EXPERTS BY EXPERTS · HIRING PLAYBOOK

Agentic Engineering Interview Guide.

A sequential hiring guide for senior IT hires in 2026. Four phases, 21 questions, plus one optional take-home. Tests workflow maturity, not whiteboard algorithms.

45 min	4	21	+1
INTERVIEW SLOT	PHASES	CORE QUESTIONS	TAKE-HOME

For CTOs, engineering leads, and hiring managers who screen IT freelancers and senior engineers — and want to spot vibe-coding reflexes early.

ELEVATEX.DE

WHY THIS GUIDE EXISTS

This format does not test algorithms. It tests workflow maturity.

If you are still handing out whiteboard puzzles in 2026, you are testing whether the candidate would have been a good engineer in 2019 — not whether they operate agentically today. That is why this guide works with workflow narratives, live-task simulations on a real codebase, and self-reflection questions — not with isolated logic puzzles.

> *Most people have still not refactored their hiring process for agentic engineering capability. If you're giving out puzzles to solve, this is still the old paradigm.*
>
> ANDREJ KARPATHY · SEQUOIA AI ASCENT 2026

TABLE OF CONTENTS

Phase 01 Workflow narrative

What does a typical day look like?

Seven questions on daily driver, tools, subscriptions, skills, and review setup. No screen sharing, no live task. This is where it shows whether the candidate lives the workflow — or only knows it from reading about it.

OPENING QUESTION

„Walk me through what a typical workday looks like for you.

DURATION	QUESTIONS	SETUP REQUIRED
15 minutes	1 – 7	None

Phase 1 is the icebreaker. Candidates find their rhythm here and you get the strongest differentiation signals without any technical setup. A candidate who struggles in phase 1 will rarely surprise you in the live task.

QUESTION 01 How much code do you still write by hand on a typical day?

GREEN FLAGS

- Concrete number: *5–15 percent*
- Explains *which* kind of code they still write by hand — architectural decisions, critical business logic
- Mentions Karpathy's 80/20 → 20/80 flip or a similar concept
- Has built a deliberate routine, not random practice

RED FLAGS

- *"Still quite a bit"* or *"Most of it"* (over 50%)
- Cannot give a number
- Says 0% — signals a vibe-coding reflex with no oversight
- Does not understand the question or considers it unimportant

QUESTION 02 Which subscription tier do you run on your daily driver?

GREEN FLAGS

- Max 5x or Max 20x at Anthropic, Pro+ or Ultra at Cursor, or direct API access
- Names a rough monthly spend (over $100)
- Has blown through their quota at least once
- Justifies the tier choice with concrete workload

RED FLAGS

- Pro at $20 — *not senior in agentic 2026*
- *"My employer pays for it"* with no further detail
- Does not know what they pay
- Has never hit a quota

QUESTION 03 Describe your workflow for a new feature. Step by step — what do you do first?

GREEN FLAGS

- Mentions spec or tests *before* coding — but iteratively, not waterfall-style
- Explicitly uses Plan Mode (or an equivalent)
- Describes review between phases, not only at the end
- Mentions parallel agents or sub-agents for reviews
- Has a personal rule of thumb for spec depth (e.g. "spec sized for 1–2 days of work")

RED FLAGS

- Jumps straight to *"Then I ask Claude to build it"* (vibe-coding reflex)
- Spends 60+ minutes on the spec before any agent runs (waterfall reflex)
- No explicit planning step
- Accepts AI output without reviewing the diff
- Tests are written after the code, if at all

QUESTION 04 Which Skills or slash commands have you written yourself in your current project? Do you have a pattern for them?

REFERENCE — HOWIE LIU, AIRTABLE FOUNDER

Skills are *"the most important primitive"* in the frontier-agents world. His framing: picture Albert Einstein — generally highly intelligent, but he knows nothing about real estate. Give him the right briefing, a playbook, a manual, and he'll solve it. Skills are exactly that briefing. Anyone who treats Skills as a boilerplate library has missed the point.

