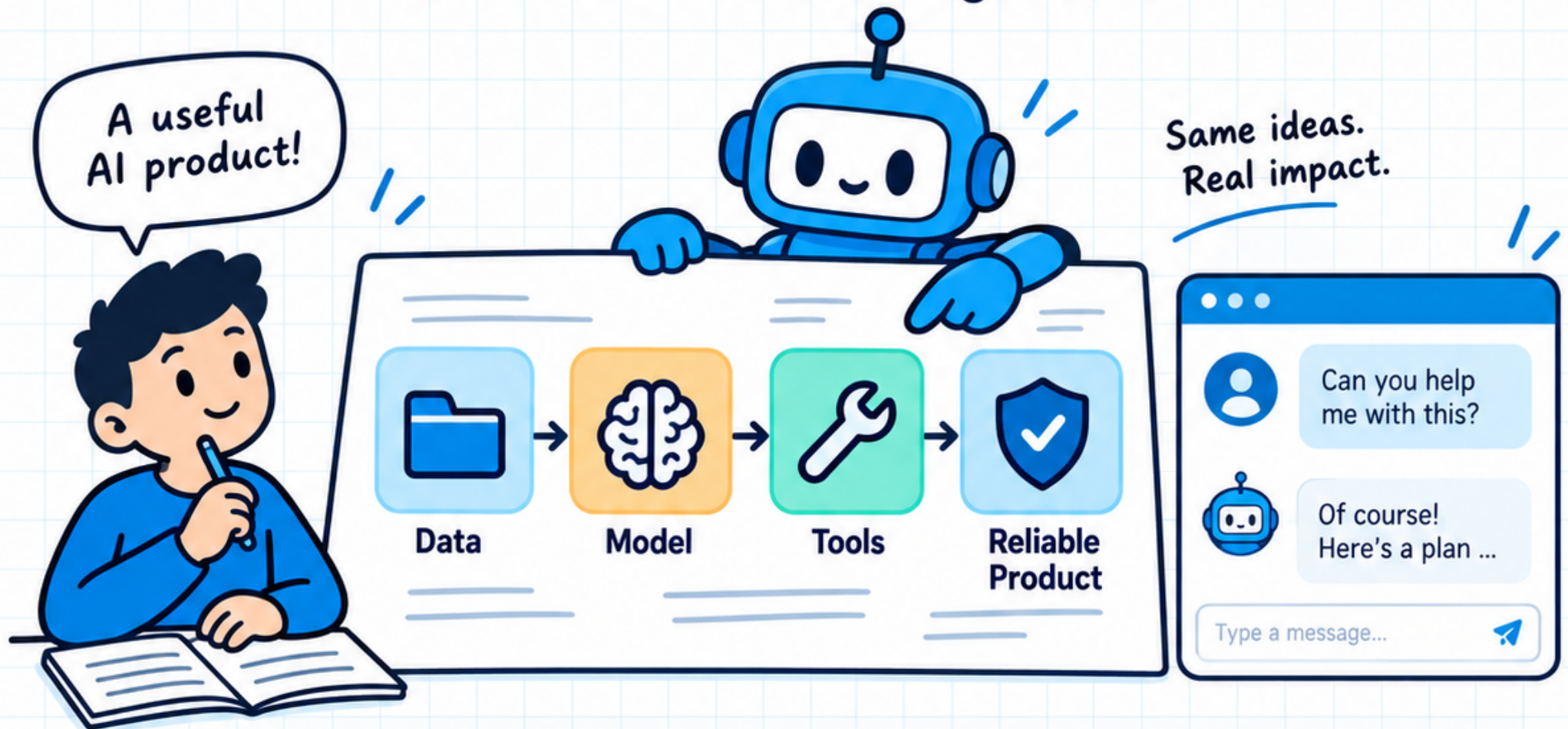


WHAT IS AI SYSTEM DESIGN?

