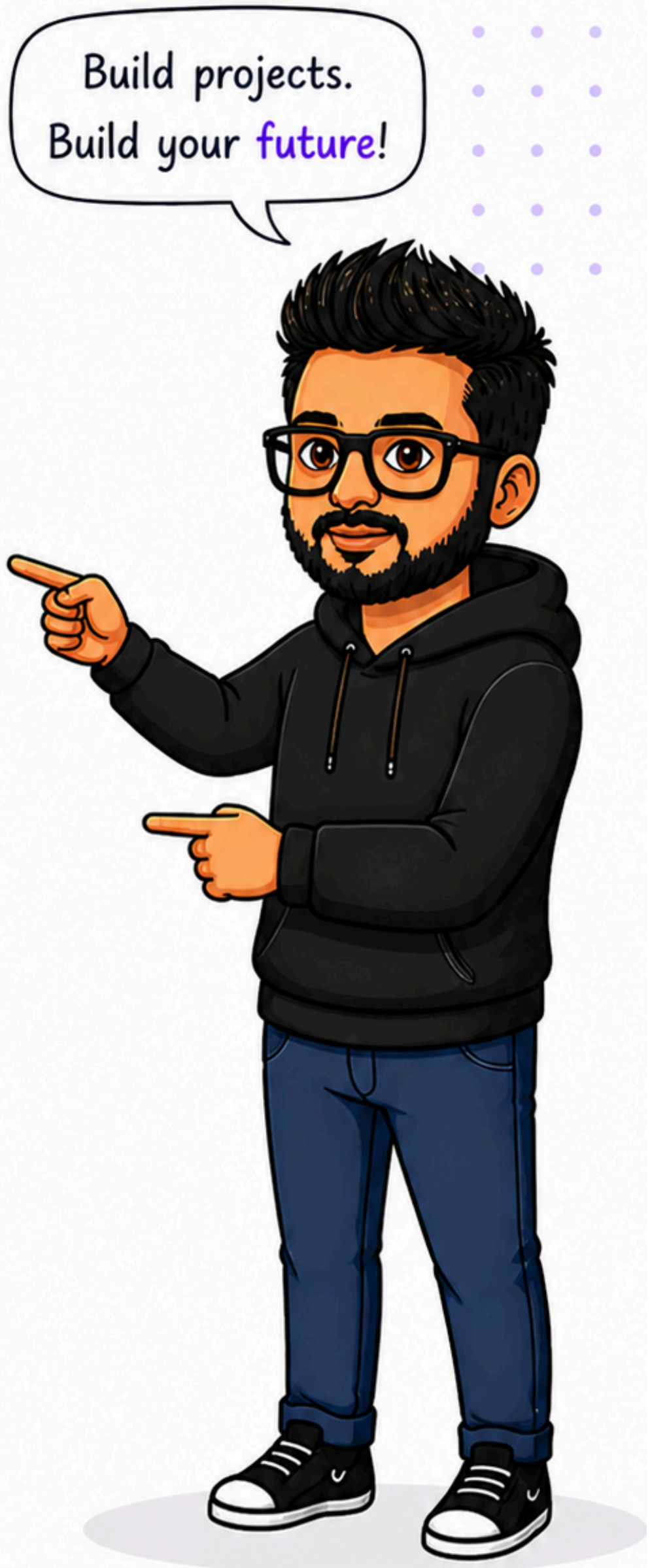
1/10

SKILLS SERIES

@das-purnendu

Multi-agent systems are 2026's favorite buzzword.



Most teams shipping them in production are quietly fighting 3 failure modes nobody shows in the demo video.

