Current State of AI – Aug 2026

Natural Language / Multimodal input

Gemini
xAI
AI21 labs
stability ai
Qwen
Claude
Pi
phind
01

Natural language output

Speech

Image / video

…

“Large language / Multimodal Machine Learned models”

Current State of AI – Aug 2026

- Powerful language & multimodal models
- General-purpose chat UX
- Users care about outcomes, not orchestration
- LLMs can do more than Q&A
- Workflows unlock real-world utility
- Developers abstract the complexity

Consider this example problem:

A student asks: “Can I graduate this semester, and what exactly must I do next?”

What kind of information is needed to solve this?

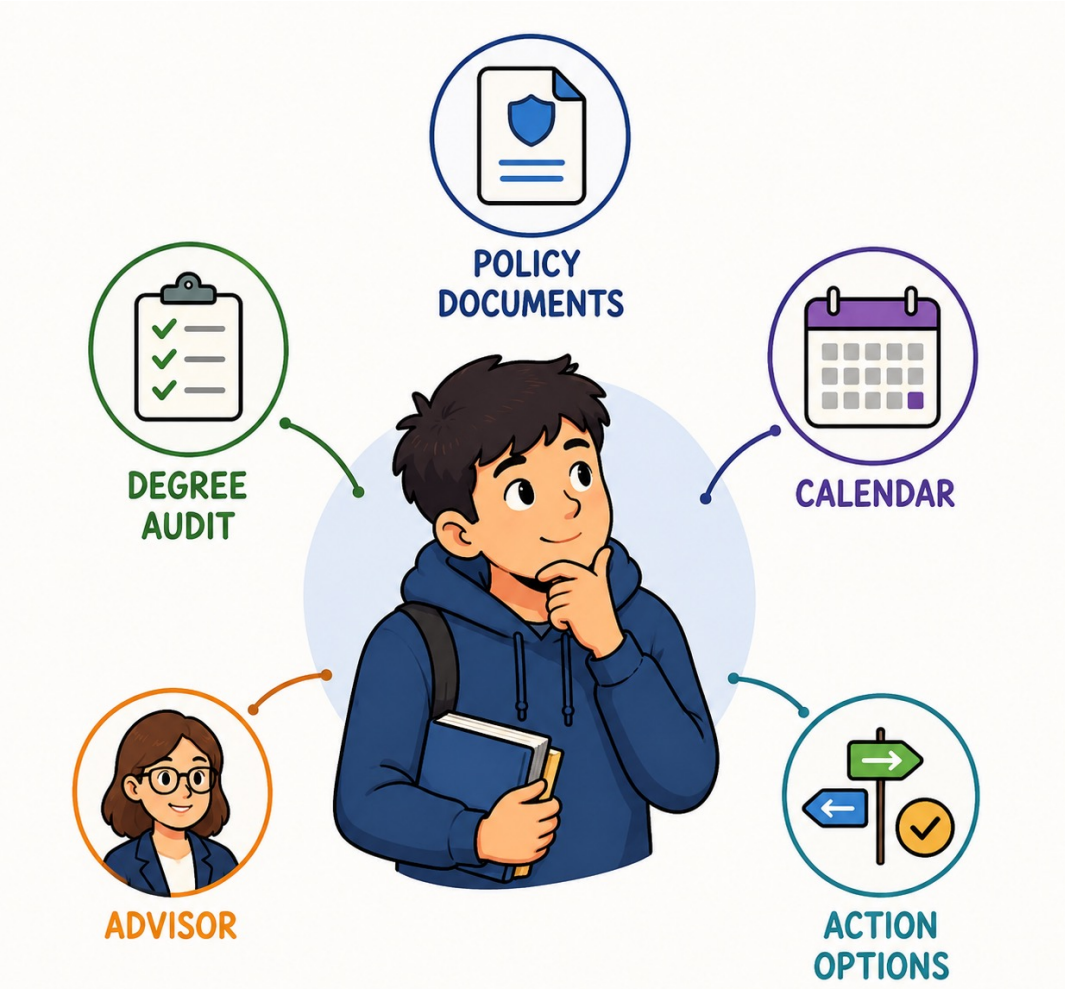
Consider this example problem:

A student asks: “Can I graduate this semester, and what exactly must I do next?”

This looks like a language question. It is actually a systems problem. Why?

