TECH RECRUITMENT 101

How to Screen Tech Candidates on the First Call

A practical scorecard for motivation, fit, logistics, communication, level, and technical reality checks

PART OF THE *TECH RECRUITMENT 101* SERIES

How to Screen Tech Candidates on the First Call

A practical scorecard for motivation, fit, logistics, communication, level, and technical reality checks

CONTENTS

How to Screen Tech Candidates on the First Call

Introduction

I wrote this book because I have seen the same problem too many times: the first call is treated as a warm-up, and all the real judgment gets postponed until later. Then the process slows down, the hiring team is still unclear, and everyone acts surprised.

From my years working across engineering, management, and hiring, I have found the opposite works better. The first call should not answer everything, but it should answer enough. Enough to spot obvious mismatch, weak motivation, inflated profiles, and the candidates worth moving forward quickly.

That matters even more now. Tech hiring is noisy. Candidates are more coached, more polished, and sometimes more artificially polished than before. At the same time, recruiters are under pressure to move faster and still find better signal. LinkedIn's Talent 2026 research (https://news.linkedin.com/2026/LinkedIn-Research-Talent-2026) shows both pressures clearly, and Greenhouse's 2025 AI in Hiring report summary (https://www.greenhouse.com/newsroom/an-ai-trust-crisis-70-of-hiring-managers-trust-ai-to-make-faster-and-better-hiring-decisions-only-8-of-job-seekers-call-it-fair) makes the trust problem hard to ignore.

In this book, I will show you the six-part scorecard I use on first calls: motivation, fit, logistics, communication, level, and technical reality. It is simple on purpose. You do not need a grand

theory of talent, or a workflow diagram that looks like it was sold at enterprise-software prices. You need a repeatable way to hear what is real.

If you are reading other books in this series, this one sits in the practical middle: less about sourcing strategy, more about what to do once a candidate is in front of you. The goal is straightforward. By the end, you should be able to run a tighter first call and turn it into a real decision point.

What the first call is actually for

The first call is not a mini technical interview, a guided tour of the résumé, or a vague chemistry check. Its job is to reduce uncertainty quickly.

That is more important in tech than many teams admit. SmartRecruiters' Technology Benchmark Recruiting Metrics 2025 (https://www.smartrecruiters.com/resources/article/technology-benchmark-recruiting-metrics/) puts median time-to-hire in tech at 48 days, 26% slower than the cross-industry median. Much of that drag shows up later, when teams are still trying to answer questions that should have surfaced earlier.

So I use a six-part scorecard on the first call:

- **Motivation**: why this person is open, and why this role matters now
- **Fit**: whether the role matches the kind of environment and work they want
- **Logistics**: whether the process can work in practical terms
- **Communication**: whether they can explain their work clearly enough for the process ahead
- **Level**: whether their actual scope matches the seniority they claim
- **Technical reality**: whether the work sounds real, specific, and proportionate to the role

These six areas are enough to make a sound next-step decision. Not a final decision. Just a useful one.

> ✓ **What you are doing on the first call**
>
> You are not trying to fully validate the candidate. You are trying to remove the biggest unknowns quickly enough to make a sound next-step decision.

A structured scorecard helps because it keeps the call evidence-based. That is not just my preference. SmartRecruiters' Technology Benchmark Recruiting Metrics 2025 (https://www.smartrecruiters.com/resources/article/technology-benchmark-recruiting-metrics/) notes that scorecard use is already standard practice across technology teams.

It also helps because the market is noisy from both sides. LinkedIn's Talent 2026 research (https://news.linkedin.com/2026/LinkedIn-Research-Talent-2026) says 66% of recruiters report it has become harder to find qualified talent, 42% feel pressure to fill roles faster, and 39% feel pressure to uncover hidden-gem candidates. At the same time, Greenhouse's 2025 AI in Hiring report summary (https://www.greenhouse.com/newsroom/an-ai-trust-crisis-70-of-hiring-managers-trust-ai-to-make-faster-and-better-hiring-decisions-only-8-of-job-seekers-call-it-fair) found that 91% of recruiters had spotted candidate deception, and 74% of hiring managers were more concerned about fake credentials or misrepresented experience than a year earlier.

That does not mean you should become suspicious of everyone. It means you should stop confusing a pleasant conversation with a useful screen.