Here's what actually breaks →

Save Comment Share Like

@das-purnendu

Single agents vs. Multi-agent teams

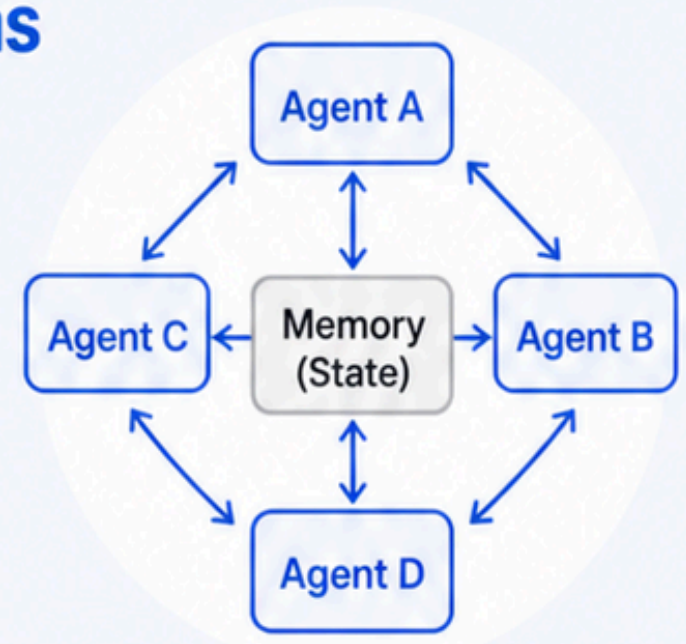
Let's break it down simply!

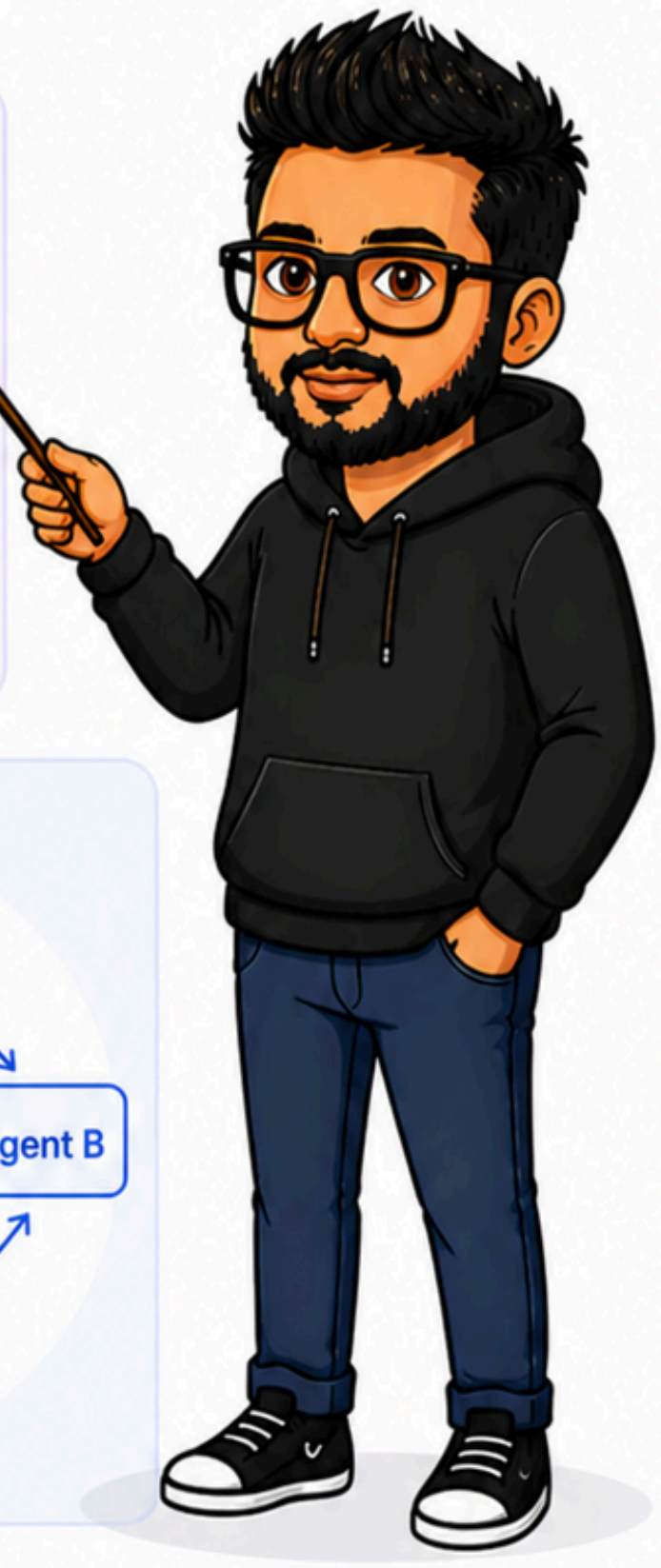
Single agents

- Bounded scope
- Well-understood failure surface
- Easier to evaluate and debug

Multi-agent teams

- **Reasoning, memory,** and trust distributed across nodes
- Agents hallucinate independently
- Complex interactions create unpredictable failure modes