From models to useful, reliable products

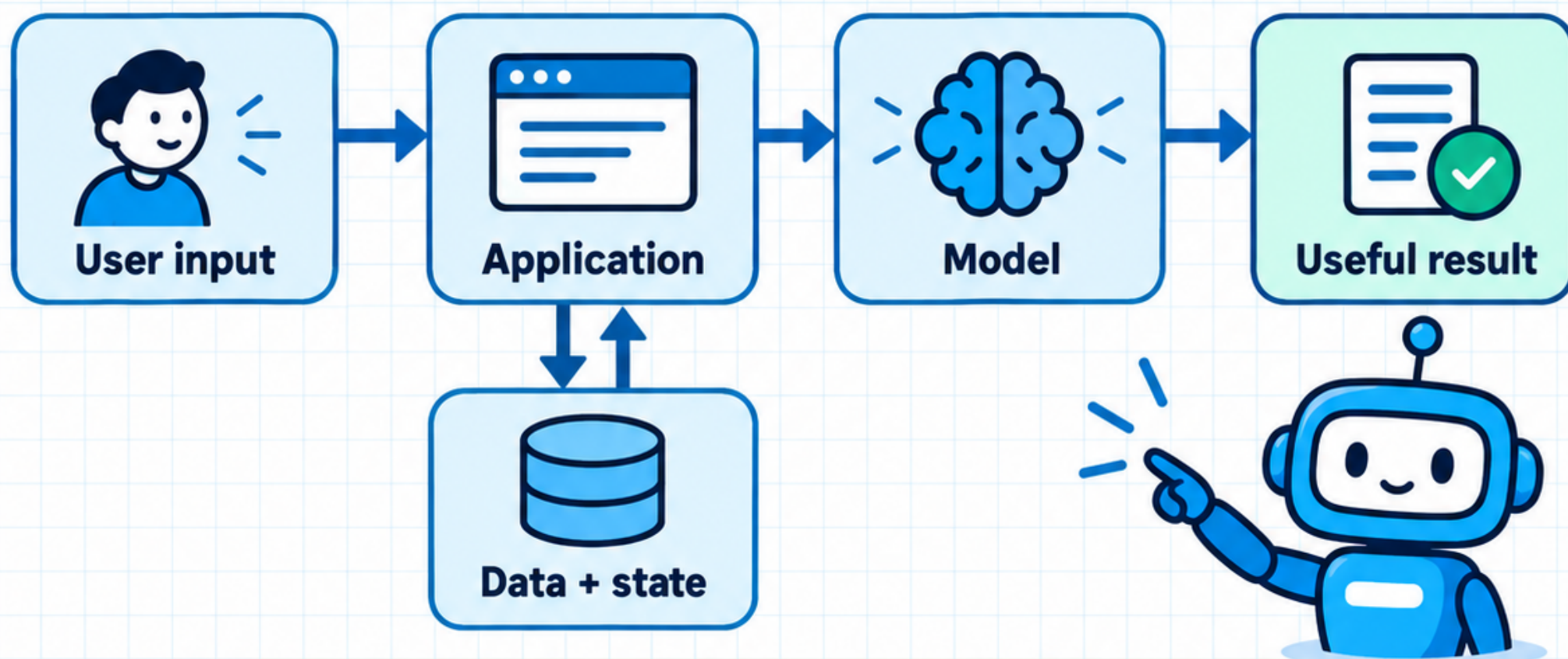
ML • LLMs • GenAI • Agentic AI



Learn how the pieces work together.

DESIGN THE WHOLE PRODUCT

Choose how data, models, software and people work together to meet a goal.



Access checks • failure handling • measurement



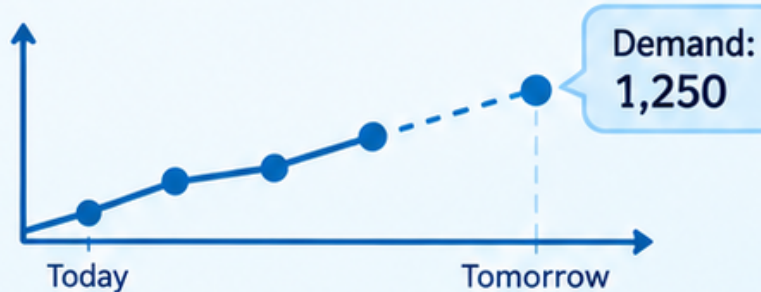
A strong model is one part of a working AI system.

FOUR IDEAS, DIFFERENT ROLES

These overlap. They are not four separate stacks.

PREDICTIVE ML

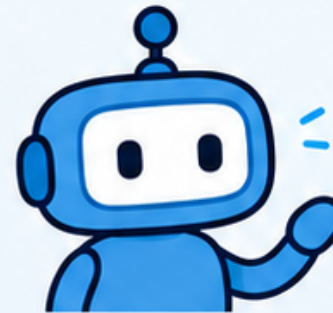
Estimate a score, class or number.



Example: forecast tomorrow's demand.

GENERATIVE AI

Create content such as text or images.



Hi!
Here's a draft answer
to your question ...

Example: draft a support answer.

LLM

A model for language tasks.



Often used inside GenAI applications.

AI AGENT

A system that chooses actions toward a goal.



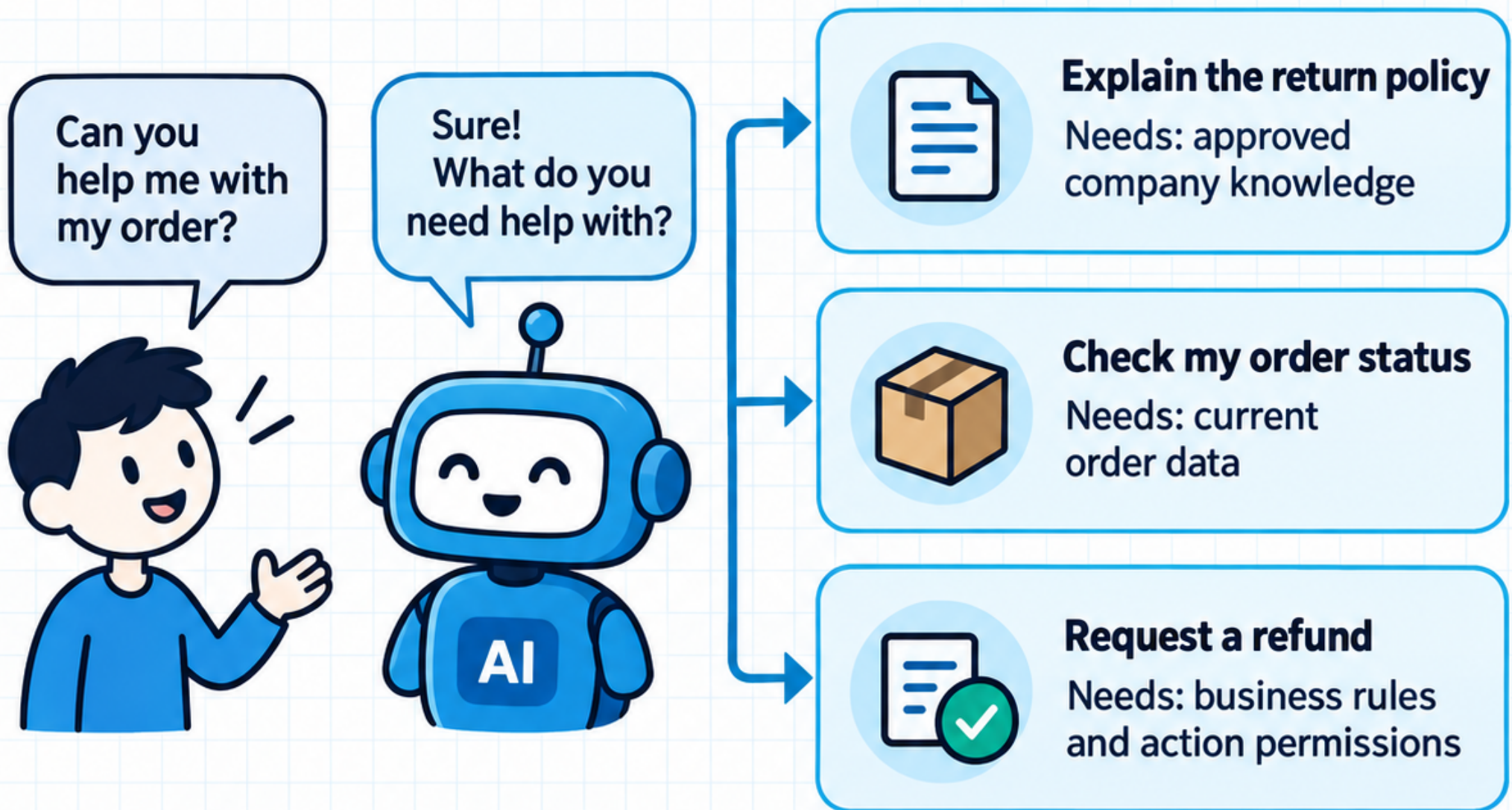
Example: investigate a delayed order.