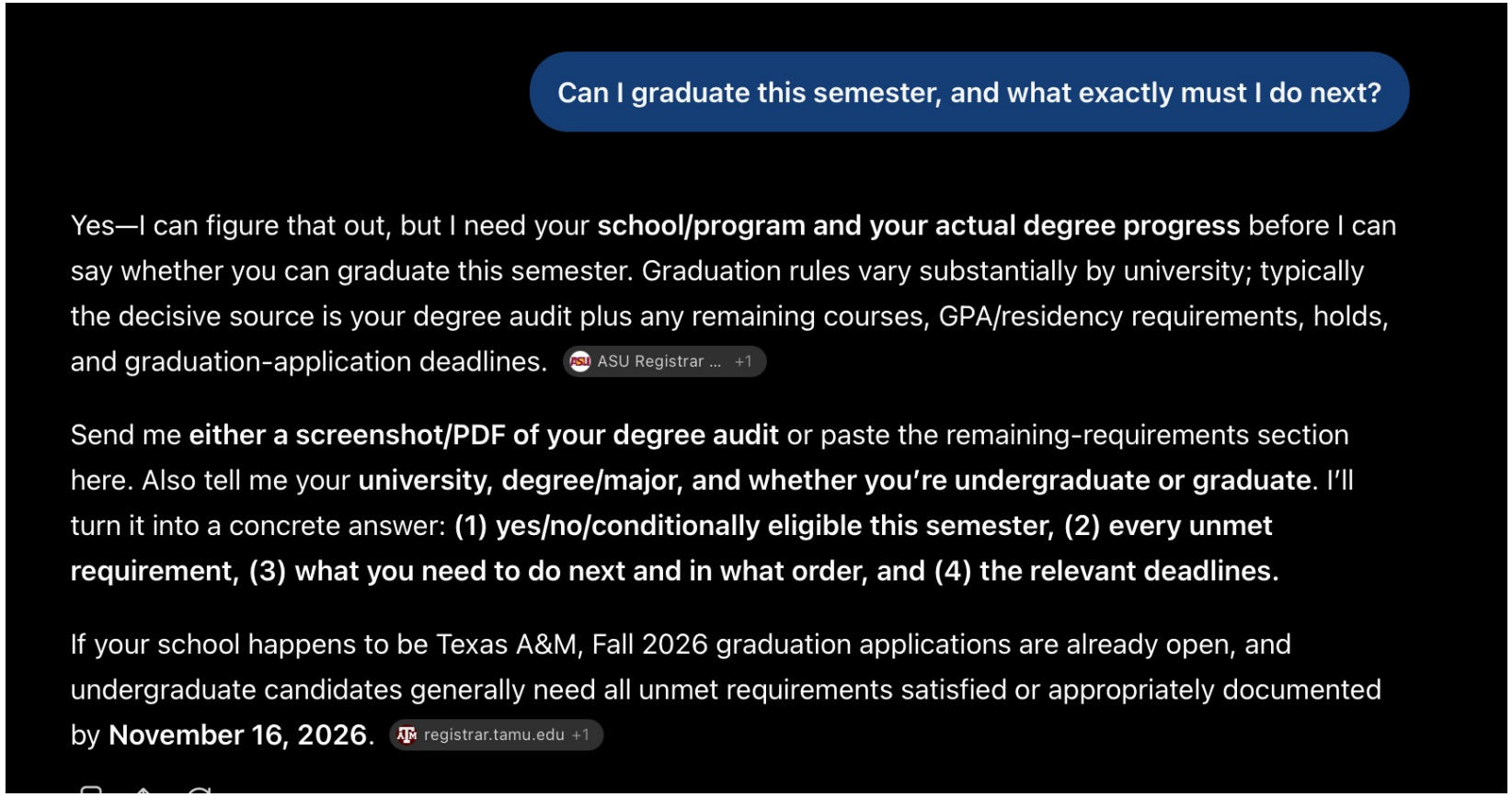
Why this is a serious AI use case?

- The request is expressed in natural language, not database fields.
- The answer depends on policy, degree records, dates, exceptions, and the student's intent.
- A useful response must explain the answer and identify the next action.
- A wrong answer can waste time or create real academic consequences.

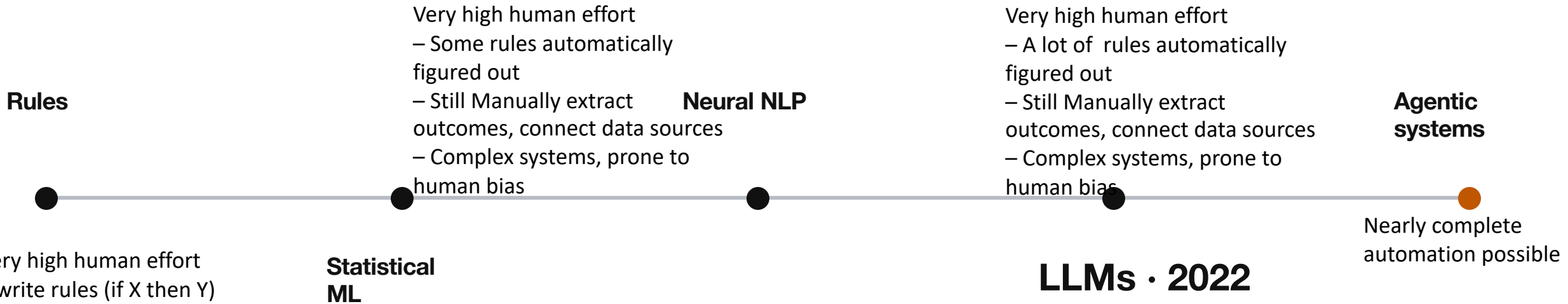


Can I graduate this semester, and what exactly must I do next?

