codility

A Practical Guide to Running

Better Technical Interviews

Tech Recruiting Strategies for Hiring
Managers and **Interviewers**

codility.com | UK: +44 (208) 970 7868 | US: +1 (415) 466-8085

Introduction

Goal: To conduct better in-person interviews to understand the backgrounds, qualifications, and aspirations of candidates applying to your technical roles.

Approach: Build an intentional recruiting process, dive deeper with insightful questions, and ensure every interaction is candidate-centric.

You've got a ton of things on your plate, and sometimes prepping for your next developer interview falls through the cracks. But winging phone interviews and conducting half-hearted whiteboard exercises has an impact on whom you eventually hire (or don't). **So how can you avoid these pitfalls and master the technical interview?** In this ebook we'll show you how to make a great impression on your candidates and simultaneously create a clear picture of their capabilities as potential team members.

Codility works with **over 1,000 businesses**, and we've sent over five million tests on behalf of our clients. Over the years we've learned how to quickly assess programmer expertise and relative suitability for every open position our clients are looking to fill. Codility created the tech assessment movement and **we've tested 3x more candidates** than newer entrants in the space. The insights we've gained through this experience form the content of this ebook.

over
1,000 clients
ranging from startups
to global enterprises.

Facilitating over
5 million
candidate
assessments.

TomTom · Samsung · PayPal · accenture · Rakuten · zalando · LiveRamp · amazon.com · FLATIRON · Barclays · Deloitte. · ocado · intel · EA · citi · icims · twitch

6 Steps for Better Technical Interviewing

1. Ensure Your Overall Recruitment Process is Sound

Having a solid plan before in-person interviews will set your hiring team members up for success.

2. Ask the right technical questions

Choosing the right topics to hone in on will give you key insights.

3. Focus on the candidate's favorite tool

Diving deeper into tool preferences shows you candidates' work style.

4. Observe how your candidate responds to feedback

Focusing on their collaboration style determines team fit and growth potential.

5. Dig-in to their dev-related activities outside of work

Uncovering what programmers do in their personal time reveals what drives them.

6. Ask about issues with their current team or project

Surfacing previous or current challenges highlights solution-oriented coders.

1. Ensure Your Overall Recruitment Process is Sound

Everything that happens towards the start of your process has a big impact on the success of the technical interview.

You need your sourcing efforts to attract the right sorts of candidates and **you need an efficient screening process** to help identify which candidates merit further interview time and resources. Looping in the recruiting team to manage the screening process will free up you and your team to focus on more in-depth stages.

When it comes to screening, there are typically three scenarios:

No Technical Screening (the worst option):

Many companies rely too heavily on resumes or online profiles and don't utilize technical screening at all. There is little correlation between candidate technical skills and their ability to create a standout resume, LinkedIn, or GitHub profile. Junior developers often have visually appealing resumes with impressive details about their skill set, but might lack the experience of more senior developers. Conversely, some of the best developers have terrible resumes. Without a proper screening mechanism, you'll advance unqualified candidates and lose out on ones that might be a great fit.

Homemade Technical Screening (better than nothing):

Many companies write a few problems to share with candidates as "homework." This can work fine at a small scale, but once you have a decent number of candidates, plagiarism and benchmarking become an issue. You'll need to invest more and more time reviewing each candidate's solutions by hand and checking for red flags. Sometimes candidates even complain that homemade coding problems can take hours to solve (and grading can take even longer).

A Screening Platform (the most reliable option):

Implementing a code assessment tool allows you to see whether candidates have the basic coding skills you require before inviting them to more time-consuming interviews. Online coding tests are automated and enable both technical and non-technical team members to identify the strongest candidates. At the same time, the platform is designed to deliver a great experience to the candidate.

2. Ask the right technical questions

Your goal when interviewing candidates is to gauge whether they can perform the responsibilities for your open role.

Make sure your interviewers are armed with the right questions that touch on candidate knowledge, skills, and aptitude to learn new technical concepts.

Here are three topics to focus on:

Scalability

Ask candidates to solve a simple programming problem that has multiple solutions with varying simplicity and scalability trade-offs, like computing prefix sums of a sequence of numbers. Then assess whether candidates notice the scalability aspect and whether they can deliver the working solution with better scalability. Follow that up by asking, "can it possibly be done better?" This question opens up a discussion about possible optimizations and can sometimes go down to the hardware level. It also touches on intrinsic limitations of computing machinery, which greatly illustrates whether candidates can apply theory to a concrete problem.