An agent can use an LLM, tools and other ML models.

START WITH THE USER'S JOB

Example: a customer-support assistant



The job decides the data, tools and controls.

MATCH THE DESIGN TO THE TASK

Three small examples

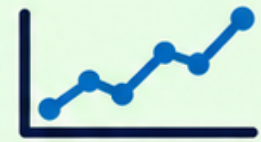
DEMAND FORECAST



Recent signals



Trained model



Predicted demand

POLICY ANSWER



Question +
retrieved policy



LLM



Answer

ORDER INVESTIGATION



Find out why
my order was
delayed.

Goal



Model and tool steps



Outcome

Each result can change the next step.

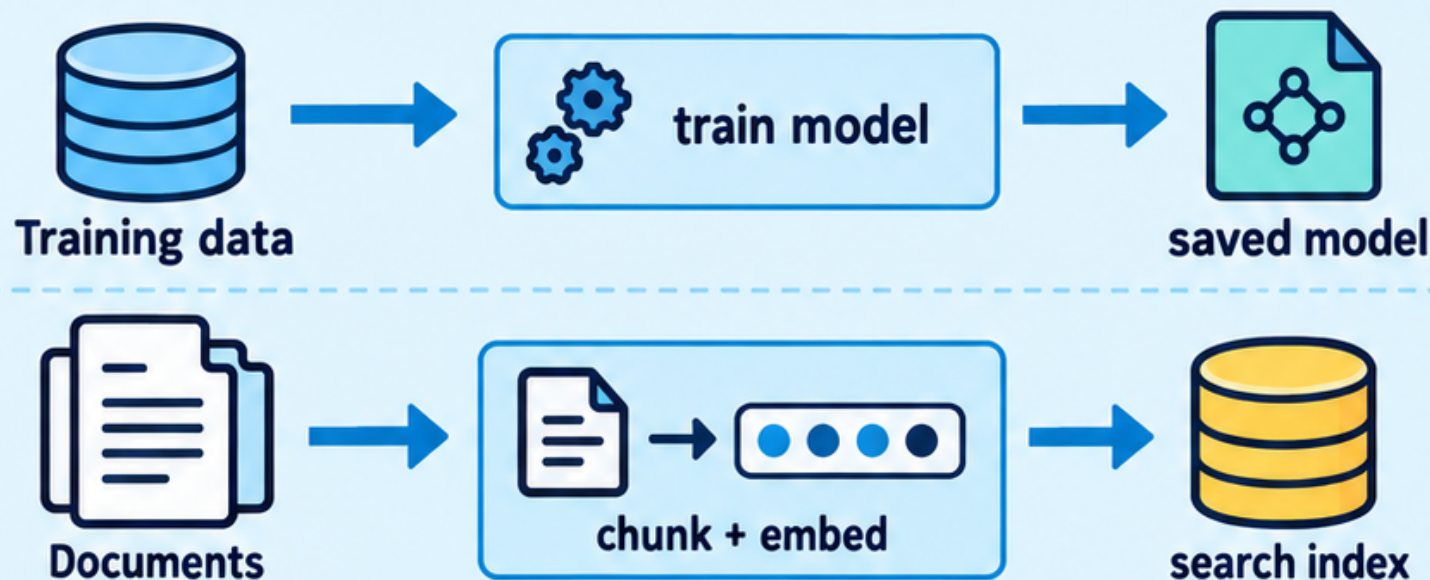


Reuse only the components the task needs.

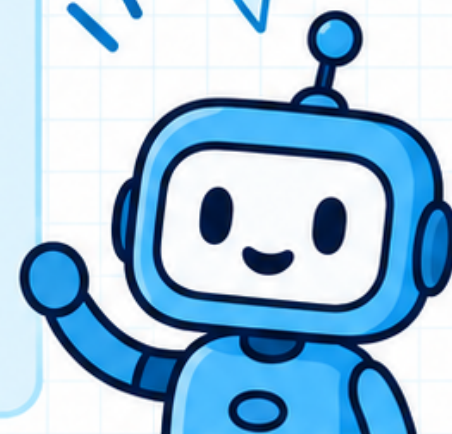
SEPARATE PREPARATION FROM SERVING

Some work happens before a user request.