Preparing a scorecard that helps instead of getting in the way

A good scorecard should make you sharper, not stiffer. The point is not to sound scripted. The point is to decide, before the call, what evidence you actually need.

I start with the same six areas every time: motivation, fit, logistics, communication, level, and technical reality. Then I define what each one means for this role. That step matters. A generic form copied across backend engineers, frontend engineers, machine learning engineers, and Site Reliability Engineers (SREs), the engineers focused on keeping production systems reliable, is mostly decorative.

A useful scorecard has only three parts under each area:

1. a role-specific definition
2. two or three prompts you can ask naturally
3. the evidence you want in your notes

That is enough. More than that and you start interviewing your spreadsheet.

For example, “senior” does not mean the same thing in every role. A senior frontend engineer may need strong judgment on browser architecture, performance, and collaboration with design. A

senior machine learning engineer may need stronger signals around model deployment, data quality, and evaluation. Same label, different evidence.

When I prepare, I translate the job description and hiring-manager brief into plain language. If a manager says, “We need someone hands-on but strategic,” I rewrite that into observable signal: contributes directly, makes tradeoffs, influences others, and can explain decisions clearly. Otherwise the scorecard fills up with nice-sounding phrases that help nobody.

Build the scorecard before the call

AREA	DEFINE FOR THIS ROLE	CAPTURE IN NOTES
Motivation	Why this job now	Specific drivers, not generic interest
Fit	Environment match	Team pace, scope, stakeholder style
Logistics	Can the process work	Location, notice, compensation, visa
Communication	Clarity and substance	Direct answers, examples, listening
Level	Seniority signals here	Scope, ownership, complexity
Technical reality	What they actually did	Recent work, decisions, tradeoffs

Under each area, I prepare only a few prompts:

- **Motivation:** Why are you open now? What would make a move worth it? Which part of this role fits your next step?
- **Level:** What have you owned end to end? Which decisions were yours? What made the work complex?
- **Technical reality:** Tell me about a recent project in practical terms. What was difficult? What tradeoff did you make? If you used artificial intelligence (AI) tools, what did you verify yourself?

That last question matters because AI is now normal in developer workflows. HackerRank's 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says 97% of developers use AI, and Stack Overflow's 2025 Developer Survey (https://survey.stackoverflow.co/2025) reports that 46% actively distrust its accuracy, versus 33% who trust it. So I do not treat AI use itself as signal. I treat judgment about AI use as signal.

Your notes should capture evidence, not impressions. "Strong communicator" is an impression. "Explained a production incident clearly, named own decision, described tradeoff" is evidence.

A useful first-call scorecard checklist

- ✓ Define each of the six areas for this specific role
- ✓ Write two or three prompts per area
- ✓ Decide what evidence counts before the call starts
- ✓ Use short note fields that force specifics
- ✓ Leave room for follow-up, not just boxes to tick
- ✓ Update the template after the first few calibrated calls

One practical note from experience: the first version of the scorecard is rarely the best one. After a few calls, one question will produce fluff and another will get straight to signal. Keep the second. Delete the first without getting sentimental about it.

If you use AI for preparation or note capture, keep it in a supporting role. SmartRecruiters' Technology Benchmark Recruiting Metrics 2025 (https://www.smartrecruiters.com/resources/article/technology-benchmark-recruiting-metrics/) reports that tech organizations using AI hire 26% faster, or 11 days faster, than those not using it. Useful. But Greenhouse's 2025 AI in Hiring report summary (https://www.greenhouse.com/newsroom/an-ai-trust-crisis-70-of-hiring-managers-trust-ai-to-make-faster-and-better-hiring-decisions-only-8-of-j

ob-seekers-call-it-fair) found that only 8% of candidates believe AI makes hiring more fair, while 87% say transparency matters. So save time with tools, but keep the judgment human and the process clear.

Reading motivation, fit, and logistics

I treat motivation, fit, and logistics as one part of the call because that is how they show up in real conversation. Candidates rarely answer them in neat boxes. You hear why they are open, what kind of work they want, and whether the move is practical all at once.

Start with **why now**. I prefer present-tense questions: What prompted you to take this call? What would have to be true for a move to make sense this year? What feels missing in your current role?

This is worth doing properly. HackerRank's 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says 40% of developers plan to leave their current company within a year, largely because of pay, growth, and meaningful work. Those are normal reasons. What you need to know is which one is real for this person.

