

Software Engineer interview checklist

A typical process is an online coding assessment, one or two data structures and algorithms rounds, a system design round for experienced candidates, a hiring manager or behavioural round, and an HR discussion.

10 QUESTIONS TO PRACTISE OUT LOUD

1. When would you use an array and when would you use a linked list?

What it checks: Whether you understand how memory layout drives time complexity for access, insertion and deletion, not just textbook definitions of both structures.

2. Tell me about a code review where you strongly disagreed with the feedback.

What it checks: Whether you can separate ego from code, argue with evidence and accept a decision, which is essential for working in shared codebases.

3. What is your current CTC, and what are you expecting?

What it checks: Whether you know your worth for the level you are interviewing at and can discuss compensation calmly, including fixed, variable and stock components.

4. How does a hash map work internally, and what happens on a collision?

What it checks: Whether you know why lookups are $O(1)$ on average, what makes them slow in the worst case, and how resizing works behind the scenes.

5. Walk me through a production incident you handled.

What it checks: How you behave under pressure: triage, communication, mitigation before root cause, and whether you drive a blameless follow-up that prevents repeats.

6. Are you open to relocating, or to working hybrid from our Hyderabad office?

What it checks: Your flexibility on location and work mode, and whether any constraints you state are genuine and clearly explained up front.

7. What is the time complexity of binary search, and when can you use it?

What it checks: Whether you can reason about complexity and recognise when a problem has a sorted or monotonic property that allows binary search beyond simple arrays.

8. Tell me about a technical decision you made that turned out to be wrong.

What it checks: Self-awareness and judgement: whether you can admit a mistake, explain why it seemed right at the time, and show how you corrected course.

9. Why are you looking to leave your current company?

What it checks: Whether your reasons are forward-looking and professional, and whether the same issue would make you leave this company too.

10. Design a URL shortener.

What it checks: Your system design approach: clarifying requirements, estimating scale, choosing storage and key generation, and handling read-heavy traffic with caching.

TIPS FOR THIS INTERVIEW

- Practise problems by pattern, such as two pointers, sliding window, BFS and DFS, heaps and DP with memoisation, instead of memorising individual questions.

- Always state time and space complexity before the interviewer asks, and talk through a brute-force approach before optimising it.
- For system design, follow a fixed order: requirements, rough scale estimates, API, data model, high-level diagram, then bottlenecks and trade-offs.
- Prepare two production stories with numbers: an incident you handled and a design decision you made, including what you would change now.

THE DAY BEFORE

- ☐ Confirm the time, venue or video link and the interviewer's name
- ☐ Plan your route and leave time for traffic, or test your camera, microphone and internet
- ☐ Keep copies of your resume, ID proof, certificates and photographs in a folder
- ☐ Iron your clothes and charge your phone and laptop
- ☐ Do one final run-through of your introduction, then stop and rest

ON THE DAY

- ☐ Arrive 15 minutes early, or log in 5–10 minutes early for a video call
- ☐ Greet everyone politely, including the receptionist or coordinator
- ☐ Listen to the full question, pause, then answer in a clear structure
- ☐ Take brief notes of names and key points
- ☐ Ask your prepared questions and confirm the next steps before you leave