PREPARATION

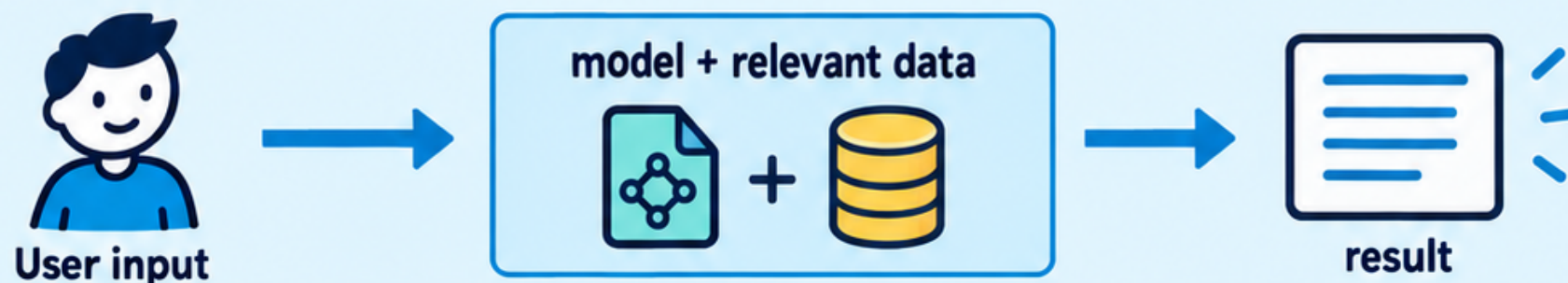


Do this ahead of time!



Preparation may run continuously in the background.

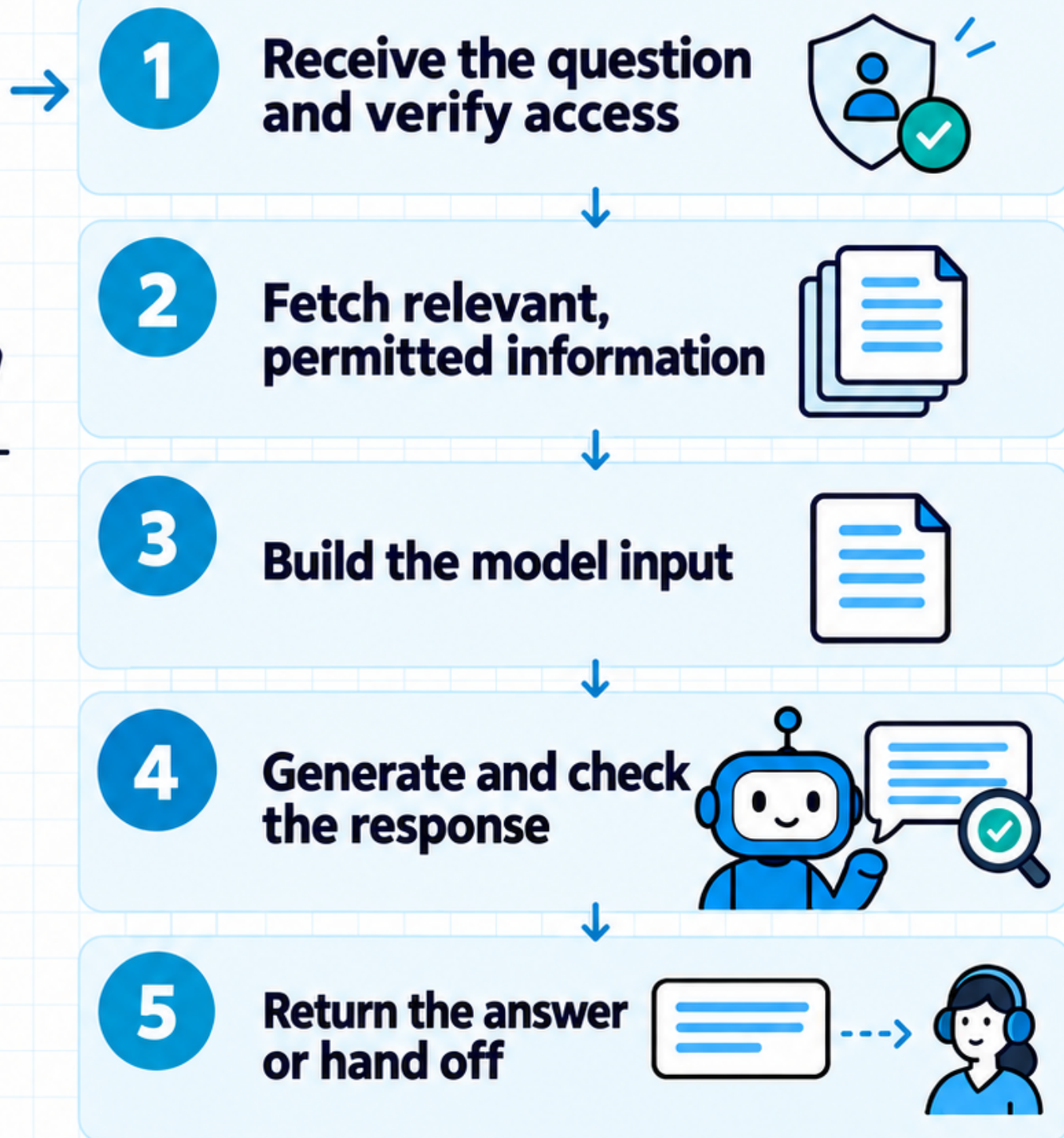
LIVE REQUEST



Indexing documents is not the same as training an LLM.

TRACE ONE REQUEST

Example: answering a company-policy question



Trace timing, errors and cost



Every step can affect quality, delay and reliability.

DATA HAS DIFFERENT JOBS

Know which source answers which need.

TRAINING DATA

Teaches model patterns.



Example: historical demand.

RETRIEVED CONTEXT

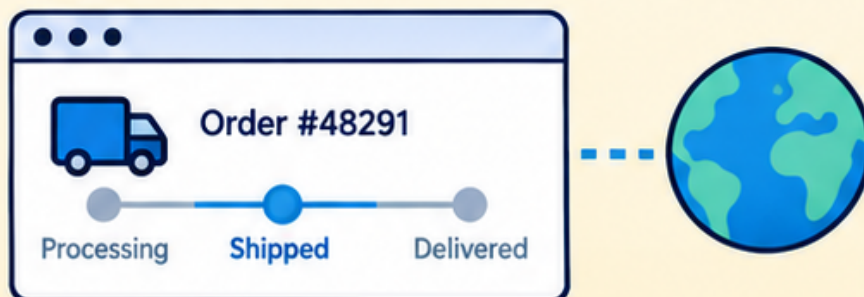
Supplies evidence at request time.