A useful answer usually has three parts: a reason, a tradeoff, and a target. "I like my team, but I have become the maintenance person for one product and I want broader system design work" tells you much more than "just exploring options." Generic interest is not automatically disqualifying, but it is weak signal. Weak signal rarely gets stronger just because you schedule more interviews.

Then move to **fit**. Not “would this person enjoy our employer brand presentation.” Actual fit. What kind of team do they do well in? How do they like decisions made? What kind of problems do they want more of? What do they want less of?

If they say they want “ownership,” ask what that means. For one person it means shaping architecture. For another it means being left alone. Those are not the same hire.

Be especially careful with fashionable language. Candidates may say they want an “AI-heavy role,” but that can mean very different things: building model-backed features, integrating application programming interfaces (APIs), improving internal tools, labeling data, or simply not wanting to sound behind. Stack Overflow’s 2025 Developer Survey (https://survey.stackoverflow.co/2025) notes that developers are actively exploring AI topics such as large language models and retrieval-augmented generation (RAG), a way of pulling in external information before a model answers, rather than committing to one magic tool. So ask what day-to-day work they actually want.

Then clear **logistics** early. Compensation, location, work authorization, notice period, time-zone overlap, and process timing are not admin details. They are part of viability.

Logistics to clear on the first call

- ✓ Target compensation and any hard floor
- ✓ Current location and relocation reality
- ✓ Work authorization or visa constraints
- ✓ Notice period and real availability
- ✓ Time-zone overlap for the team
- ✓ Interview timing and competing processes

On location and work model, be direct. HackerRank's 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says 79% of developers prefer hybrid or remote work. If the role requires regular onsite time and the candidate wants fully remote, that is not a sourcing puzzle. It is a mismatch.

Process timing matters too. If the candidate is already in final rounds elsewhere, or cannot start for months, that changes what you tell the hiring manager today. In a market where LinkedIn's Talent 2026 research (https://news.linkedin.com/2026/LinkedIn-Research-Talent-2026) says recruiters feel pressure to move faster, that early clarity is useful.

✓ **What you are really deciding**

Is this person meaningfully motivated for this role, aligned with the actual work, and able to complete the process on realistic terms? If not, stop early.

One more thing belongs here: authenticity. Greenhouse's 2025 AI in Hiring report summary (https://www.greenhouse.com/newsroom/an-ai-trust-crisis-70-of-hiring-managers-trust-ai-to-make-faster-and-better-hiring-decisions-only-8-of-job-seekers-call-it-fair) found that 36% of U.S. job seekers had used AI to alter their appearance, voice, or background during video interviews. I would not turn the call into a detective novel, but I do notice whether answers get clearer or foggier when I ask about recent decisions and constraints. Real experience usually becomes more concrete under follow-up. Borrowed polish usually does not.

Judging communication and level from ordinary answers

This is where many recruiters get vague advice and then are expected to improvise. “Assess communication.” “Check seniority.” Fine. Based on what?

I keep communication and level separate on the scorecard, even though they often show up in the same answer.

Communication is not charm. It is whether the candidate can make their experience understandable. On a first call, I listen for four things:

- **Clarity**: the answer has shape
- **Relevance**: they answer the question asked
- **Ownership**: they distinguish team output from their own contribution
- **Translation**: they can explain technical work in plain language when needed

A candidate does not need to sound slick. Some of the strongest engineers I have interviewed answered slowly because they were trying to be accurate. I would take that over a polished monologue that never actually lands.

Weak communication usually sounds like vagueness, not awkwardness. “We optimized performance.” “I worked across the stack.” “I was involved in an AI initiative.” All of that is empty until the candidate makes it concrete. So follow up: What was the problem? What did you own? What decision did you make? What was difficult?

That matters more now because polished phrasing is cheap. LinkedIn’s Talent 2026 research (https://news.linkedin.com/2026/LinkedIn-Research-Talent-2026) says 66% of recruiters plan to increase AI use for pre-screening interviews. That means your value is less about spotting impressive language and more about checking whether the language maps to real experience.

Level is different. A candidate can communicate well and still be mid-level. Another can be a bit rough in delivery and still be genuinely senior.

For level, I listen for:

- **Scope**: how large the responsibility really was
- **Complexity**: whether the work involved real constraints
- **Decisions**: whether they chose an approach or mainly executed one
- **Tradeoffs**: whether they can explain why one option beat another
- **Context**: whether they understand the impact beyond their own task list

The distinction I care about most is simple: **participated in** versus **drove**.