Aliasing

Ask candidates to solve a simple programming problem that involves pointers and references, like manipulations on single-link lists. Even though programmers rarely deal with direct list manipulation nowadays, this exercise shows whether candidates can deal with real-life scenarios where multiple levels of indirection occur, which is common in data models.

Recursion

Understanding and handling recursion is a big indicator of programmer fitness. For some jobs, this skill is crucial. But even for jobs where it's not, if a programmer knows how to do it, it shows that they've been doing mental push-ups and sit-ups for a long time—a sign that they're driven. The classic question here is to take a simple recursive function and ask candidates to rewrite it into an iterative one.

3. Focus on the candidate's favorite tool

Your technical interview should emphasize what the candidate knows best.

For a great engineer, learning a new tool, language, or technology is easy. What you should really focus on when hiring programmers is how well they know their favorite tools. **Do they only know enough to get by, or do they also understand the underlying mechanics of the tool?** This can tell you a ton about a candidate, including how they learn, how curious they are, how they communicate, and whether they'll be able to adopt the tools your team uses.

Example questions:

Tell me about your favorite tool and what makes it great to work with?

What is the main concept of this tool?

How does it work under the hood?

What are some of the drawbacks?

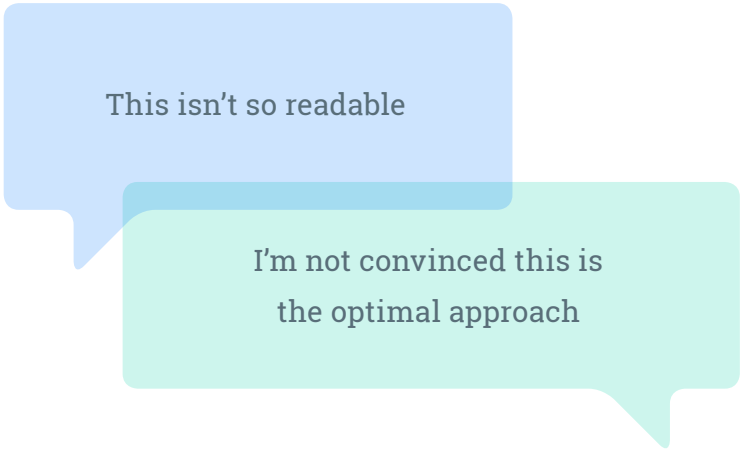
4. Observe how your candidate responds to feedback

No team is exempt from conflict, so use the technical interview to see how your candidates react to feedback on their ideas and work.

Try choosing something controversial, or take a stance on something they've provided to see how they consider an alternate perspective. Candidates might say, "Well, this is just how we do it" or "That's not a problem anyone else has had." These are red flags. Better candidate responses might be, "Can you explain more?" or "How could I make it better?" **This shows that the candidate is open to a discussion and wants to learn**. It also demonstrates that the candidate is focused on collaborating around an issue instead of preserving their way of thinking. Curious and humble programmers will take you far, while programmers that deflect criticism instead of learning from it will drag you down.

Example questions:

Take a sample of their code from their coding test or a project they've shown you and criticize it. Say, "This isn't so readable" or "I'm not convinced this is the optimal approach" and see how they respond.



5. Dig-in to their dev-related activities outside of work

To understand how your candidates can help drive development on your team, you need to learn more about what they do outside of their current work projects.

Programmers who code on their own time are active learners and often can juggle multiple initiatives and deadlines well. **Completing personal projects also shows that they require little micromanagement to be successful**. But most importantly, you get a better understanding of candidates' real passion. Perhaps they use legacy code at work, but on their own time they use modern options. Candidates love being asked about their passion projects so it's a great way to improve candidate engagement as well.

Example questions:

What is your favorite open source project? Have you contributed to it?

What's the most exciting project outside of work you've spent time on recently?

Where do you want your career to go? How do you think learning new technologies will help you get there?

6. Ask about issues with their current team or project

There's a reason programmers are interviewing with you. Ask what attracts them to your open role and your team specifically, and discuss any issues or concerns they've had in their current or previous teams and projects.

If the candidate is open to discussing what they're looking for, **you can more easily highlight similarities and differences between their expectations and the role you're looking to fill**. Have your recruiting team sell candidates on the opportunity and screen them for desired personality traits to do a quick check for culture fit. Remember, once the technical interview is completed, the candidate is going to have a stronger opinion of whether they want to work with your team, so you should invest some time in giving them the information they need to make the right decision.

Example questions:

What were some of the challenges you experienced in your last role?

What are the most important traits of your dream company?

What do you think makes a great development team great?

codility

01