Example: the current return policy.

LIVE TOOL DATA

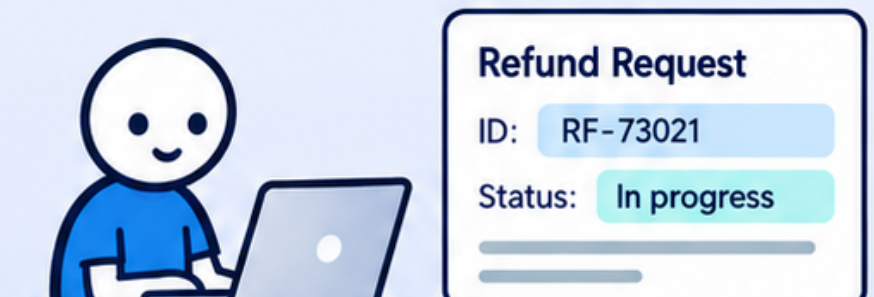
Reports the latest external state.



Example: an order's shipping status.

APPLICATION STATE

Tracks the current task.



Example: a saved refund-request ID.



Model weights are not a live company database.

RAG CONNECTS ANSWERS TO SOURCES

Retrieve evidence, then ask the model to use it.



Question:
What is the return window?



Toy policy:
Returns are accepted
within 30 days.



**Retrieve
allowed policy**



**LLM: question +
evidence**



**Answer with
source**



The policy says 30 days. [1]



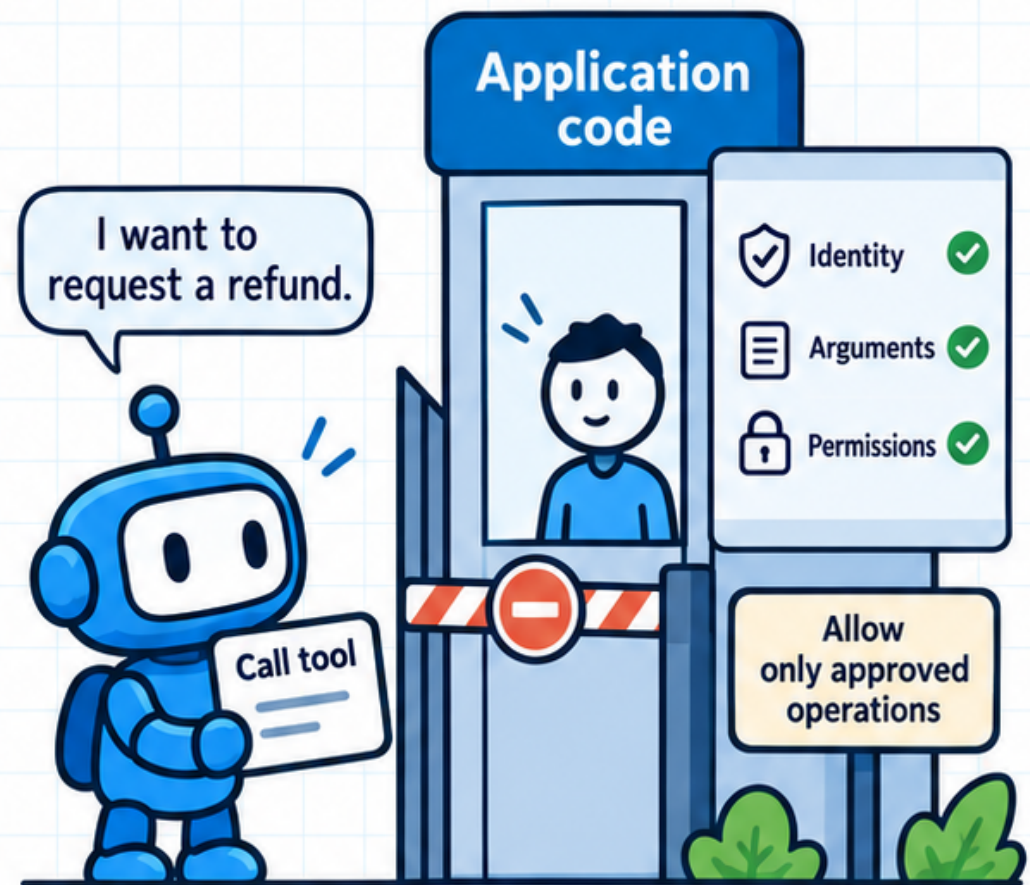
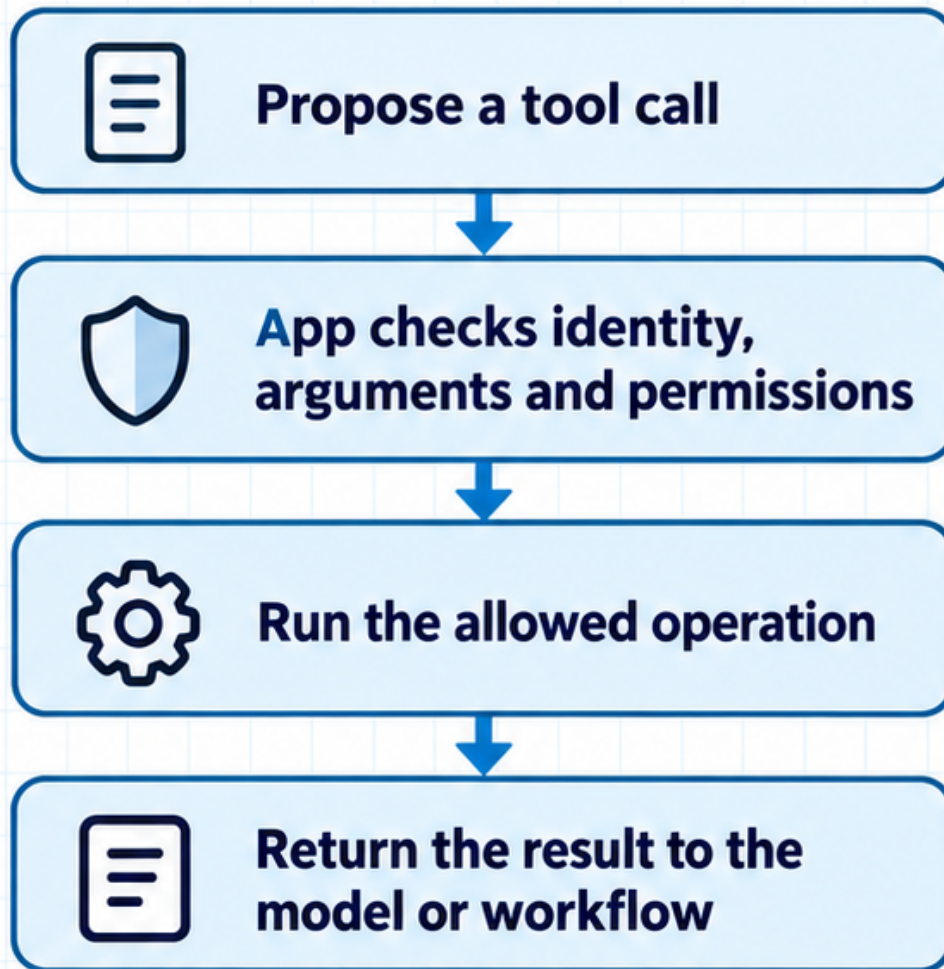
[1] Toy returns policy