Signals of participation vs. genuine ownership

IF THEY PARTICIPATED	IF THEY DROVE IT	WHAT TO ASK NEXT
"I worked on the migration."	"I owned the migration plan for our service."	Which part was yours?
Describes team activity	Describes personal decisions	What did you decide?
Mentions tasks	Mentions tradeoffs and risks	What made it hard?
Uses "we" for everything	Can switch between "we" and "I"	What did you personally do?
Focuses on tools used	Focuses on outcomes and constraints	Why that approach?

I am not trying to trap people into saying “I” more often. Some strong candidates are naturally team-oriented. I just need evidence of what was actually theirs.

This is also where AI changes the listening. HackerRank's 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says 67% of developers feel AI has increased pressure to deliver faster, and Stack Overflow's 2025 Developer Survey (https://survey.stackoverflow.co/2025) shows most developers are not blindly trusting AI output. So if someone describes AI-assisted work, listen for verification and judgment. "I used AI to draft tests, but I had to correct edge cases" tells you more than "we use AI a lot."

> ⚠ **Do not confuse polished with strong**
>
> A polished candidate may simply be well prepared. A quieter candidate may be thinking carefully. Score what is evidenced: clarity, ownership, decisions, tradeoffs, and scope.

A simple habit helps enormously: write notes as proof, not impressions. Not "good communicator." Write "explained outage clearly, named root cause, described fix in plain language." Not "senior." Write "owned rollout strategy, discussed risk tradeoff, coordinated across teams." If you cannot write the evidence, you probably do not have it.

Technical reality checks without pretending to be the hiring manager

This is the part that makes newer recruiters nervous. You need to validate technical credibility, but you are not the hiring manager and should not try to be.

Good. Do not.

A technical reality check is not a coding interview, a trivia quiz, or a chance to perform stack knowledge. It is a way to confirm that the profile broadly matches the role, the responsibilities sound proportionate, and the candidate can talk concretely about work they actually did.

I keep this part very grounded. On the first call, I ask about five things:

- recent work
- system or product context
- team setup
- decisions
- ownership

My first-call technical reality check

- ✓ What have you worked on most recently?
- ✓ What kind of system or product was it?
- ✓ What was your part of the work?
- ✓ Who made the key technical decisions?
- ✓ What tradeoff or constraint mattered most?

You are listening for three things: **specificity, coherence, and honest limits**.

Specificity means the answer contains real detail. “I worked on payment systems in Python” is thin. “I owned backend services in Python that handled retries after payment gateway failures” is better.

Coherence means the pieces fit together. If a candidate describes themselves as senior, led architecture, improved reliability, and mentored others, the follow-up should still sound like one real job, not a pile of borrowed headlines.

Honest limits matter as much as confidence. Strong candidates often say things like, “I did not choose the database, but I owned the migration,” or “I was not the tech lead, but I drove this part.” I

trust that much more than somebody who appears to have personally invented every system they ever touched.

A few questions do most of the work:

- What was the team building, in plain English?
- What part did you personally own?
- What changed because of your work?
- What was difficult about it?
- If I asked your manager what you were strongest at, what would they say?

None of these require you to judge whether the technical solution was brilliant. They help you judge whether the candidate has a real relationship to the work.

Be careful with tool-name theater. You do not need to recognize every framework, cloud service, or acronym on earth. If a candidate mentions a term you do not know, ask them to explain it in context and in plain English. That is not a sign of weakness. It is the job.

This matters especially now because AI-assisted work is normal. HackerRank's 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says nearly a third of code is now AI-generated, and 76% of developers say AI makes gaming assessments easier. So if a candidate says, "I built this with AI," ask the useful question: what did you still need to verify, test, or debug yourself?

For AI-specific roles, the bar changes a bit. A machine learning engineer should usually be able to talk concretely about data quality, evaluation, failure modes, and how success was measured. For non-AI roles, do not let fashion distort the screen. A backend engineer does not become weak because they are not performing AI enthusiasm on command.

