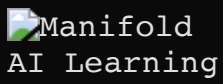


The System Design Question That Separates Prototype from Production Mindset

Agent system design edition

The Moment

"Design a customer support agent. 50K users, 200 req/hr, LLM bills per token."

Engineer talks about models, tools, retrieval...

I ask: **"What breaks first?"**

Long pause. "Probably... latency?"

That's when I knew they hadn't carried this system in production.

The Question Structure

> "Design an agent system for [domain].
> You have [scale constraint].
> Your LLM provider [cost/latency constraint].
>
> What breaks first?"

The last part is the entire interview.

Design-Focused Answer

"I'd use GPT-4 for quality. Tool calls for ticket lookup. RAG for knowledge base. Stream responses for lower latency."

Correct. Thorough. Missing the signal.

Production-Focused Answer

"First thing that breaks is cost. 200 req/hr × 2.5K tokens avg = 12M tokens/day = $270/day before retries. Need per-request and per-user token caps before I build anything else."

What Interviewers Listen For

→ **Failure-first thinking:** Named failure before being asked?

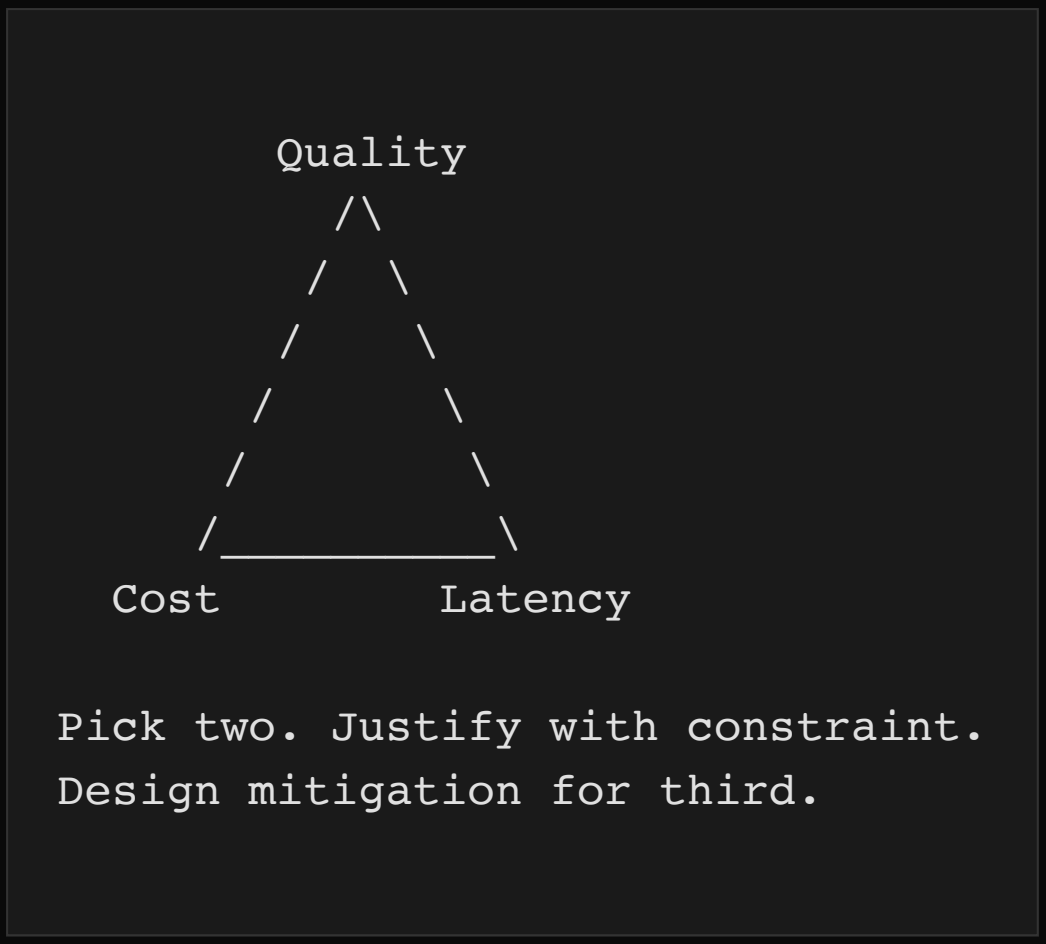
→ **Trade-off commitment:** Chose X over Y because of constraint Z?

→ **Cost reasoning:** Did the math on token spend?

→ **Instrumentation:** Mentioned observability before being asked?

Production-minded engineers think in failures. Demo builders think in features.

The Inevitable Trade-off



Production-minded engineers don't try to optimize all three.

Cost Reasoning (Staff Signal)

```
200 requests/hour
× 2,000 tokens in
× 500 tokens out
= 500K tokens/hour
= 12M tokens/day

Input:  12M × 0.6 × $0.01/1K = $72/day
Output: 12M × 0.4 × $0.03/1K = $144/day
Total:  $216/day = $6,480/month

Before retries. Before spikes.
```

If you didn't do this math, you haven't shipped this.

Mental Model: Constraint-Driven Design

Start with what **can't** work.

→ Cost budget: $10K/month → can't use GPT-4 for all traffic

→ Latency SLO: 1s p95 → can't do multi-hop retrieval

→ Quality floor: 90% CSAT → can't use smallest model

Architecture fits inside constraints.

Not the other way around.

How Design-Focused Engineers Answer

"I'd build a multi-agent system with a planner, retrieval agent, and response generator."

How Production-Focused Engineers Answer

"Latency budget is tight. Multi-agent orchestration adds 300ms. Using single agent with tool calls. Trade-off: lose modularity, gain latency. Acceptable because..."

Failure Anticipation Signal

Engineers who've operated production systems don't wait for the failure question.

→ Mention retry budget limits upfront

→ Talk about token cap guardrails before being asked

→ Design rollback before being asked "what if it breaks"

→ Instrument observability in the initial design

They've debugged this at 2am.

Interview Anti-Pattern

Building the happy path, then retrofitting failure handling.

This signals: demo experience, not production experience.

Engineers with a production mindset design guardrails first, features second.

What Actually Breaks First

→ **Cost:** Single retry loop user blows daily budget

→ **Latency tail:** p99 at 3s kills user experience

→ **Tool reliability:** External API flakes, agent fails

→ **Context overflow:** Long conversation exceeds window

Not "the database." Not "scalability."

Agent-specific failure modes.

Takeaway

Production-focused interviews test: Have you operated this?

The signal:

- → Start with constraints, not capabilities
- → Name failures before being asked
- → Make trade-offs, justify with numbers
- → Design guardrails first

"I've built this" ≠ "I've operated this"

Manifold
AI
Learning

Keep connected

More production-pattern thinking

community.nachiketh.in

When you're ready for structured learning

bootcamp.nachiketh.in