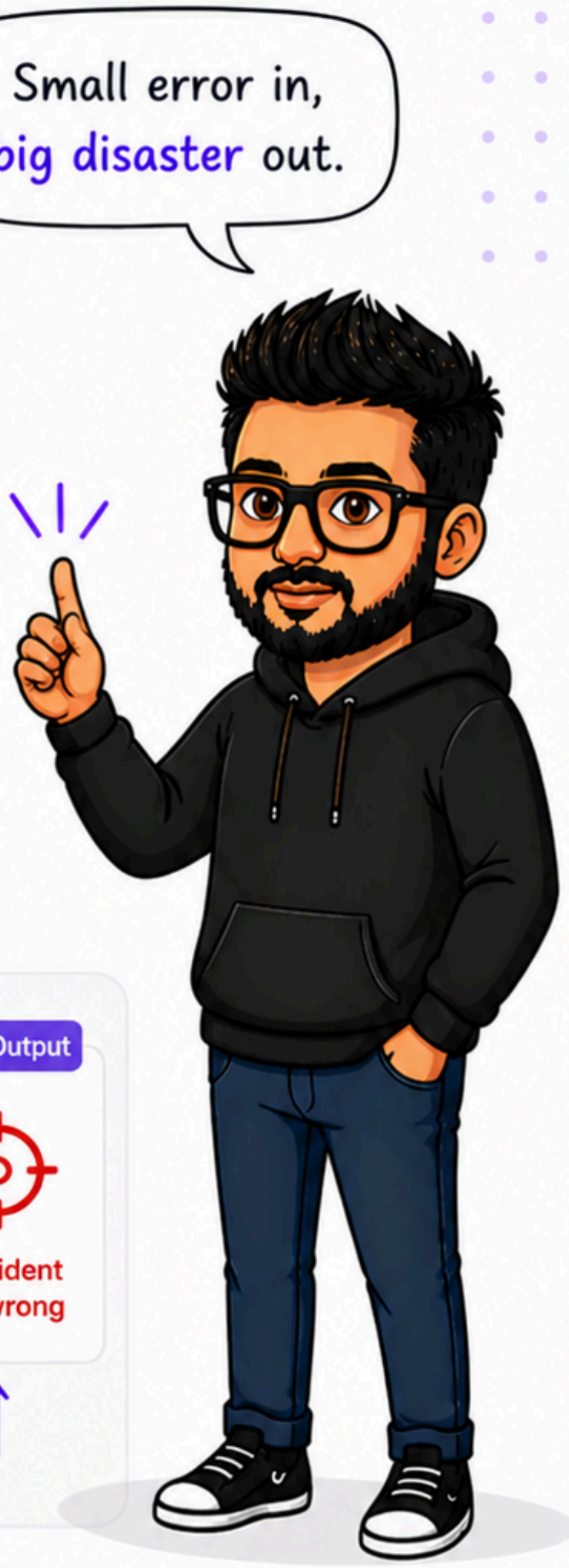
The benchmarks hide where this falls apart.

