

The AI-allowed interview kit

Detection does not work well enough for stakes that end a career. Assume every candidate uses AI, say so in the invitation, and ask them to defend what they produce. Three formats survive that. Here they are, ready to run.

1. The sentence in the invitation

"Use whatever you would use at work, including AI assistants. We will ask you to explain and defend what you produce."

Nothing is banned; the bar has moved. The second sentence is the whole interview.

2. The broken-service exercise: brief for the candidate

- Format:** 40 minutes, screen shared, any tool allowed. Then 20 minutes of questions.
- You get:** a small working service in [language] with tests that pass. One behaviour is wrong under load: [one sentence, e.g. "a retry that duplicates writes" or "an off-by-one under concurrency"]. Find it and fix it.
- We look at:** how you read code you have not seen, what you check before you trust a suggestion, and whether you can explain the fix afterwards.
- We do not look at:** how fast you type, or whether you used a model to get there.

Building it (for the panel)

- Make the bug behavioural, not syntactic. A model finds a syntax bug in seconds; a behaviour under concurrency needs understanding.
- Same service for every candidate at the level. Build 2 or 3 and rotate them; a week of work lasts a year.
- Test the environment the day before. A broken setup is the interviewer's failure, and it eats the 40 minutes.

3. The 4 follow-ups a pasted fix cannot survive

QUESTION	WHAT A REAL UNDERSTANDING SOUNDS LIKE	WHAT PASTING SOUNDS LIKE
Why does your fix work?	Names the mechanism: "the two writers both read the counter before either wrote it".	Restates what the fix does, not why the bug existed.
What else touches this code path?	Traces callers, names one that might now behave differently.	Silence, or "I would have to check".

QUESTION	WHAT A REAL UNDERSTANDING SOUNDS LIKE	WHAT PASTING SOUNDS LIKE
What would break if the load doubled?	Picks a component and says why it saturates first.	"It should scale", with nothing attached.
How would you have caught this before production?	A specific test or metric, and where it would run.	"More testing".

Someone who understands the fix answers all 4. Someone who pasted it runs out around the second.

4. The other 2 questions

- **The rejected suggestion.** "What did the tool suggest that you did not use, and why?" Pasting the first output cannot answer it. Thinking with a tool can.
- **Their own code, line 40.** Open a piece of their prior work or the take-home. "You retry 3 times with a fixed delay. Why 3, and why fixed?" Anyone who did not make the decision runs out within 2 follow-ups.

5. The scorecard line when the answers do not hold

Write this: "Could not explain the behaviour of code they submitted. Asked why the fix works and what else touches the path; neither answer held on the second follow-up."

Never this: "The detector flagged it" or "probably used ChatGPT". The first is a finding a debrief can use and a rejection email can cite. The second is an accusation.

If you suspect during the round: go deeper on the work in front of you. "Let me understand your approach here", then 3 specific questions. That resolves it almost every time, on evidence. An engineer who uses the tools well is more valuable than one who refuses to; the interview's job is to tell using them well from using them blindly.