And obviously…..



How NLP / AI has aimed to solve it?



The solution strategy changed in five major waves

Before machine learning: encode the language and policy as rules

- Match phrases such as “graduate,” “degree,” or “requirements.”
- Route the request through a hand-built decision tree.
- Return a scripted answer for the matched intent.
- Ask a human to handle anything outside the anticipated cases.

Reliable inside the rule boundary; brittle outside it.

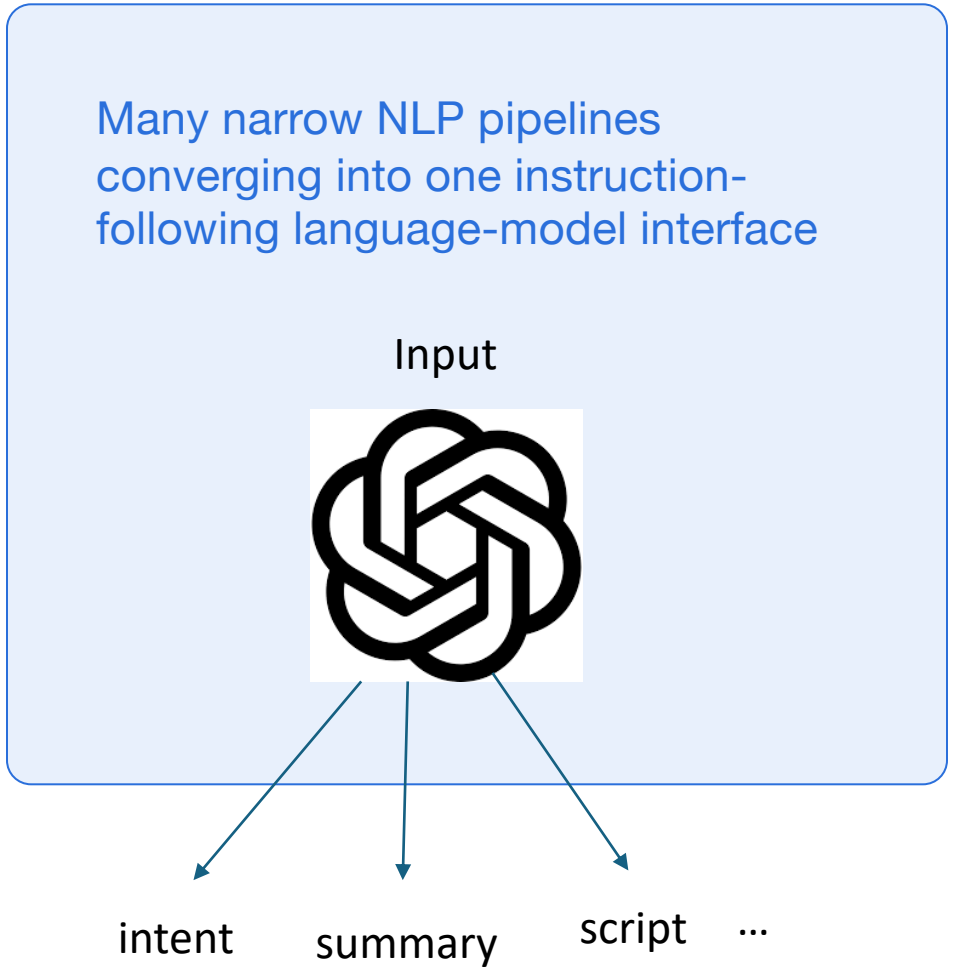
Machine-learning NLP: learn patterns from labeled examples

- Train an intent classifier from historical questions.
- Extract entities such as program, semester, or requirement.
- Retrieve a prepared answer or route the case to a specialist.
- Retrain when categories, language, or policy distributions change.

Better coverage—but the task and output categories still have to be designed in advance.

Large language models changed the interface in 2022

- One general model could follow instructions across many language tasks.
- Users could express goals conversationally rather than select a fixed intent.
- The model could summarize, explain, extract, classify, and draft in one interface.
- Few-shot examples and natural-language instructions replaced some task-specific training.



But answering is not the same as completing the work

- The model still needs current policy evidence – STAY UPDATED
- It cannot assume access to the student's degree record - HALLUCINATION
- It must calculate, verify, ask follow-up questions, and respect permissions - GUARDRAILS
- Someone—or something—must decide what happens after each result - REFLECTION

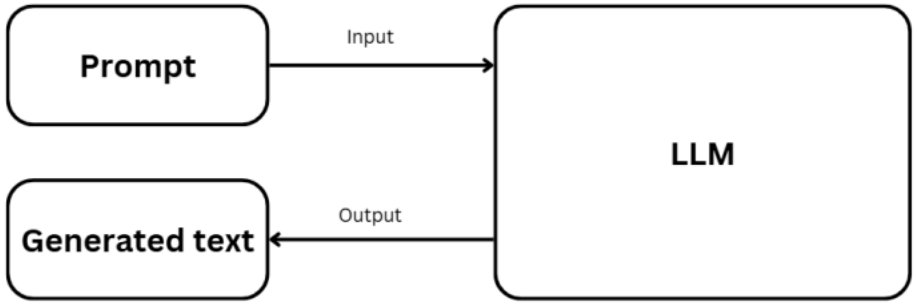
That gap motivates the move from an LLM response to an agentic system.