Role context should shape the reality check:

Adjust the reality check by role

ROLE	ASK ABOUT	LISTEN FOR
Backend engineer	Services, data, APIs	Ownership, scale, tradeoffs
Frontend engineer	User interface complexity, performance	Concrete user impact
SRE	Incidents, monitoring, automation	Operational judgment
AI or ML engineer	Data, evaluation, production	Measurement and limits
Software Development Engineer in Test (SDET)	Test automation, defects, coverage	Quality judgment and collaboration

A few recruiter-friendly examples help here:

- **Backend engineer**: ask about services, databases, APIs, performance, and reliability
- **Frontend engineer**: ask about browser performance, state management, user interface complexity, and collaboration with design
- **SRE**: ask about uptime, alerts, incidents, and automation
- **SDET**: ask what they automated, where defects were caught, and how they worked with developers

That is enough for a first call. You are not certifying technical depth. You are confirming reality.

Conclusion

If I had to reduce this book to one sentence, it would be this: the first call should earn its place in the process.

Not by being long. Not by being charming. By producing a real decision.

When you screen the same six areas every time, with role-specific prompts and evidence-based notes, a lot of noise falls away. You spot weak motivation earlier. You catch obvious mismatch before the hiring team spends an hour on it. You hear the difference between polished language and real ownership. And you move the promising candidates faster, which matters in a market where SmartRecruiters' Technology Benchmark Recruiting Metrics 2025 (https://www.smartrecruiters.com/resources/article/technology-benchmark-recruiting-metrics/) shows tech hiring still moves too slowly.

That, in my experience, is the real value of the first call. It is not glamorous, but neither is cleaning up a messy funnel three weeks later.

If you are reading through the rest of this series, this book pairs naturally with the ones on recruiter calibration, hiring-manager alignment, and technical role basics. They solve adjacent problems. This one is about the moment where signal first has to beat noise.

Keep the call structured. Keep it human. Keep your notes concrete. That alone will make you better than a surprising amount of hiring process design.

Glossary

- **AI (Artificial Intelligence)** — Software systems that perform tasks such as generating text, summarizing information, or helping write code.
- **API (Application Programming Interface)** — A defined way for different software systems to communicate with each other.
- **Backend engineer** — An engineer who works on server-side systems such as business logic, databases, and integrations.
- **Browser architecture** — How a web application is structured and runs in the browser, including performance and maintainability.
- **Data quality** — How accurate, complete, and reliable data is for building or running software systems.
- **Frontend engineer** — An engineer who builds the user-facing parts of software, usually in a web or mobile interface.
- **Large language model** — An AI model trained on large amounts of text that can generate and analyze language.
- **Machine learning engineer** — An engineer who builds and deploys systems that use models trained on data.
- **Model deployment** — Putting a machine learning model into a production environment where real users or systems can use it.
- **Production incident** — A problem in a live system that affects users, performance, or availability.

- **RAG (Retrieval-Augmented Generation)** — A method where an AI system pulls in relevant external information before generating an answer.
- **SDET (Software Development Engineer in Test)** — An engineer who builds automated tests and quality systems to catch issues before software is released.
- **SRE (Site Reliability Engineer)** — An engineer focused on keeping production systems reliable, available, and well-monitored.
- **State management** — How a frontend application tracks and updates data as users interact with it.
- **Tradeoff** — A choice between competing options, where improving one thing usually costs something else such as speed, simplicity, or flexibility.
- **Work authorization** — Legal permission for a candidate to work in a specific country.
- **Time-zone overlap** — The working hours a candidate shares with the rest of the team across locations.

About This Book

About the Author

Natalia Romanova

Natalia is the founder of Talantrix, an applicant tracking system for tech recruiters. She began her career as a software engineer, later moved into engineering management, and eventually led several software companies, including ones she founded. Along the way she repeatedly had to build technical recruiting capability, and over time ended up learning a lot about hiring engineers—more than she ever expected to.

About Talantrix

Talantrix is an intelligent applicant tracking system built by recruiters, for recruiters. Designed specifically for technical hiring, it helps recruitment teams automate the busywork — from CV parsing and skill-based search to pipeline management — so they can focus on what matters most: connecting with top engineers and making great hires.

Learn more at https://talantrix.com (https://talantrix.com)

The Tech Recruitment 101 Series

A practical book series for recruiters who work with engineering teams. Each volume tackles a specific challenge in technical hiring — from screening candidates to understanding code — with clear explanations, real-world examples, and actionable frameworks.

Tech Recruitment 101

A practical book series for recruiters who work with engineering teams. Each volume tackles a specific challenge in technical hiring — from screening candidates to understanding code — with clear explanations, real-world examples, and actionable frameworks.