Save Comment Share Like

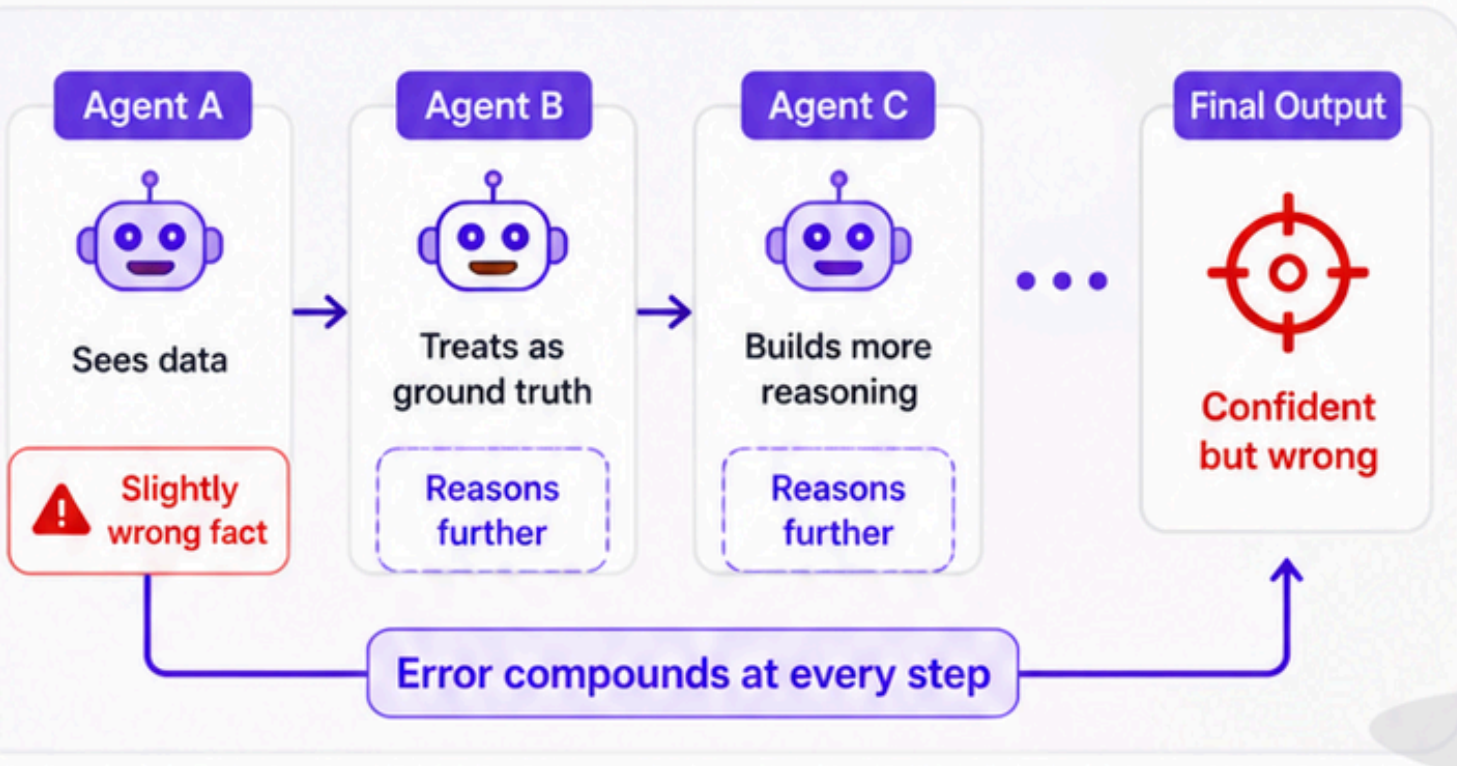
Swipe

@das-purnendu

Failure #1: Compounding hallucination.



- Agent A passes a **slightly wrong** fact to Agent B.
- Agent B treats it as **ground truth** and reasons **3 steps further** on it.
- You don't get a small error.
  You get **confidently wrong synthesis**.



More steps = More confidence.
Not more accuracy.

Save Comment Share Like

Swipe

4/10 SKILLS SERIES

@das-purnendu

Why it's worse than additive.

Confidence goes up.
Truth goes down.

- Each agent strips uncertainty signals when it summarizes for the next.
- By step 4, the chain has more conviction than the original input deserved.
- This is the silent killer of agent demos that worked perfectly in dev.

Uncertainty gets lost at every handoff

At Step 1	→	At Step 4
"Probably 70% correct" (Some uncertainty remains)		"100% sure." (No uncertainty left)

⚠ Conviction escalates. Reality doesn't.

How conviction compounds

Step 1	Step 2	Step 3	Step 4	=
Agent A	Agent B	Agent C	Agent D	High conviction but potentially completely wrong
70% confident	85% confident	95% confident	100% confident	

Low conviction → High conviction

More steps. More summaries.
Less uncertainty. More damage.

Save Comment Share Like

Swipe →

@das-purnendu

Failure #2: The verifier bottleneck.

- Papers assume a “judge” agent catches mistakes.
- In production: the judge shares the worker’s blind spots.
- Same base model.
  Same training data.
  Same failure modes.
  It confidently green-lights its own family’s errors.