RAG can reduce guessing.
It does not guarantee correctness.

TOOLS CONNECT AI TO THE WORLD

A tool is an interface to data or an action.



Read: `get_order_status`

Write: `request_refund`

For this example, refunds require human approval.



Application code enforces what a tool may do.

FIXED WORKFLOW OR AGENT?

Choose how the next step is decided.

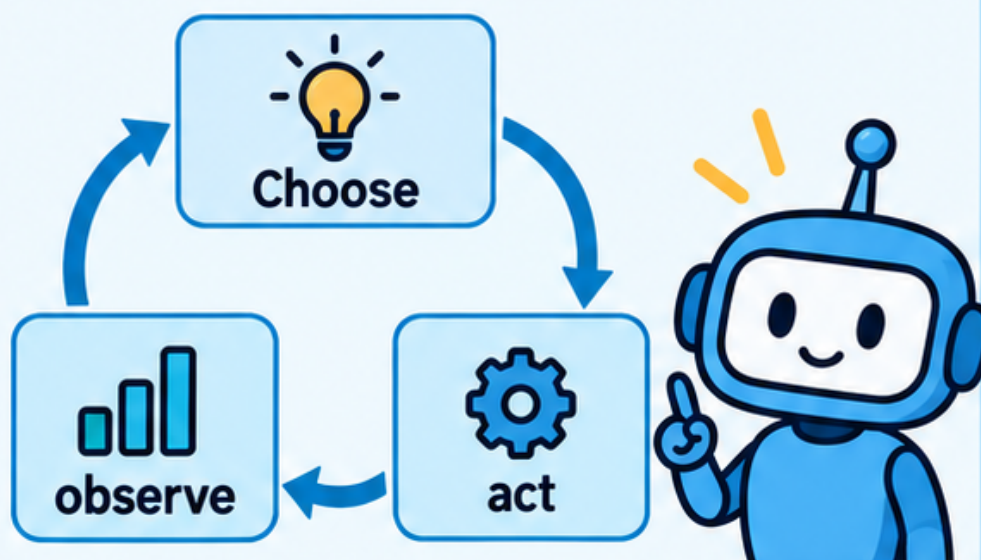
FIXED WORKFLOW

Code chooses the sequence.



AGENT

The model chooses steps from results.



Set step, time and spend limits.



Use more autonomy when the task needs flexible decisions.

STATE, MEMORY AND CACHE

Similar storage, different purposes

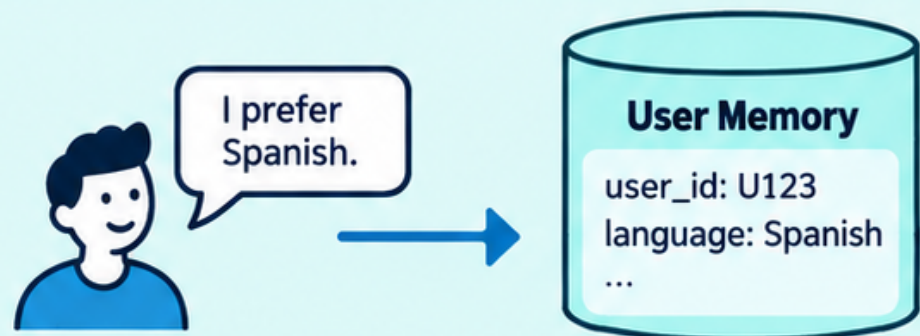
STATE: Where are we in this task?

Example: refund request #R42 is awaiting approval.



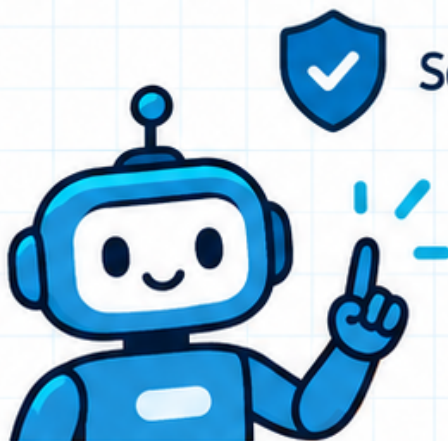
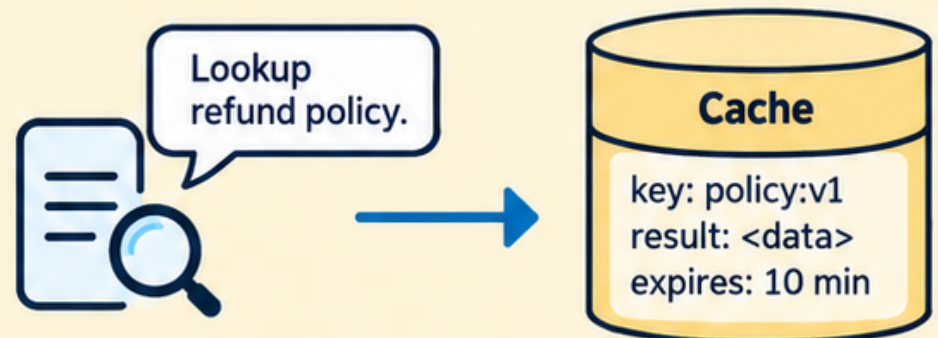
MEMORY: What may help across interactions?