GREEN FLAGS

- At least 1–2 self-written Skills
- Has a *structured pattern* for Skills (e.g. rules + checklist + guide per topic)
- Can describe concretely which problem a given Skill solves
- Mentions reuse across projects
- Sees Skills also as *onboarding tools*
- Understands Skills as a *contextualization layer* for generally intelligent models

RED FLAGS

- *"Haven't needed any yet"*
- Uses only default or downloaded Skills
- Has 1–2 Skills but no recognizable pattern
- Does not understand the concept
- Cannot recall the last time they wrote a Skill
- Describes Skills as a *"prompt collection"* or *"snippet library"*

QUESTION 05 Which data or secrets must never enter the agent context? How do you enforce that?

GREEN FLAGS

- Clear list: production credentials, customer data, infrastructure code (Terraform, Helm), PII
- Knows pre-tool hooks or comparable mechanisms to keep the agent out of critical directories
- Has a deliberate separation between code and infrastructure
- Recognizes adversarial agent behavior — and plans for it
- Mentions *.gitignore* discipline, separate test databases, log masking

RED FLAGS

- *"I don't pay close attention to that"*
- Has accidentally pasted secrets into prompts before
- Doesn't know pre-tool hooks or comparable safeguards
- Mixes infrastructure and code logic in the same agent context
- *"Surely the agent handles that safely on its own"*

QUESTION 06

Once the agent is done with the implementation — what does your review setup look like?

GREEN FLAGS

- Multiple review layers with clear roles: a second agent as reviewer, a PR bot like GitHub Copilot, a final human review
- Can explain why each layer does what it does (e.g. *"Codex catches structural inconsistencies, the human checks domain logic"*)
- Clearly understands: AI is good at structural pattern checks, humans at business logic and context
- Has a deliberate heuristic for when a second agent review runs and when it doesn't

RED FLAGS

- *"I just give it a look"*
- One agent does everything — implementation and review in the same run
- Doesn't know the concept of *"AI reviewing AI"*
- Considers review unnecessary if the agent wrote good tests

QUESTION 07

What is your standing instruction to the AI before you kick off an implementation?

GREEN FLAGS

- Has a clear standing instruction — e.g. *"the AI doesn't implement on its own; it proposes 2–3 options, I pick"*
- Uses visual aids: has flow diagrams, sequence diagrams, or Mermaid charts generated before accepting code
- Can show concretely how they prompt the agent for *option generation* rather than direct implementation
- Has their own templates or prompt snippets that they reuse regularly

RED FLAGS

- *"I just say what I need, and it does it"*
- Accepts the first implementation blindly
- Never asks for options, only finished solutions
- Does not see visual outputs (diagrams) as a tool

Phase 02 Setup-Demo + Live-Task

Show me your setup.

Three setup questions, then a real mini-task on a real codebase. Whoever lives the workflow shows it in the first 60 seconds. Whoever only knows about it stumbles here.

OPENING INSTRUCTION

„Now I'd like to see your setup. Share your screen briefly — terminal, IDE, AI tools, whatever you normally have open. Three short questions about it, then we move on to the actual task.

DURATION	QUESTIONS	SETUP REQUIRED
20 – 25 min	8 – 11	Screen share

Phase 2 is the proof of practice. The setup demo surfaces daily driver, configuration, and parallel agent instances. Then follows a 10–15-minute live task from four profiles (bug fix, refactor, greenfield, or adversarial take-home).

QUESTION 08 Which AI tools are running on your system right now, and which one is your daily driver?

