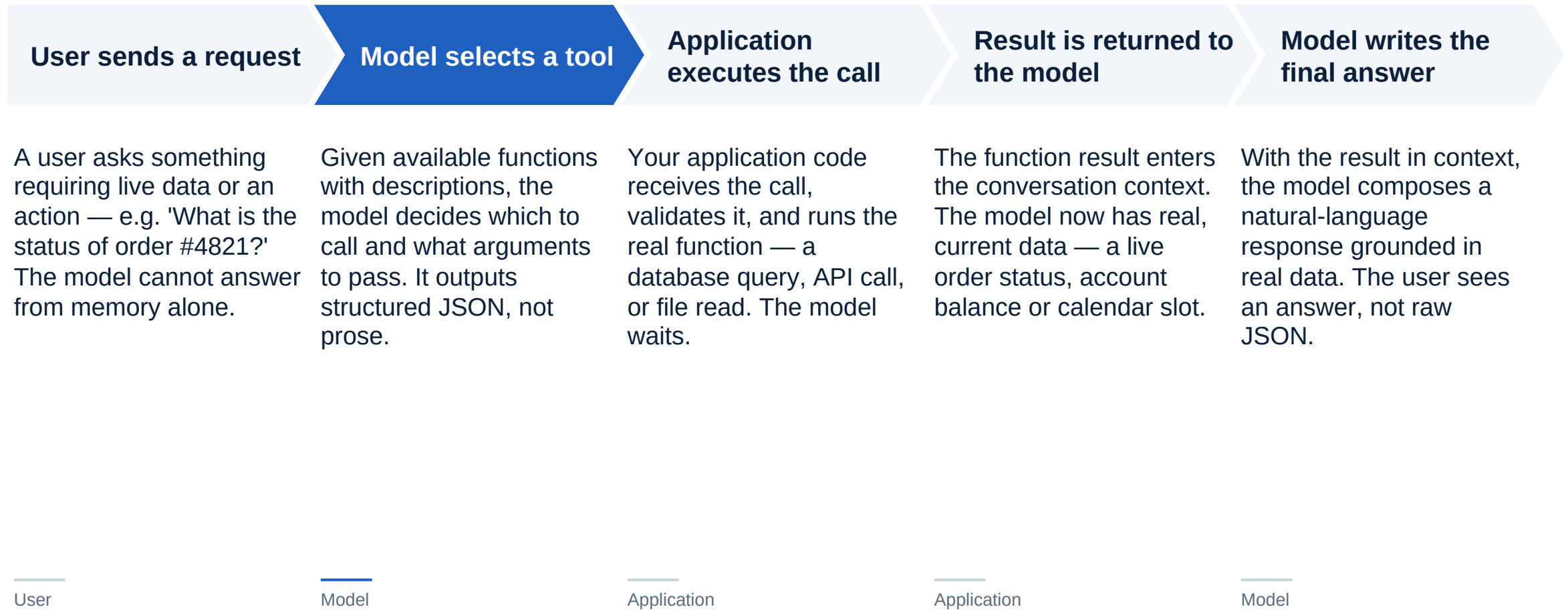
Check with a second pass or external lookup

For high-stakes outputs, a second model call asks: 'Is this answer supported by the provided sources?' Validate extracted values against a database. Adds cost and latency but is right for regulated decisions.