The same case across three system generations

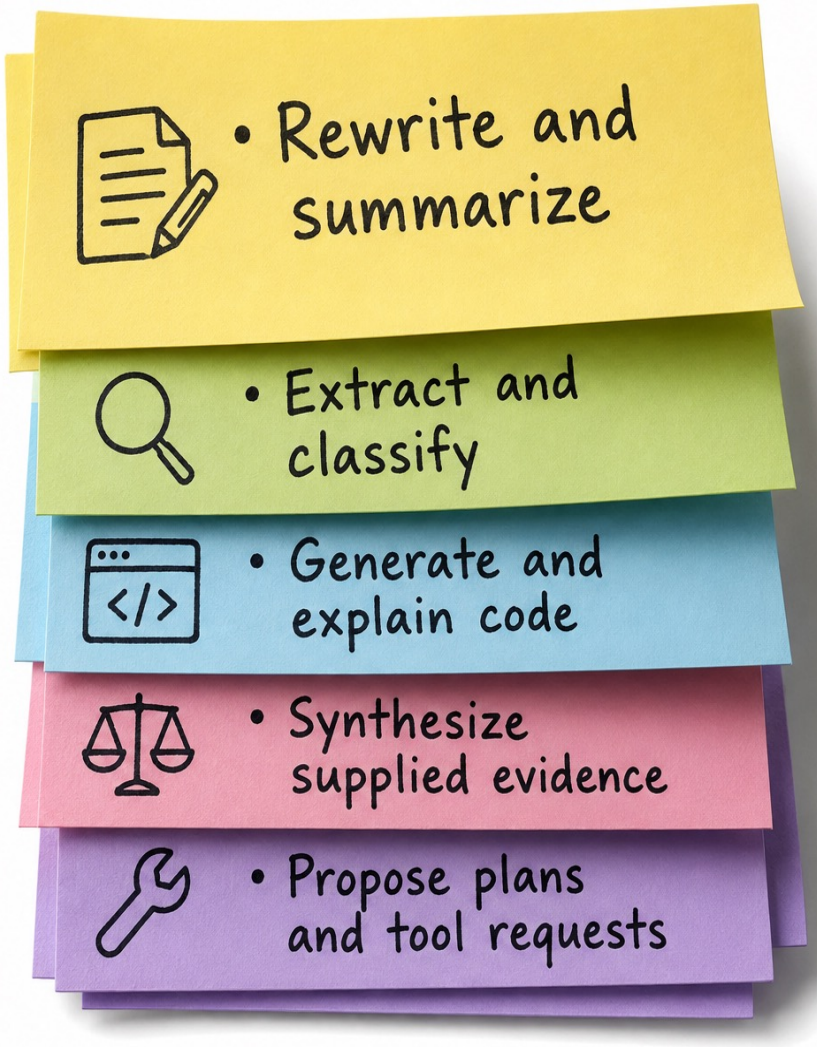
Pre-LLM system	LLM application	Agentic system
Classify and route	Explain supplied evidence	Pursue a defined outcome
Fixed decision path	One or several model calls	Choose among approved next steps
Prepared responses	Fluent synthesis	Retrieve, calculate, verify, clarify
Human resolves exceptions	Human supplies context	Human governs consequential actions

Large Langauge Model: A language model maps context to a response

• It receives tokens representing instructions, examples, evidence, and user input.
• It produces a probabilistic continuation or response.
• The response depends on model, context, parameters, and provider behavior.
• Large → bulky transformer based neural networks that train on trillions of text or multimodal tokens or simply words
• Language models→ Models trained to predict next word from previous words



The “same" model is remarkably useful



Useful does not mean guaranteed

Capability

- Can produce a plausible classification
- Can propose a sequence of steps
- Can draft a tool request

Guarantee

- Classification is correct
- Plan is executable
- Request is authorized or executed