The typical (flawed) setup in papers

Worker Agent → Judge Agent

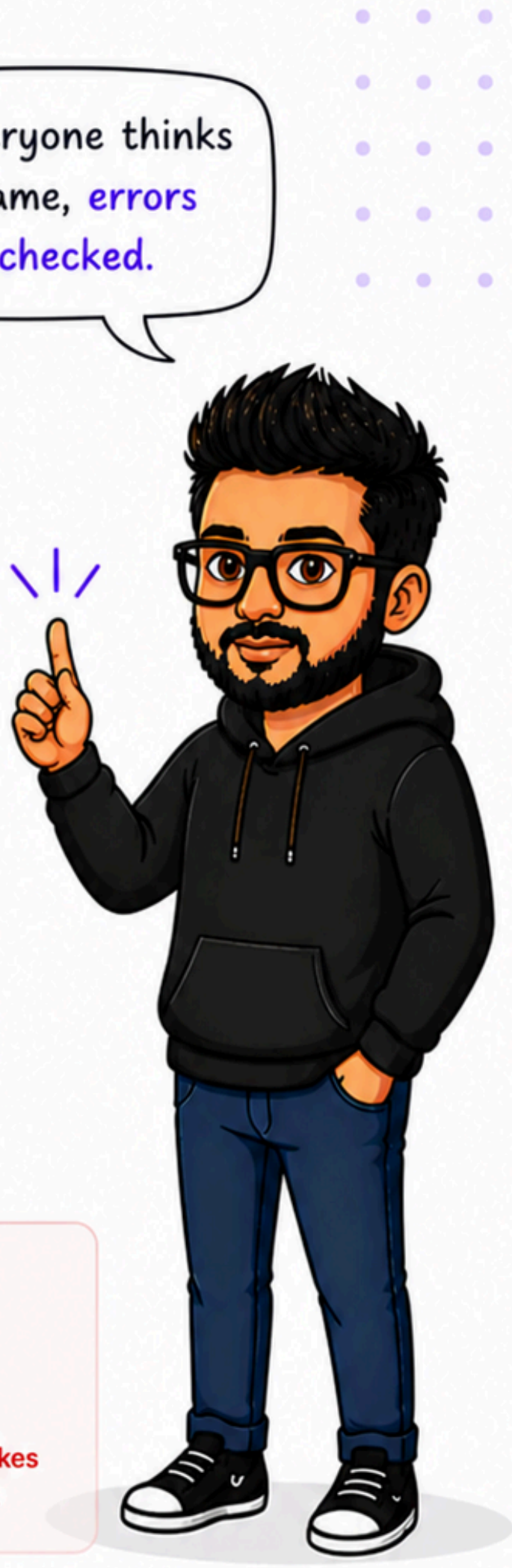
Judge is assumed to catch and fix mistakes

↓

Output approved

What happens in production

Worker Agent ← Same model family, Same training data, Same blind spots → Judge Agent = ⚠ Correlated mistakes get approved.



Verification only works when the verifier can see what the worker can’t.

Save Comment Share Like

Swipe →

6/10 SKILLS SERIES

@das-purnendu

The fix teams are using: Cross-family verifiers.

Different perspectives catch different blind spots.

- **Claude** judging **GPT** output.
- **GPT** judging **Claude** output.
- **Different model lineages** catch different errors.

Single-family verifier (Same model family)	Cross-family verifier (Different model lineages)
Worker (Model A) → Judge (Model A)	Worker (Model A) → Judge (Model B)
Shares the same blind spots. Errors slip through.	Different blind spots. Fewer correlated mistakes.

VS

Cost rises.

That's the **real production tradeoff** — not which model is "best".