GREEN FLAGS

- Names 2–3 tools specifically (Claude Code + Codex CLI + Cursor)
- Has a clear daily driver with reasoning
- Mentions parallel usage instead of relying on a single tool
- Explains why they use tool A for task X and tool B for task Y

RED FLAGS

- *"I just use ChatGPT"* with no further differentiation
- Knows only one tool
- Names Copilot as their primary agentic tool — shows a 2024 mindset
- Cannot justify tool choices

QUESTION 09 Show me your `CLAUDE.md` or `AGENTS.md`. How long is it, what's in it?

GREEN FLAGS

- File exists and is visible
- At least 100 lines, usually more
- Contains architecture notes, tool configuration, coding standards
- Updated regularly (version history visible or evident)
- Mentions custom Skills, slash commands, or pre-tool hooks

RED FLAGS

- *"I don't have one"* or *"Haven't gotten to it yet"*
- File is 5–10 lines or a stock template
- File hasn't been touched in weeks
- Candidate doesn't know what the file is for

QUESTION 10 Show me how you open a second or third agent instance. What does your parallel-work setup look like?

GREEN FLAGS

- Opens 2–3 instances in under 60 seconds
- Has an established setup: terminal grid, multiple CLI sessions, or Git worktrees
- Knows *worktrees* as a concept and has actively tested them
- Can explain when they work in parallel and when they don't
- Mentions conflict avoidance (separate branches, cleanly scoped tasks)

RED FLAGS

- Only has one instance, doesn't know how to open a second
- *"Don't need it"* with no reasoning
- Long setup time signals rare usage
- Doesn't know worktrees or treats them as specialist knowledge — a clear senior disqualifier in 2026

LIVE TASK SIMULATION · 10 – 15 MIN

You solve it with your normal setup.

"I'll observe and ask a single accompanying question. Pick one of the tasks below, depending on the candidate profile." — These tasks are *deliberately not isolated algorithms or whiteboard puzzles*. They are realistic mini-projects with ambiguity, context demands, and architectural decisions — exactly what matters in 2026.

FOUR TASKS — PICK BY PROFILE

TASK	FOCUS	WHEN TO USE
A — Bug reproduction with fix 10 MIN	Test-driven bug fixing, Plan Mode discipline, review behavior	Default for backend senior hires
B — Refactor with spec 10 MIN	Spec discipline, acceptance test, diff review	Frontend / mobile senior hires
C — Greenfield feature with architecture 10 MIN	Clarifying questions, architecture decisions, out-of-scope definition	Tech-lead hires or when architectural strength is central
D — Adversarial take-home 60 MIN	Build-and-break: candidate builds securely, agents try to break it	Senior architect or security-relevant hires; take-home only

A Bug reproduction with fix

> *"I'll give you a small API with two endpoints. Users report: on the third pagination page, the same items sometimes come back. Reproduce the bug, find the root cause, write a fix with a matching test. 10 minutes."*

WHAT TO WATCH FOR

- → Do they *write a reproduction test first* that catches the bug? Or jump straight into the code?
- → How do they use the agent — as a reproduction helper (good) or directly as a fix generator (red flag)?
- → Did they use Plan Mode before changing anything?
- → Do they read the test output thoroughly, or accept the first green signal?

B Refactor with spec

> *"Here's a view component, about 220 lines, grown across multiple iterations. Refactor it so business logic is separated from the view and tests become possible — without changing the UI. Write down briefly what you define as the goal, then refactor. 10 minutes."*

WHAT TO WATCH FOR

- Do they first write a tight spec for the refactor goal (1–2 sentences: what changes, what doesn't)?
- Do they define an acceptance test (e.g. *"all existing snapshot tests must keep passing"*)?
- Do they have the agent *propose options*, or implement directly?
- How do they handle the *spec-waterfall trap* — do they start doing after 2 minutes, or stay 8 minutes in the spec?
- Do they review diffs before accepting?