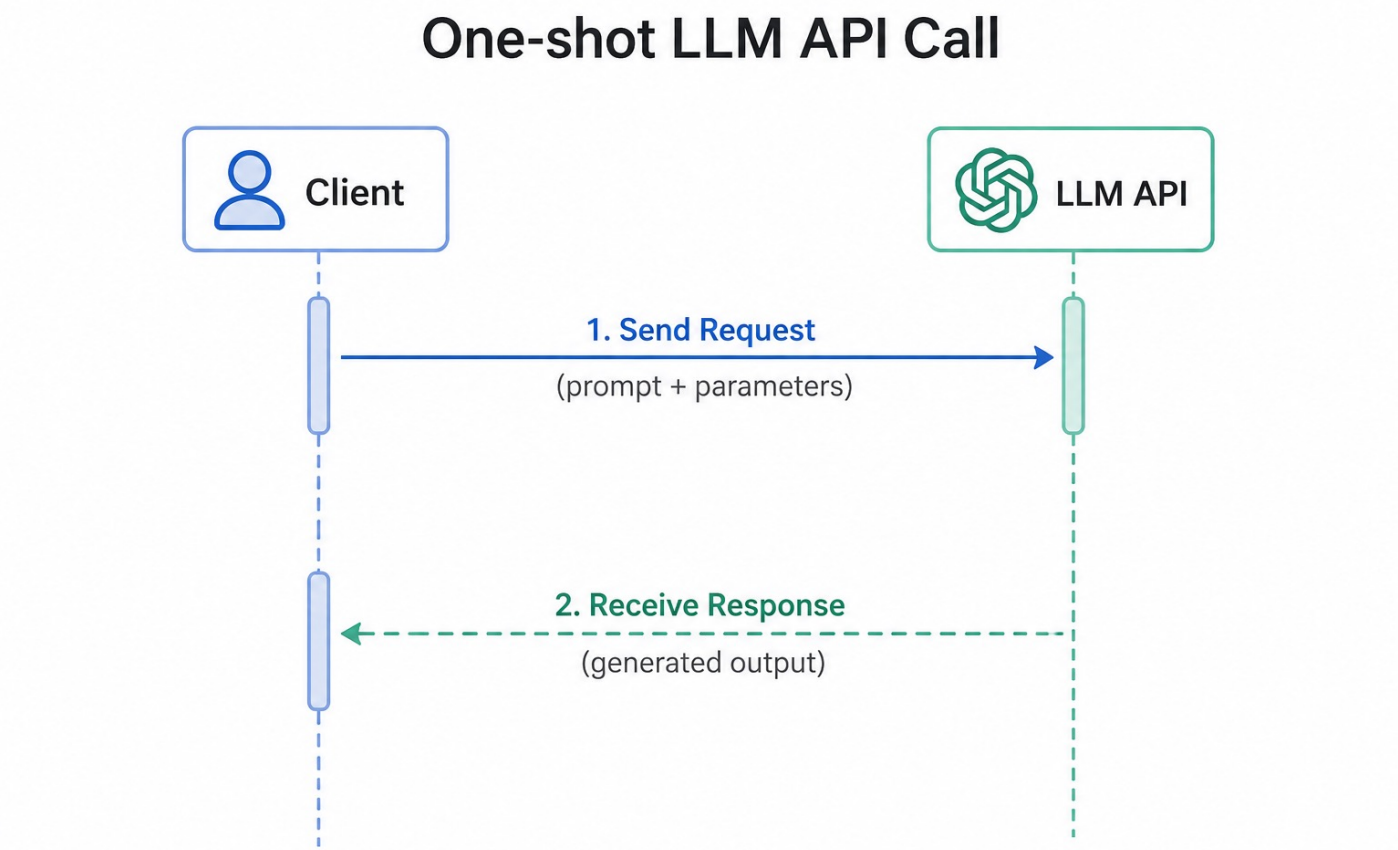
Fluency is a user-interface property

It is not proof of truth, evidence, execution, or permission.

A minimal model call has a small footprint

- Choose a model.
- Supply instructions and user content.
- Receive output and usage metadata.
- Stop.

Lacks broader context buildup



But…the same model can live inside very different architectures

- A one-shot rewriting tool
- A fixed retrieval pipeline
- A stateful workflow
- An adaptive tool-using agent
- A multi-agent research system

Quick check

If an application calls a model three times in a fixed sequence, is that sufficient for solving a workflow problem?

Make your case using control flow—not the number of calls.

Why this question matters now?

- Models no longer only generate text—they are embedded in systems that retrieve, plan, use tools, remember, and act.
- The hard problems increasingly sit outside the model: control, evidence, failure, cost, security, and accountability.
- This course is about building those systems with discipline.

INF 385T

Today’s introduction

The anatomy of an agentic system—and the boundaries that keep it useful

What is an Agent?

Stuart Russell and Peter Norvig define an agent in [*Artificial Intelligence: A Modern Approach*] as ”**anything that perceives its environment through sensors and acts upon that environment through actuators (or effectors). An agent function maps any given percept sequence to an action”**.

Classical AI

Environment → Sensors → Percepts → Agent → Action → Actuators → Environment

Generative AI

Digital environment → Context/observations → LLM → Tool/action selection → Tools/APIs → Digital environment

An agentic system has several cooperating parts

No single component supplies capability, evidence, authority, and accountability by itself.

A goal needs more than a topic

- Requested outcome
- Required evidence
- Important exclusions
- Completion criteria
- What should trigger clarification or escalation

Tools extend what the system can do

- Retrieve a record
- Search documents
- Calculate or transform
- Draft an artifact
- Create an external effect

Tool design determines the operational capability surface.

A tool request is not an executed action

The model proposes. Trusted application code validates, authorizes, and executes.