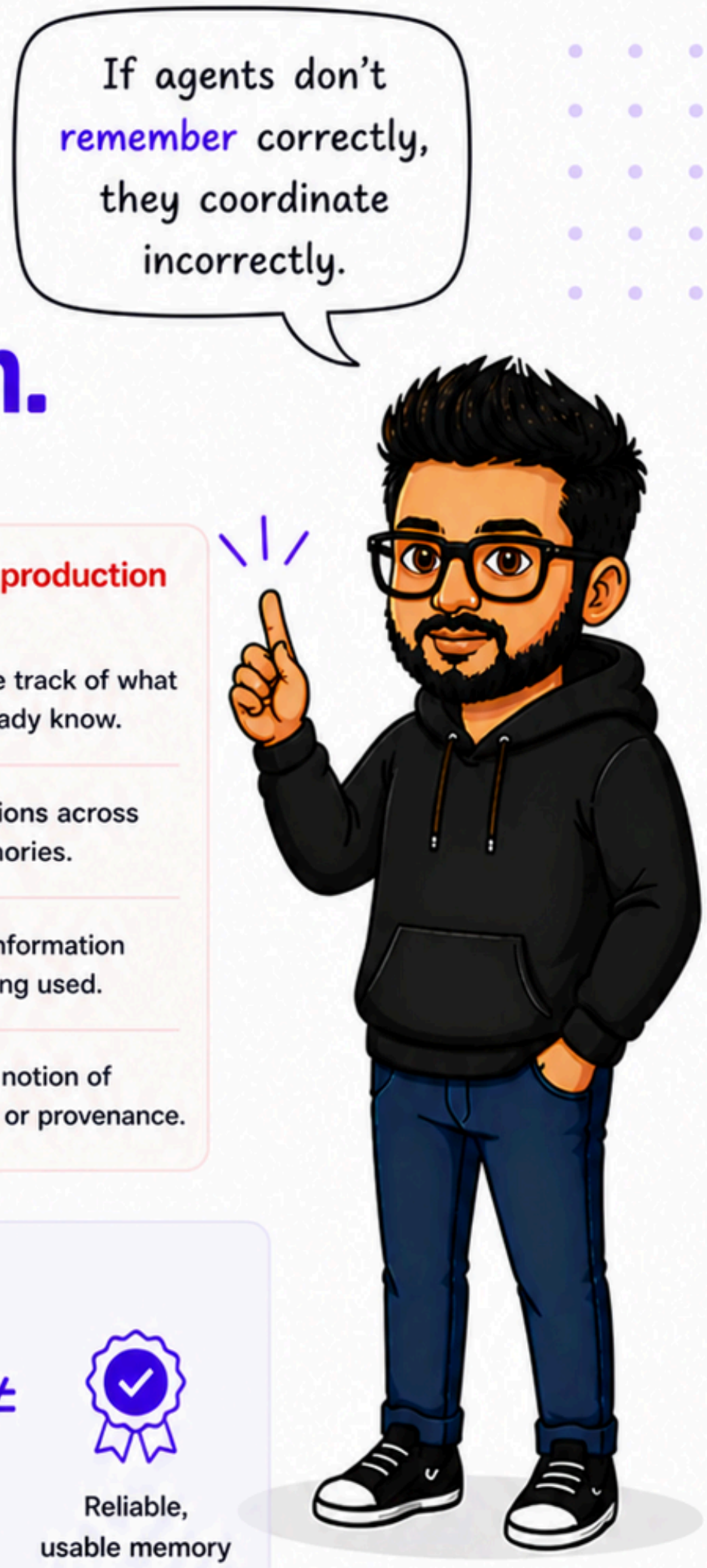
In production, diversity of models > size of model.

Save Comment Share Like

Swipe

@das-purnendu

Failure #3:
Memory is the actual hard problem.



- Multi-agent coordination needs **shared state**.
- "Shared state" today = **stuff the context window** and pray.
- Vector retrieval alone doesn't capture **who** learned **what**, **when**, or **how confidently**.

What breaks in production

- ✗ Agents lose track of what others already know.
- ✗ Contradictions across agent memories.
- ✗ Outdated information keeps getting used.
- ✗ No reliable notion of confidence or provenance.

Why this is hard

Raw data (Unstructured) ≠ Retrieved chunks (Context) ≠ Agent understanding (Internal state) ≠ Reliable, usable memory (What's needed)