C Greenfield feature with architecture decision

> *"We're building a small banking app. You'll add a simple feature: users can forward invoice PDFs by email to a project-specific address — the app parses them and stores them as pending payments. How would you set up the data model and service architecture? Then implement the data model and a first service stub. 10 minutes."*

WHAT TO WATCH FOR

- Do they ask *clarifying questions before starting* (context awareness vs. vibe-coding reflex)?
- Do they have the agent propose 2–3 architecture options before deciding?
- Do they cleanly separate *data-model decisions (their job)* from *boilerplate generation (agent's job)*?
- Do they recognize what *explicitly should not be implemented* in 10 minutes — and define it as out of scope?

D Adversarial take-home (60 minutes)

REFERENCE — ANDREJ KARPATHY, SEQUOIA AI ASCENT 2026

„Hiring has to look like — give me a really big project and see someone implement that big project. Like let's write a Twitter clone for agents and then make it really good, make it really secure. And then I'm going to use 10 codex agents to try to break your website. They should not be able to break it."

"In 60 minutes, build a small, clearly scoped system with your normal setup — e.g. an API with authentication, rate-limiting, and a persistence layer. Make it as secure as you can. When time's up, I'll point several agents at the system that try to break it — bypass auth, exploit race conditions, find validation gaps."

DURING THE BUILD

- Do they build *defensively from the start*, or does security come later?
- Do they use agents for security reviews during the build, not only for implementation?
- Do they specify threat models explicitly before coding, or leave them implicit?
- Do they have their own Skills or patterns for rate-limiting, input validation, auth hardening?

DURING THE ADVERSARIAL PHASE

- How do they react when gaps are found? Defensively defensive, or analytically?
- Can they explain *why* a particular gap arose?
- Do they patch only the symptom, or the underlying class of problems?

How to use: This task isn't suited for 10-minute live slots — it's a take-home or a 60–90-minute onsite slot with a debrief afterwards. It's the sharpest hiring probe available for senior architects and security-relevant roles, because it combines build and break in a single setup.

QUESTION 11 While the work is in progress: how do you know right now that the agent is on the right track?

GREEN FLAGS

- Actually reads the plan or the diffs
- Has clear intermediate checks (running tests, code reads)
- Actively stops when something doesn't fit
- Doesn't blindly trust the agent, not even on simple tasks

RED FLAGS

- *"I'll check at the end"* with no intermediate checks
- Relies only on green tests
- Cannot explain how they would spot the agent going off track

Phase 03 Failure narrative

When did the agent build nonsense?

The most honest part. A candidate who breezes through without self-reflection is either under-practiced or running on Dunning-Kruger.

OPENING QUESTION

„Tell me concretely about a moment when the agent built complete nonsense — something you caught. When was it, what was the task?

DURATION	QUESTIONS	MODE
10 minutes	12 – 15	Narrative

Four questions on misfires, withdrawn PRs, saying no to the AI, and approve-drift on large PRs. Concrete stories rather than theory — anyone who cannot name concrete cases doesn't live the workflow in practice.

QUESTION 12 How did you notice it was going wrong?

GREEN FLAGS

- Concrete story with context (project, task, symptom)
- Caught via test, code read, or logic check — *not* only in production
- Can name the trigger
- Built a heuristic out of it

RED FLAGS

- *"That doesn't happen to me"* — too little practice or no self-reflection
- Story is vague and without detail
- Caught only via a production bug, not in review
- Blames the tool, takes no ownership

QUESTION 13 Have you ever withdrawn one of your own pull requests because you realized after the fact that you didn't actually understand the agent's output?

GREEN FLAGS

- Yes, with a clear story
- Can explain *why* understanding came only later
- Drew a consequence from it (different workflow, more reviews)
- Shows self-reflection without finger-pointing

RED FLAGS

- *"No, never."* — either too little practice or Dunning-Kruger
- Story is staged or non-specific
- Doesn't see this as a problem
- Defends blind acceptance

QUESTION 14 When was the last time you said *no* to the AI? What did it suggest that you rejected?