Surface uncertainty honestly in the UX

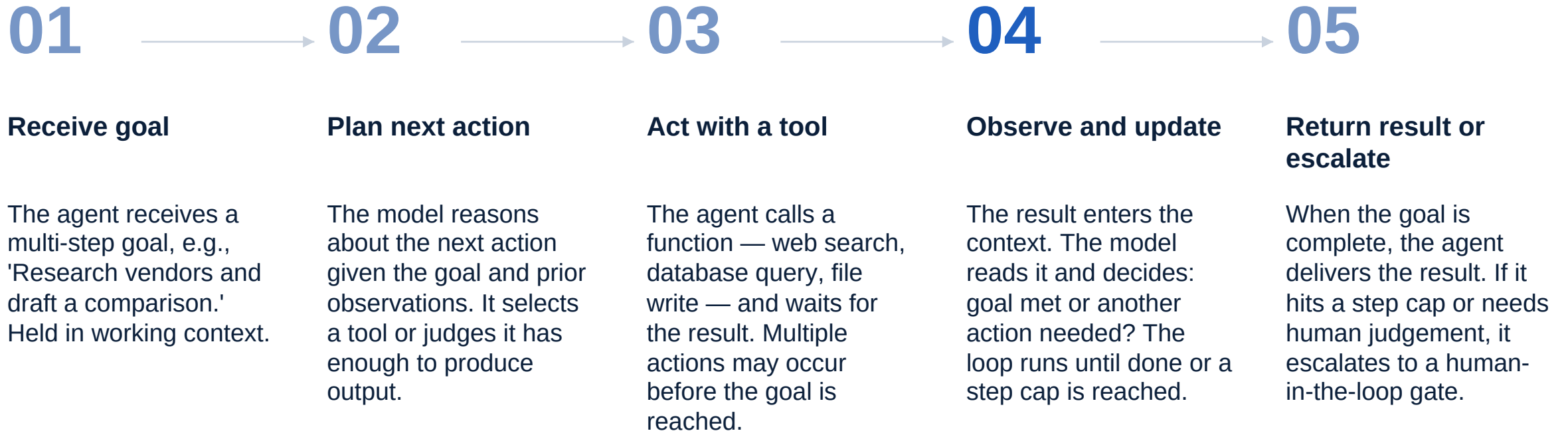
When confidence is low, show it. Labels like 'not found in our documents' or 'consult a specialist' shift accountability appropriately. Hiding uncertainty is a product design choice that amplifies risk.

Function calling turns the model from a writer into an actor on your systems



Source: OpenAI platform documentation — models and pricing (2024) ; Anthropic Claude model documentation (2024)

An agent is a loop that plans, acts and checks itself; memory decides how far it can go



If you cannot measure quality on a fixed test set, you cannot ship or improve the system

Build a frozen test set before writing any prompts

Fixed Q&A pairs created before build are your only honest measure. Tuning on your test examples measures memorisation, not generalisation. Target 50–200 examples depending on task complexity.

Mix automated metrics with human review

Automated checks — exact match, semantic similarity, factual consistency — run fast in CI. Human review catches errors automation misses: tone, coherence, unsafe outputs. Both are needed; neither is sufficient.

Run the test set before every change, not just at launch

Prompt changes, model upgrades and new data can degrade quality silently. Treating the test set as a regression gate — like unit tests in software — catches regressions before they reach users.

Monitor production quality after launch

A static test set cannot cover the full distribution of real queries. Sample production outputs and route a fraction to human review. Track metrics over time; plan for model deprecations that force re-evaluation.

Guardrails, cost and latency are the three gates every AI feature passes before launch

DECISION REQUESTED

Confirm alignment on these ten concepts, assign owners for guardrails, cost and latency, and schedule the project kickoff

1

Define guardrails: prompt injection tests, PII masking, content policy thresholds

Product lead · Before kickoff

2

Set inference cost budget per request; configure usage alerts (44% cite cost unpredictability)

Engineering lead · Before kickoff

3

Agree latency SLA and identify caching or fallback strategy when the model is slow

Engineering lead · Before kickoff

4

Schedule pre-launch eval review against frozen test set and the three gate criteria

Project sponsor · Before launch

5

Assign on-call owner for post-launch monitoring of quality, cost and guardrail alerts

To be named · At launch

Source: OpenAI platform documentation — models and pricing (2024) ; Anthropic Claude model documentation (2024) ; McKinsey Global Survey on AI 2024 (2024) ; OWASP LLM Top 10 2025 (2025)