Example: a language preference the user chose to save.



CACHE: What result can we safely reuse?

Example: a recent policy lookup, within its expiry.



Scope stored data to the right user and retention rules.

Important action status belongs in durable storage.

CHOOSE TOOLS BY RESPONSIBILITY

Examples to explore, not a required shopping list.

RESPONSIBILITY	EXAMPLE TOOL
API and application logic	FastAPI
Data and durable state	PostgreSQL
Vector search in PostgreSQL	pgvector
Model inference	Hosted model API or vLLM
ML runs and model versions	MLflow
Tracing and evaluation	OpenTelemetry + Phoenix
Caching and rate limits	Redis



Choose the requirement first, then the tool.

DESIGN FOR TRADE-OFFS

Improve the outcome within real constraints.

QUALITY



Does the task succeed?

LATENCY



How long does the user wait?

COST



What does a successful task cost?

Example: retrieving more documents may add useful evidence, but also delay and token cost.



Measure the effect of a change. More is not always better.

DESIGN THE FAILURE PATH

The system needs a response when a step fails.



No reliable evidence



Say so or hand off



Model request times out



Use a bounded retry or fallback



Action is not authorized



Block it and explain



Refund result is unknown



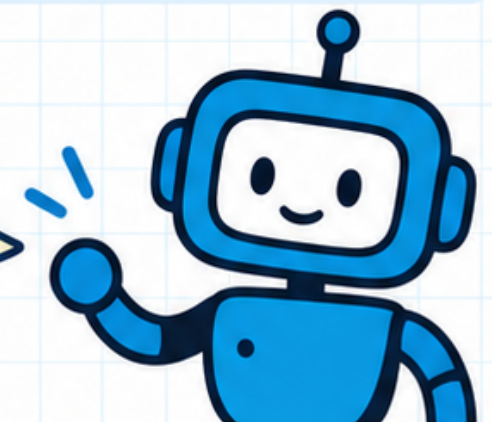
Check saved status before retrying



Use idempotency keys to prevent duplicate write effects.



A useful fallback is part of the architecture.



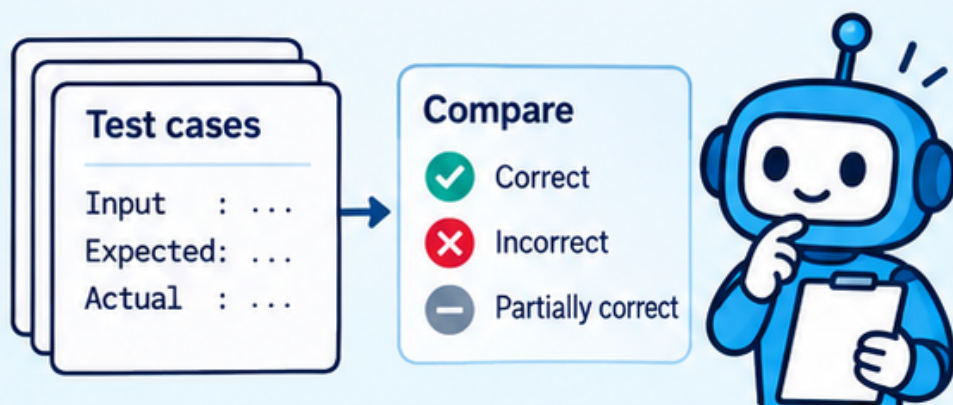
EVALUATE THE TASK. TRACE THE RUN.

Quality checks and operational visibility work together.

EVALUATION

Did the system do the right thing?

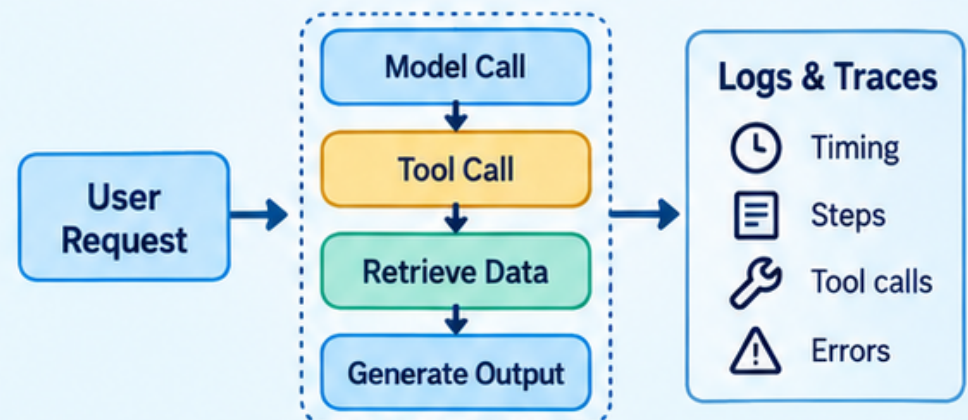
Use test cases with expected outcomes.



TRACING

What happened during this request?

Inspect steps, timing, tool calls and errors.

