TECH RECRUITMENT 101

How Tech Hiring Is Different

A practical field guide to what makes engineering hiring its own kind of work

PART OF THE *TECH RECRUITMENT 101* SERIES

How Tech Hiring Is Different

A practical field guide to what makes engineering hiring its own kind of work

CONTENTS

How Tech Hiring Is Different

I wrote this book for recruiters who already know how to recruit and have now discovered that technical roles refuse to behave neatly.

On paper, tech hiring looks familiar. You still run intake, source, screen, coordinate interviews, and close. What changes is the signal quality. The role is harder to read from the outside, titles are less reliable, interview quality matters more, and market conditions shift faster by specialty and seniority.

I have been on both sides of this table for years, and that changes how I see the process. I know why recruiters get stuck, but I also know why engineers lose patience. Usually the problem is not that tech hiring is mystical. It is that too many decisions are made on fuzzy inputs.

This book is part of a broader series on hiring, but this one focuses on the five differences that matter most in technical recruiting: ambiguity, skill depth, specialization, interviewer dependence, and market dynamics. If you understand those five, you will make better judgments, ask better questions, and run a process that works for engineers instead of accidentally fighting them.

Why tech hiring feels harder, even when the process looks familiar

Tech hiring is not a different profession. It is the same recruiting process operating on harder-to-read signals.

That sounds modest, but it changes almost everything. In engineering, title matching is weaker, role definitions are often less settled, adjacent backgrounds are harder to map, and interview quality has more power to distort the outcome. A process that feels competent in general hiring can fail quickly here.

I see the same pattern repeatedly. Recruiters carry over habits that work reasonably well elsewhere: broad outreach, generic intake notes, loose interview loops, and some optimism that the team will sort it out later. In technical hiring, those habits break faster. Engineers tend to read vagueness as lack of understanding, and inconsistency as poor judgment. Usually they are not wrong.

Where tech hiring differs in practice

DIMENSION	GENERAL HIRING	TECH HIRING
Ambiguity	Role goals are often clearer early	Requirements often hide unresolved technical decisions
Skill depth	Surface signals go further	Real ability is harder to infer without context
Specialization	Adjacent titles map more cleanly	Similar titles may mean very different work
Interviewer dependence	Some inconsistency is tolerable	Weak interview design distorts decisions quickly
Market dynamics	Broad pools are more stable	Supply and expectations shift by stack and seniority

A few data points help explain why this feels difficult on both sides. HackerRank's 2025 Developer Skills Report (https://pages.hackerrank.com/hubfs/PDFs/HackerRank%202025%20Developer%20Skills%20Report.pdf) says 78% of tech leaders struggle to find candidates, while 74% of developers struggle to land jobs. That usually means the bottleneck is not raw supply alone. It is matching, calibration, and process design. LinkedIn's Future of Recruiting 2025 (https://busines

s.linkedin.com/content/dam/lem/business/en/hire/resources/future-of-recruiting/future-of-recruiting-2025.pdf) also reports that 93% of talent acquisition professionals see accurate skill assessment as crucial to quality of hire. In tech, that is the whole game.

> ✓ **The shift that matters**
>
> Do not treat tech hiring as general hiring with extra jargon. Treat it as the same recruiting process operating on harder-to-read signals.

The rest of the book breaks down the five places where that difference shows up most clearly.

Ambiguity starts before the search does

One of the biggest surprises in technical hiring is how often the role is fuzzy before recruiting even begins. The title sounds settled. The actual work is not.

Two companies can both ask for a software engineer and mean completely different jobs. One needs someone to stabilize a fragile backend. Another needs a product-minded full-stack engineer, meaning an engineer who works across both user-facing and server-side parts of a product. A third really needs a technical lead who still writes code. Same title, different search.

That ambiguity usually starts upstream. The hiring manager has part of the picture. The team speaks in shorthand. The job description tries to sound tidy. By the time the req reaches recruiting, the rough edges have been sanded off and the role looks clearer than it is.

This is why title matching is so weak in tech. A backend engineer might be working on application programming interfaces (APIs), internal tools, distributed systems, or what is effectively data infrastructure. A machine learning engineer might be building models, putting models into production, or wiring a third-party model into an existing product. If you recruit from title alone, you are mostly guessing.

The fix is not more sourcing. It is better intake.

Beyond the usual basics, you need to clarify the shape of the work. What problems will this person solve first? What part of the system will they own? What would make the hire feel successful in six months? What can they learn after joining, and what must they already know on day one? If those answers are vague, the brief is vague too.

In my experience, two intake questions do a lot of useful damage:

- What would make you say in six months that this hire was a great decision?
- If I find someone strong in only two parts of this role, which two matter most?

Those questions force the conversation away from buzzwords and toward outcomes. They also expose hidden pickiness, which almost every team has. Sometimes it is legitimate. Sometimes it is anxiety wearing business casual.

Skills-based framing helps here. LinkedIn's 2025 Skills Signal Report (https://economicgraph.linkedin.com/content/dam/me/business/en-us/talent-solutions/resources/pdfs/the-skills-signal-report-2025.pdf) found that workers matched by skills rather than titles qualify for more than three times as many roles, and companies using skills-based searches are 12% more likely to recruit high-quality hires. That tracks with what I have seen. In tech, the title is often the least precise part of the profile.