Observations close the loop

- Tool results
- Environment feedback
- Validation errors
- Human approval or correction
- Evidence that changes the next decision

State and memory are related—but not identical

State

- Current step
- Completed actions
- Budgets and retries
- Terminal status

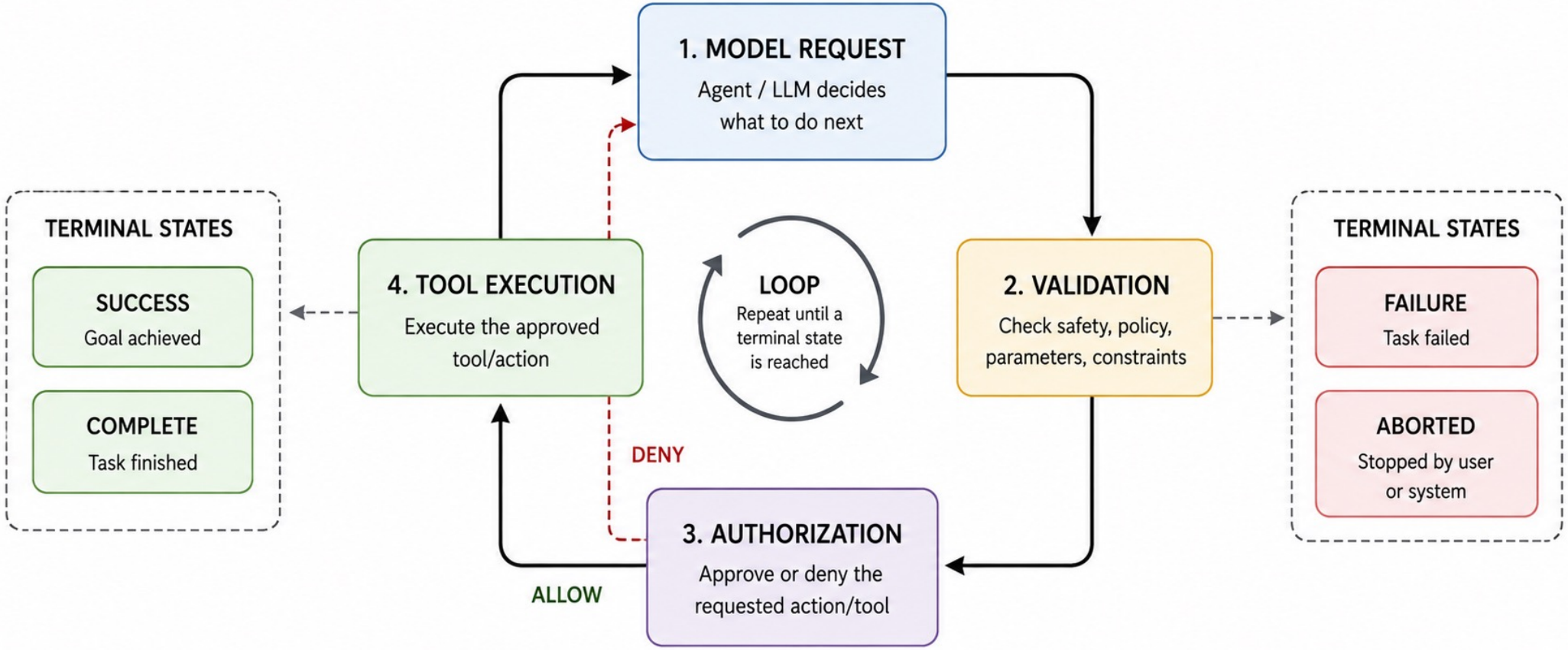
Memory

- Prior interaction
- Persistent fact
- Past episode
- Reusable procedure

Policy defines the boundary

- Which tools are available?
- Which data may be accessed?
- Which actions require approval?
- What must stop or escalate?
- Which resource limits apply?

A bounded agent loop makes the control points visible



Every loop needs more than one limit

- Maximum turns
- Maximum tool calls
- Retry and repetition limits
- Time, token, and cost budgets
- Explicit completion, clarification, blocked, and failure states

No more tool calls does not prove completion

Completion is an application-level claim that must be checked against the user's requested outcome and evidence.

Evaluation is part of the system

- Did the task succeed?
- Were claims supported?
- Were the right tools used?
- Did the system respect limits?
- Did it recover or stop safely?