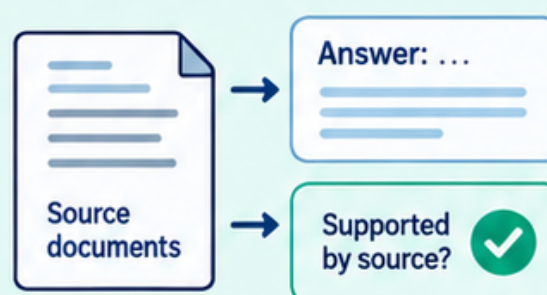


EXAMPLES

Forecast: compare predictions with actual demand.



RAG: check answers against source evidence.



Agent: verify the outcome and action permissions.

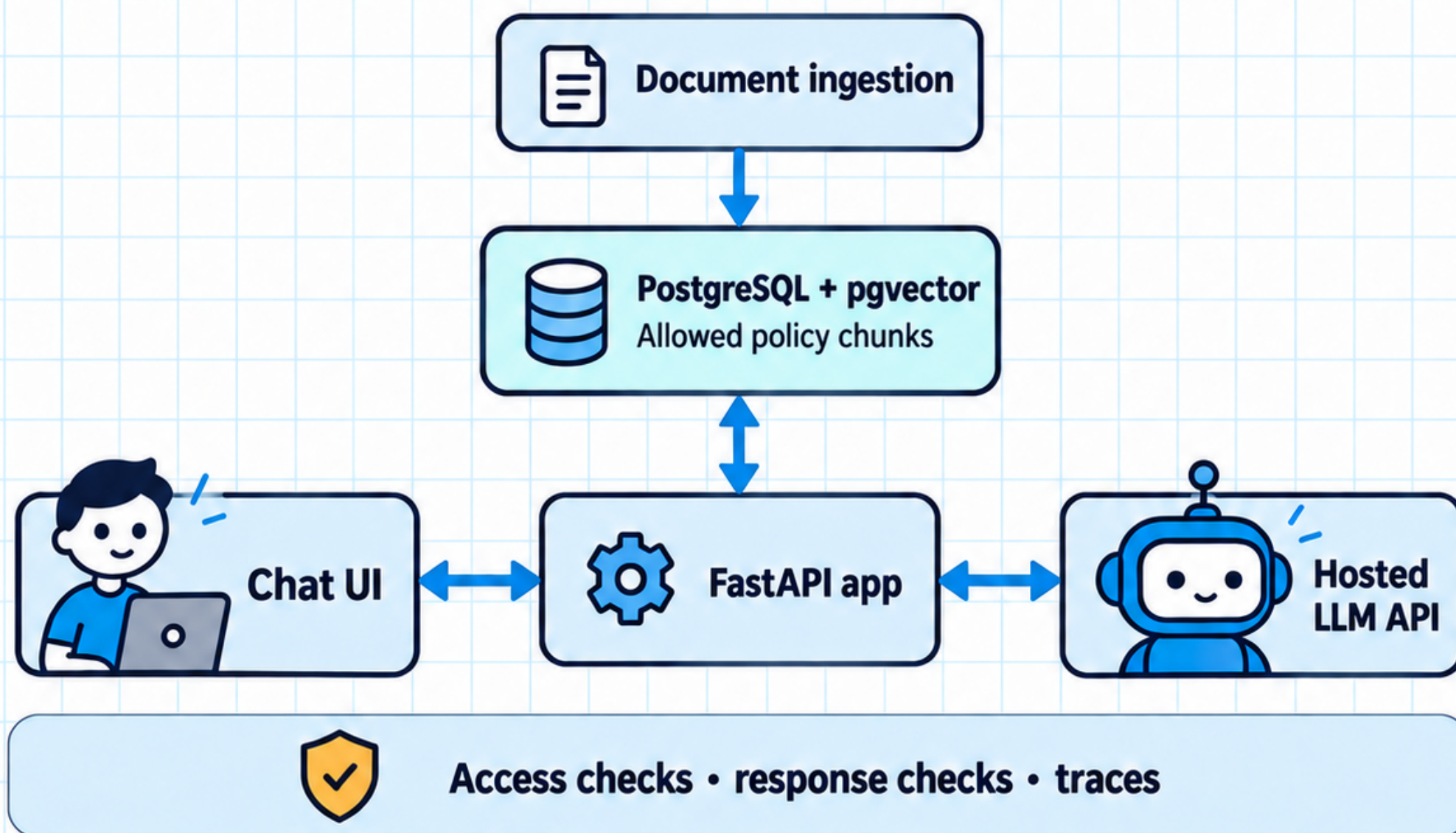


A successful API call does not prove a successful task.



BUILD THE SMALLEST USEFUL V1

Example: read-only policy assistant



Add live tools, caching or an agent loop when a need is measured.



Start with one job and a testable success condition.

YOUR FIRST SYSTEM-DESIGN EXERCISE

Choose a policy assistant or a demand forecast.

1 What does the user need?



2 Which data and model are required?



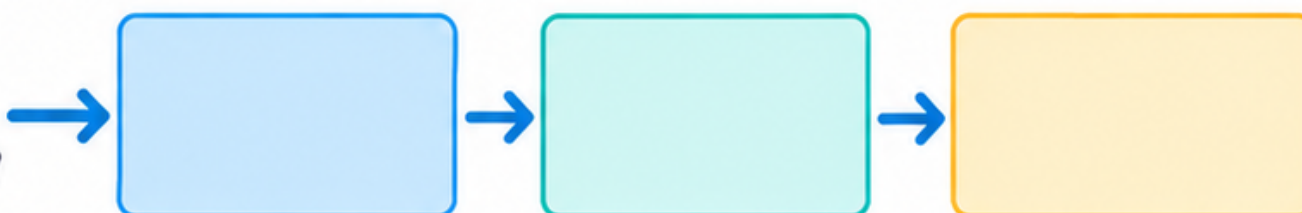
3 What can fail, and what happens then?



4 How will you measure success?



Sketch the request path. Label each tool's job.



DAY 2: Requirements and success metrics



Save this foundation • Follow for Day 2

@learn.machinelearning