GREEN FLAGS

- Concrete example, within the last days or weeks
- Reasoning was architectural or domain-driven (not stylistic)
- Translated it afterwards into a Skill rule or a *CLAUDE.md* note
- Sees *saying no* as a core senior responsibility — not an exception
- Recognizes overshooting proactivity as a typical agent-EQ failure and has heuristics against it

RED FLAGS

- *"I usually accept what it does"*
- Cannot recall a concrete case
- Justifies saying no only with style preference
- Reads the question as a test of trust in the AI, not as a test of their own responsibility
- Has no awareness for *"the agent does something technically correct that is socially or strategically wrong"*

QUESTION 15 How do you protect yourself today from approve-drift on larger PRs?

GREEN FLAGS

- Knows the term or the concept
- Deliberately keeps PRs small
- Uses automated review agents as a first filter layer
- Uses an *issue-forking pattern*: when a new topic comes up in an agent session, they have the agent create a separate issue rather than bloat the current PR
- Has a size limit for PRs that they don't exceed

RED FLAGS

- Doesn't understand the risk
- *"I just look it over"* with no structural safeguard
- Has no size limit for PRs
- Relies solely on human reviewers

Phase 04 Calibration & signal check

Where do you draw the line?

Six calibration questions — not to test what the candidate knows, but to see where they draw their lines.

OPENING INSTRUCTION

„Last phase. I'll ask you a few calibration questions — not to test what you know, but to see where you draw your lines.

DURATION	QUESTIONS	MODE
10 minutes	16 – 21	Reflective

This is where the guide's sharpest senior tests live. Question 17 (AI vs. human reviewer) is the sharpest seniority test — anyone who answers in black and white is not senior. Question 19 (persistence-curation) is the most hidden — Liu calls giving up after the first attempt the most common mistake of practitioners.

QUESTION 16

Where do you trust the AI completely? Where not at all?

GREEN FLAGS

- Clear list of what they delegate: boilerplate, test generation in clear modules, refactoring, bug reproduction, documentation
- Clear list of what they don't delegate: critical business logic, architectural foundations, security-sensitive code, regulated areas
- Has a heuristic, not just a gut rule
- Acknowledges that the line keeps moving

RED FLAGS

- *"Pretty much everything"* — vibe-coding reflex
- *"Nothing really"* — 2024 mindset
- Cannot name examples
- Black-and-white answers, no differentiation

QUESTION 17 ★ CORE SENIORITY TEST

Where is the AI better than a human reviewer? Where is it clearly worse?

GREEN FLAGS

- *Better:* structural consistency across many files, fast detection of pattern breaks, security patterns, missing edge cases in tests, blind spots in refactoring
- *Worse:* domain context (why was this decision made back then?), business-logic correctness, team conventions, long-term maintainability beyond the current file
- Has a differentiated stance — neither "AI replaces the reviewer" nor "AI adds nothing"
- Uses both strengths deliberately

RED FLAGS

- *"AI is better at everything"* — vibe-coding reflex
- *"I prefer to trust humans"* — 2024 mindset
- Cannot name an area where AI is clearly strong
- Cannot name an area where AI is clearly weak

QUESTION 18 Which 2024 best practice do you now consider wrong?

GREEN FLAGS

- Names a concrete practice (e.g. *"writing tests is expensive"*, *"PRs should be small because humans review them"*, *"code comments are mandatory"*, *"one task per branch"*)
- Explains why this has shifted
- Has actively changed their own practice

RED FLAGS

- *"It all still holds, basically"*
- Generic answer with no substance
- Doesn't understand the question
- Rejects the question as speculative

QUESTION 19 ★ HIDDEN TEST Tell me about a workflow you got working after several attempts. What was your staying power?

REFERENCE — HOWIE LIU, AIRTABLE FOUNDER

The most common practitioner mistake: they one-shot something, it isn't quite as profound as they hoped, and they give up. The agents are powerful enough to solve almost anything you want. The question is whether you invest the time, coaching, and curation to get them there. *Persistence-curation* is its own senior skill in 2026 — and not everyone has it.