Humans remain part of the control architecture

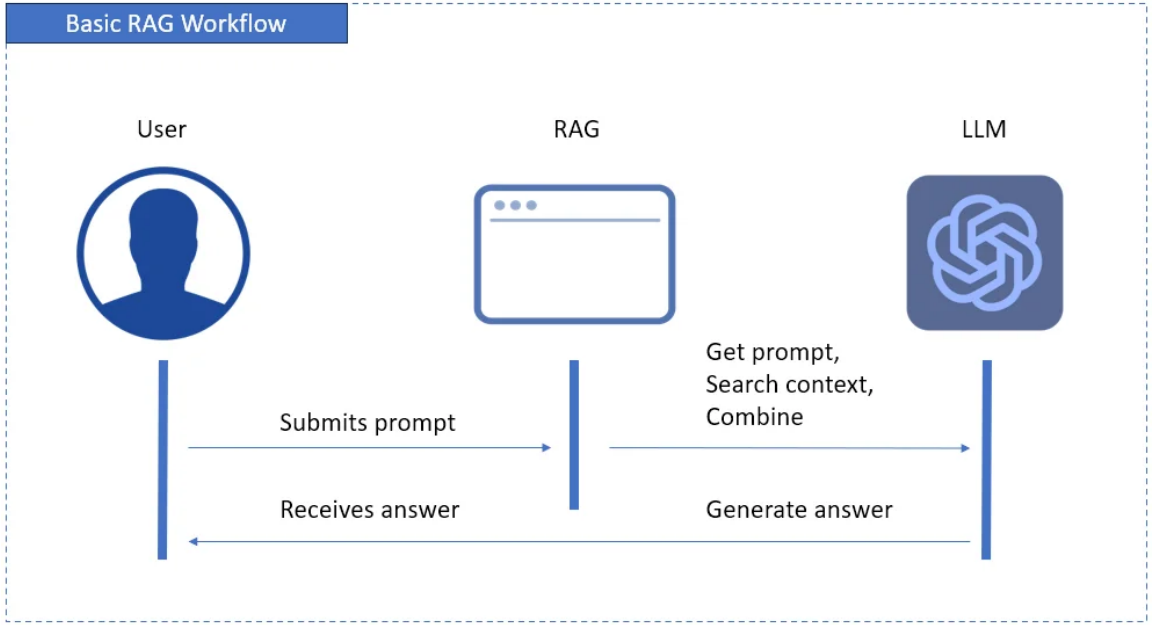
- Define goals and acceptable autonomy
- Approve consequential actions
- Correct or interrupt ongoing work
- Review uncertainty and residual risk
- Remain accountable for deployment decisions

4 · When should we use an agent?

Adaptation is valuable only when the task needs it

Use the simplest adequate architecture

- Direct call for bounded transformation
- Retrieval Augmented Generation (RAG) pipeline for grounded question answering



- Workflow for known stages and business rules
- Agent when the next step genuinely depends on observations
- Multiple agents only for measured specialization or parallelism

Keep the architecture simpler when…

- The transformation is stable and well specified.
- A deterministic rule produces the correct result.
- Latency or cost must be tightly predictable.
- The action is high impact but cannot be verified or approved.
- The agentic version has not beaten a competent baseline.

