

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

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

STORYLINE
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.

COMPPLICATIONS
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.

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

PROBLEM
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

2 How you shape its behavior: retrieval, prompting and hallucination controls are your levers

3 How you extend it into an agent: function calls and loops turn answers into actions

4 How you ship it responsibly: evaluation, guardrails and economics decide what goes live

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

HOW YOU WORK
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

8K–1M
Context windows span 8K to 1M tokens across vendors

Overflow forces a choice: chunk, summarize or retrieve

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

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

One-off
Frontier models are trained on trillions of tokens at a cost of \$10s of millions of dollars (GPT-4o and Claude 3.5). 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 price separating — on every user request. GPT-4o and Claude both operate this way (OpenAI, Anthropic, pricing page; vendor dependent). Volume drives cost.

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

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

01 Embed your documents

02 Embed the user's question

03 Retrieve the closest chunks

04 Add chunks to the prompt

05 Return and cite the answer

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

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

	Prompting	RAG	Fine-tuning
What it changes	Instructions and examples in text	Adds relevant docs at request time	Model weights trained on labeled pairs
Setup time	Hours to days	Days to weeks (pipeline build)	Weeks (labeled + training)
Update cost when data changes	Edit a text file, insert	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

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

HOW YOU WORK
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. Confining to the most effective single mitigation.

Constrain the output with schemas
Structured outputs (JSON, schema, enum choices) reserve space for expected values. Supply allowed tokens for classification, supply fixed 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 related values against a database. Add risk and review. Put in sight 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. Flagging uncertainty in product design choices that amplifies risk.

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

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

User sends a request → Model selects a tool → Application executes the call → Result is returned to the model → Model writes the final answer

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

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

01 Receive goal

02 Plan next action

03 Act with a tool

04 Observe and update

05 Return result or exit-date

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

HOW YOU WORK
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 QA pairs created before build are your only honest measure. Tuning on your test examples measures memorization, not generalization. Target 50–200 examples depending on task complexity.

Mix automated metrics with human review
Automated checks — exact match, semantic similarity, factual consistency — not fact in C. 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 identity. 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 generations that force re-evaluation.

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

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

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

- Define guardrails: prompt injection tests, PII masking, content policy thresholds
- Set inference cost budget per request; configure usage alerts (40% clip cost unacceptability)
- Agree latency SLA and identify caching or fallback strategy when the model is slow
- Schedule pre-launch real review against frozen test set and the three gate criteria
- Assign on-call owner for post-launch monitoring of quality, cost and guardrail alerts

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