GREEN FLAGS

- Concrete story: workflow X didn't work on the first try, better on the second, solid on the third
- Can describe *which iterations* were needed
- Has their own threshold for when to keep pursuing a pattern and when to drop it
- Sees investing in coaching and curation as core senior work

RED FLAGS

- *"If it doesn't work on the first try, the tool isn't mature"*
- Cannot remember a single case
- Describes only successes, never failed attempts and corrections
- Confuses persistence with stubbornness — no reflection on what gets adjusted

QUESTION 20 Who do you read or follow in the agentic space? Who taught you something recently?

GREEN FLAGS

- At least 2–3 names: Karpathy, Steinberger, Willison, Cherny, Tunguz or equivalent
- Mentions a concrete idea they've adopted
- Has their own sources routine (newsletters, Twitter list, GitHub stars, Substacks)

RED FLAGS

- Doesn't know any names
- *"I look at tutorials when I need something"*
- Follows only vendor marketing
- Has no personal learning scaffold

QUESTION 21 If you had to bring a mediocre engineer on the team up to speed on agentic — what's your first recommendation?

GREEN FLAGS

- Concrete pedagogical answer: a first small MVP setup with security rules, a pair session, one concrete tool
- Mentions a mindset shift, not just tool setup
- Understands: the methodology is what gets learned, not the tool
- Mentions structured knowledge sharing across the team — channel for wins-and-losses, workflow show-and-tells
- Shows multiplier ability

RED FLAGS

- *"They can just read up on it"*
- Recommends tutorials or workshops as the primary path
- Doesn't see that coaching matters more than the tool
- Wouldn't pair-program themselves

APPENDIX A

Scoring notes

Scoring follows a simple heuristic. Important: no number is absolute — context and role decide which phases weigh more.

15 / 21
Senior 2026 minimum bar
At least 15 of 21 questions should land clearly in the green-flag range for a senior hire in 2026.

5+
Stop signal
More than 5 red flags is a stop signal, regardless of the rest.

P1
Icebreaker phase
Phase 1 (workflow narrative) is the icebreaker — candidates find their rhythm here. A weak phase 1 rarely turns into a strong live task.

P2
Proof of practice
Phase 2 (setup + live task) is the proof of practice. Whoever lives the workflow shows it in the first 60 seconds of the setup. Whoever only knows about it stumbles here.

P3
Honesty phase
Phase 3 (failure narrative) is the most honest part. A candidate who breezes through without self-reflection is either under-practiced or running on Dunning-Kruger.

★
Question 17 + Question 19
Q17 (AI vs. human reviewer) is the sharpest seniority test in the guide — anyone who answers in black and white isn't senior. Q19 (persistence-curation) is the most hidden test.

APPENDIX B

Sources for the references

- **Andrej Karpathy** — *From Vibe Coding to Agentic Engineering.* Sequoia AI Ascent 2026 · April 2026
- **Greg Brockman** — *Why Human Attention Is the New Bottleneck.* Sequoia AI Ascent 2026 · May 2026
- **Howie Liu** — *Making $ with AI Agents (Hyperagent-Showcase).* Greg Eisenberg's Startup Ideas Podcast · May 2026
- **Boris Cherny** — *Why Coding Is Solved, and What Comes Next.* Sequoia AI Ascent 2026 · May 2026

As of May 2026 — updated continuously as practice shifts.

HIRE SMARTER · HIRE FASTER · HIRE VETTED

Vetted IT freelancers. In 48 hours.

ElevateX connects companies with vetted freelance developers, engineers, and IT experts. Scale your team fast with the right talent — and use this guide as a filter for your own hires.

Hire Freelancers →

MATCH	REPLY	VETTING
In as little as **48 hours**	Within **24 hours**	100% verified by experts

ELEVATE X

elevatex.de