Autonomy trades flexibility for control

Potential gain

- Adaptation
- Breadth
- Recovery
- Less manual sequencing

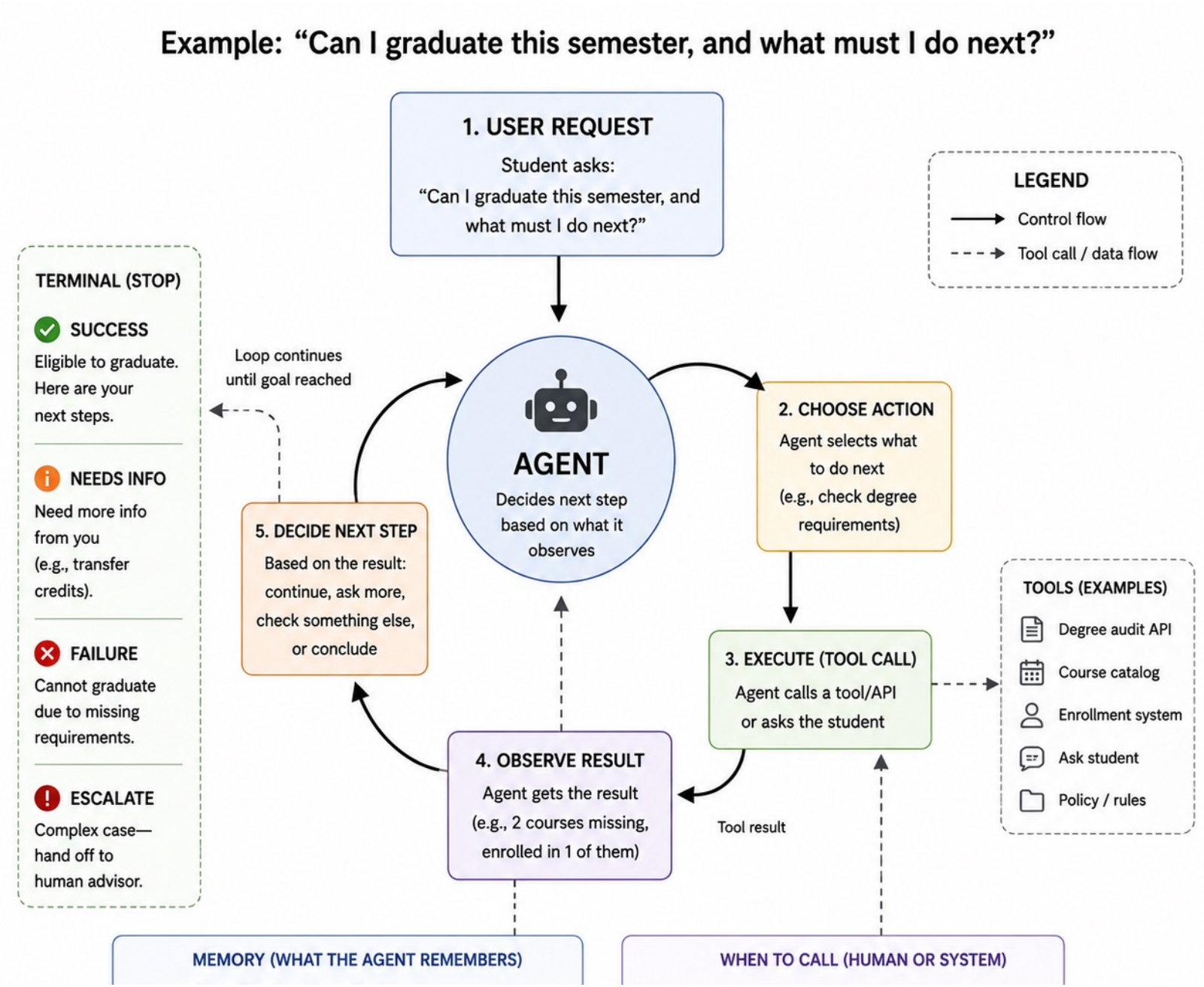
Added burden

- Nondeterminism
- Evaluation
- Cost + latency
- Security + accountability

Seven questions before adding an agent loop

- What outcome is required?
- Which steps are known?
- What information is missing?
- Which tools are truly necessary?
- Which actions have side effects?
- Who has authority?
- How will completion be verified?

Working Example – “Can I graduate this semester, what must do I DO NEXT?”



Exercise:

A graduate-program AI assistant answers policy questions, calculates deadlines, and may draft but never send messages.

Choose the simplest adequate architecture. Name one stopping condition and one human-control point.

Three ideas to carry into Week 2

- Agency is a property of the whole system.
- Control flow tells us more than marketing labels.
- Every probabilistic boundary needs an explicit contract.

WEEK 2

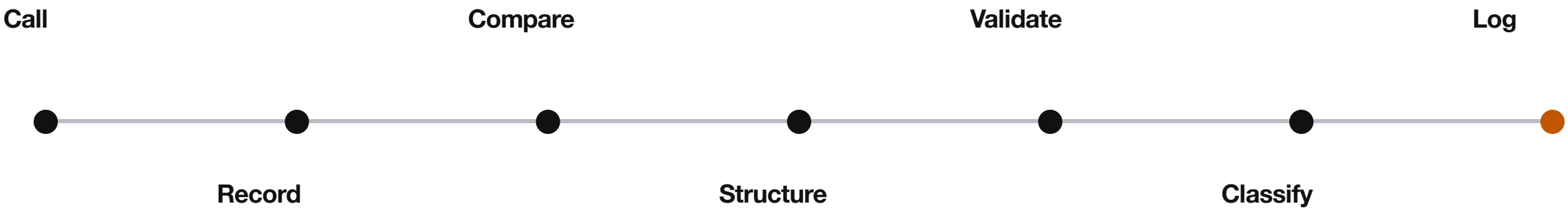
Next week: structured model interfaces and tool use

Pydantic · JSON Schema · structured outputs · function calling

5 · Today’s lab

We will build the architecture ladder one layer at a time

The notebook adds discipline around one small model call



First: make a basic call with gpt-5-nano

- Use the instructor-managed access method.
- Send a bounded synthetic task.
- Inspect the response object rather than assuming it is only text.
- Avoid rerunning completed paid cells without a reason.

Two key prompts to keep in mind:

- **Instruction Prompt:** This is the part of the prompt that defines the role, behavior, constraints, and overall task for the language model.
- **User Input Prompt**: This is the specific query or data provided by the user that the model needs to process according to the given instructions.

```
instructions = (
        "You are a concise teaching assistant for a graduate information-science course. "
        "Use plain language. Do not claim that every multi-step program is an agent."
)
user_input = (
        "Explain the difference among a language model, a deterministic workflow, "
        "and an agent. Give one example of each in no more than 180 words."
)
# model API call
response = client.responses.create( model=MODEL, instructions=instructions, input=user_input,)
```

Then: make the experiment reproducible

- Record model and instructions.
- Record parameters and input.
- Measure latency and usage.
- Keep a sanitized log of what actually happened.

A prompt comparison needs controlled variation

Hold constant

- Task input
- Model
- Relevant parameters
- Evaluation rule

Change deliberately

- Instruction wording
- Required format
- One design choice at a time

Structured output turns prose into an interface

- Named fields
- Expected types
- Closed categories
- Machine-checkable validation

Week 2 will develop this properly; today we use a lightweight first example.