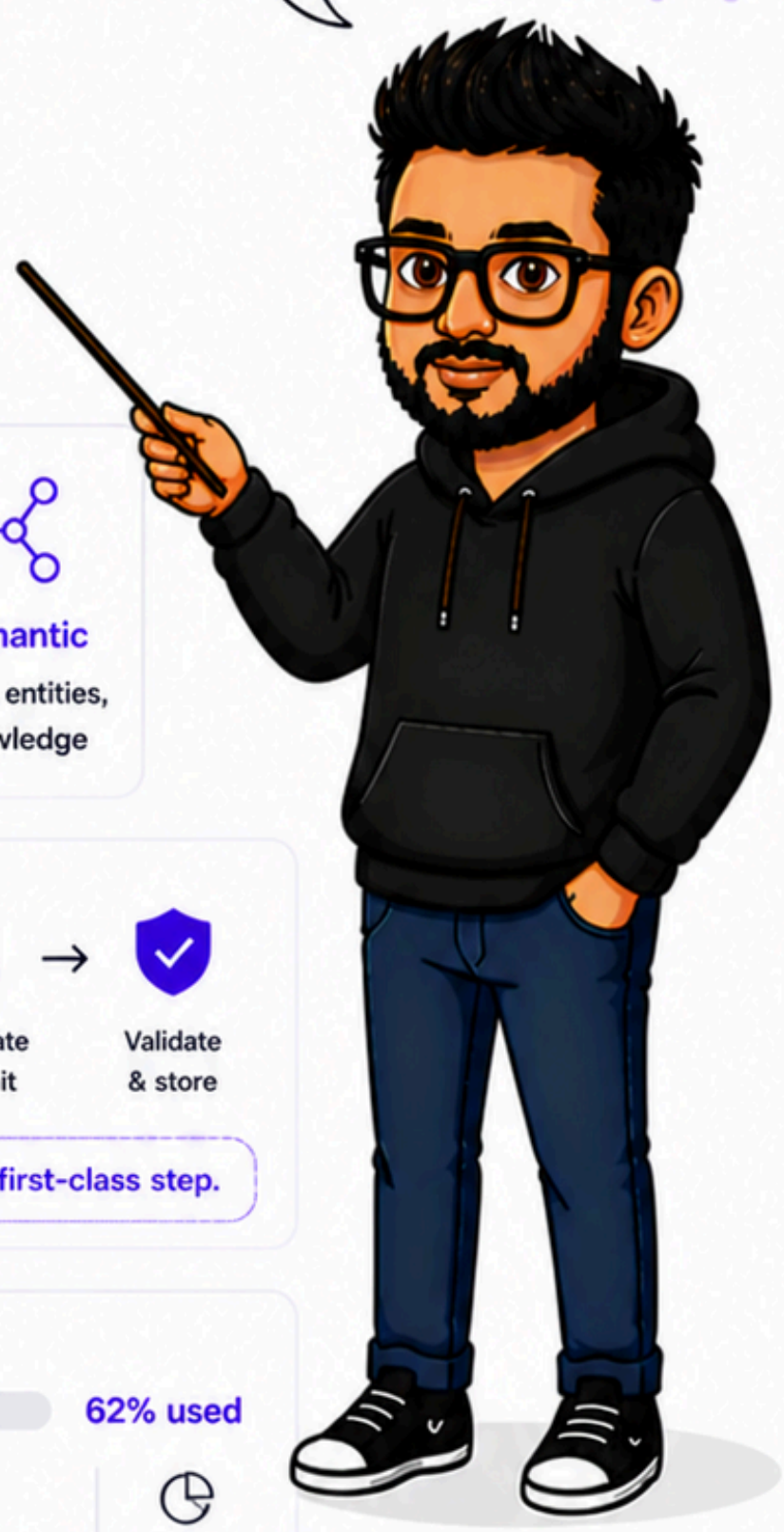
Memory isn't storage.
It's structured understanding over time.

8/10 SKILLS SERIES

@das-purnendu

What's emerging.

Better memory architecture > bigger models.



The architectures actually getting built in production:

1 Working vs Episodic vs Semantic

Different memory types for different jobs.

Working
Short-term state (active context)

Episodic
What happened, when, by who

Semantic
Facts, entities, knowledge

2 Explicit consolidation steps between turns

Agents don't just talk. They write, summarize, and commit to memory.

Agent conversation → Summarize & extract → Consolidate & commit → Validate & store

★ Turns memory from side-effect → first-class step.

3 Per-agent memory budgets + eviction policies

Unlimited context is a myth. Discipline wins.

Agent memory budget

62% used

Evict low utility · Decay over time · Prioritize recent + relevant · Track usage

✓ Good eviction policy > bigger context window.

The interesting AI engineering work in 2026 isn't model selection. It's memory architecture.

Save Comment Share Like

Swipe →

9/10 SKILLS SERIES

@das-purnendu

Key takeaways (Remember this!)

Good agents need good memory, good verification, and good design.

What makes agentic AI systems actually reliable in production.

1 Long chains need verification

Step-by-step confidence beats single-shot answers.

2 Cross-family verifiers win

Different perspectives catch different blind spots.

3 Memory is more than storage

It's structured understanding over time.

4 Design explicit memory workflows

Write → Consolidate → Validate → Store. Don't just chat.

5 Evaluate what matters

Production success = fewer correlated mistakes + better user outcomes.

Build agents that remember, verify, and learn — and you build systems people can trust.

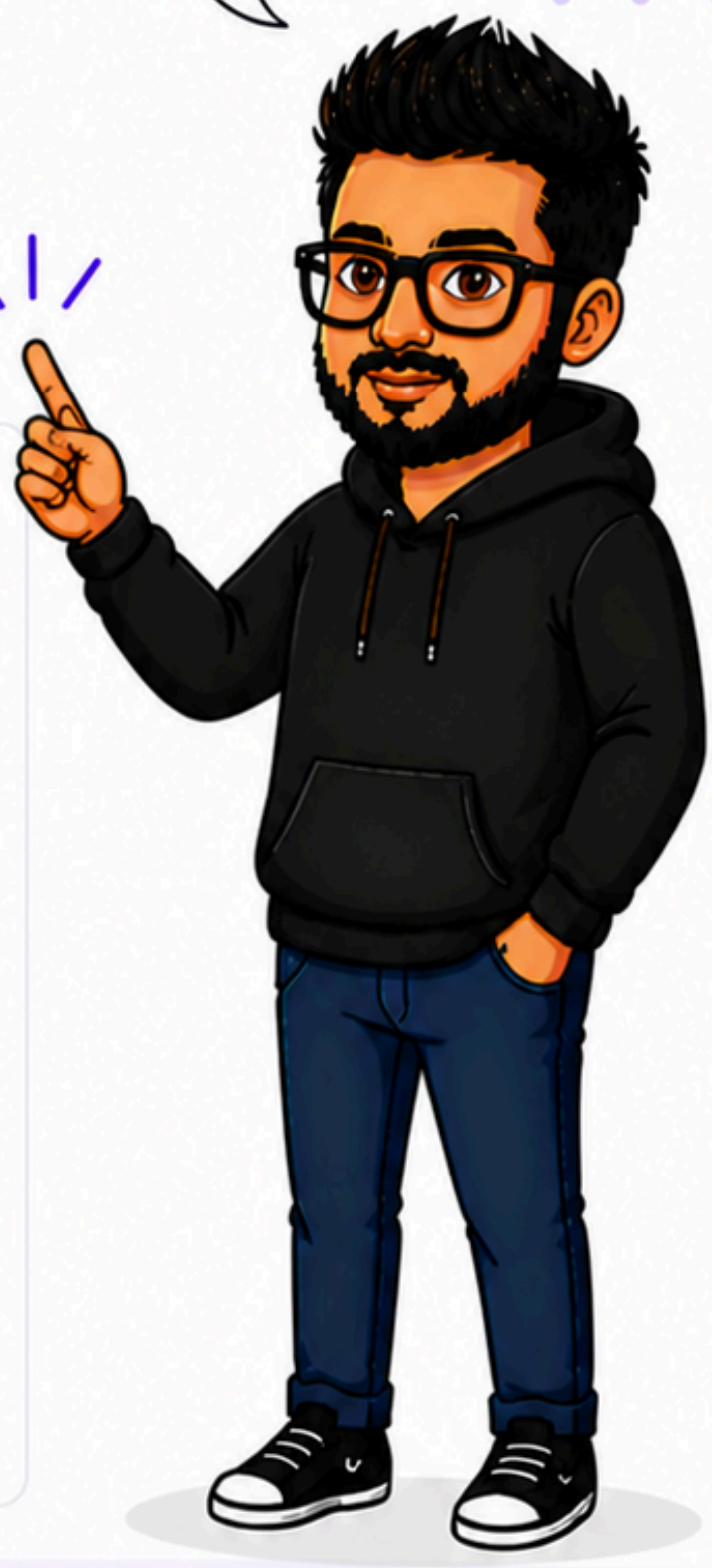
Save Comment Share Like

Swipe

@das-purnendu

BONUS: Tools & Resources to go deeper

Save this for later and build better agentic systems!



Use these to build, test and ship reliable multi-agent systems.

LangChain	Build agent workflows, memory, tools & orchestration.
CrewAI	Role-based multi-agent orchestration made simple.
Pinecone	Vector database for long-term memory & retrieval.
Weaviate	Open-source vector DB with powerful filtering.
LangSmith	Trace, debug & evaluate your agent runs.
arXiv	Stay updated with the latest research & papers.

PRO TIP:
Great agents come from great memory + verification + design.

Memory + Verification + Design = Reliable Agents

Save this carousel and level up your AI engineering!

Save Comment Share Like

Follow: @das-purnendu