AI-related roles make ambiguity worse because the labels are being used very loosely. LinkedIn's U.S. AI Labor Market Update (https://economicgraph.linkedin.com/content/dam/me/economicgraph/en-us/PDF/ai-labor-market-update-header-sept-2025.pdf) says hiring of AI engineering talent grew more than 25% year over year in 2025, while AI engineering postings reached nearly 7% of all technical job postings even though AI talent remains a very small share of members. So when a team says they need an AI engineer, you need to ask whether they mean deep model work, production machine learning, or product integration through vendor tools and APIs. Those are different markets.

Just as important, not every engineering role that mentions AI needs deep AI specialization. The same LinkedIn update (https://economicgraph.linkedin.com/content/dam/me/economicgraph/en-us/PDF/ai-labor-market-update-header-sept-2025.pdf) reports that postings requiring AI literacy grew 71% year over year, and software engineer is the top title among roles requiring it. Quite often the real need is not model building. It is an engineer who can work sensibly in AI-assisted workflows.

Intake questions that reduce ambiguity

- ✓ What problems will this person solve first?
- ✓ What skills are truly required on day one?
- ✓ What can be learned after joining?
- ✓ What does success look like in six months?
- ✓ What level of ownership is expected?
- ✓ Where is the team unusually picky?
- ✓ Which adjacent backgrounds would still work?
- ✓ If AI is mentioned, is this model work, production ML, or product integration?

If you get these answers early, everything downstream gets cleaner. Outreach improves. Screens become more consistent. Interviewers evaluate the same job instead of their private version of it. In technical hiring, clarity is not a nice extra. It is the foundation.

Skill depth changes how you read signals

The second difference is skill depth. In many functions, title progression, employer brand, and a tidy career story can take you fairly far. In engineering, those signals help, but they break often.

Two candidates can share the same title, stack, and years of experience and still differ sharply in depth. One built systems and made tradeoffs. Another implemented tickets in a narrow lane. From the outside, both can look strong.

That is why surface proxies are weaker in technical hiring. A long tool list is not evidence. A famous employer is not evidence. Even a polished resume is not much evidence on its own. They are clues, not conclusions.

When I review technical profiles, I look for context and ownership. What problems did they personally solve? What constraints mattered? Did they improve reliability, performance, cost, delivery speed, or developer workflow? A short resume with real substance usually tells me more than a dense catalog of every acronym they have brushed past.

In screening calls, I listen for a few things:

- how clearly they describe the problem
- whether they can explain tradeoffs
- whether they understand constraints
- how they debug
- how they work with other functions

You do not need to interview like an engineer to screen well. You just need to hear the difference between “I used Kubernetes” and “here is why we made that design choice, what went wrong in production, and what we changed.” Kubernetes, for non-engineers, is a system for running and managing software across many servers. Plenty of people can say the word. Fewer can explain the work.

This is another place where skills-based hiring beats title matching. LinkedIn’s 2025 Skills Signal Report (https://economicgraph.linkedin.com/content/dam/me/business/en-us/talent-solutions/resources/pdfs/the-skills-signal-report-2025.pdf) found that workers matched by skills rather than titles qualify for more than three times as many roles. That matters in tech because titles are inconsistent and depth is unevenly distributed.

AI has made some signals noisier still. HackerRank’s 2025 Developer Skills Report (https://www.hackerrank.com/reports/developer-skills-report-2025) says 97% of developers use AI assistants, and Stack Overflow’s 2025 Developer Survey (https://survey.stackoverflow.co/202

5) reports that OpenAI GPT models were used for development work in the past year by 81.4% of respondents. So “uses AI” is no longer a differentiator. The useful question is how they use it.

Strong engineers are often pragmatic and skeptical at the same time. Stack Overflow’s 2025 survey (https://survey.stackoverflow.co/2025) found more developers distrust AI accuracy than trust it, and many are frustrated by answers that are almost right. That is a very engineer-shaped complaint. Listen for verification habits, debugging discipline, and judgment, not just enthusiasm.

> ⚠ **Common false signals**
>
> Do not over-weight brand-name employers, huge skill lists, or polished self-presentation. None of them reliably tell you how deeply someone understands the work your team needs done.

A better screening question is usually narrower and more evidence-seeking. Instead of “Tell me about your backend experience,” ask, “What was the most complex service you worked on, and what made it complex?” Instead of “Have you worked cross-functionally?” ask, “Tell me about a decision that involved product or infrastructure tradeoffs.” You are not quizzing for trivia. You are making room for signal.

Specialization narrows the pool faster than teams expect

Tech hiring gets harder because the market is not one market. It is a collection of smaller ones.

A hiring manager may say, “we need a backend engineer,” as if that were a single category. In practice, that might mean distributed systems, cloud infrastructure, payments, developer platform work, internal tools, or data-heavy application backends. The title stays broad. The actual pool does not.

This is where recruiters can save a search early. Every extra must-have sounds harmless when discussed one at a time. In combination, they often carve the pool down dramatically.

I find it useful to separate requirements into four layers:

1. **Fundamentals**: core engineering ability, problem solving, debugging, and system design.
2. **Domain experience**: the kind of problems they have solved before, such as mobile apps, security, or data pipelines.
3. **Stack familiarity**: specific languages, frameworks, cloud platforms, or tools.
4. **Nice-to-haves**: preferences that make the team comfortable but are not necessary for success.

Many intake meetings blur all four together. Then everything becomes a must-have, which is how teams end up looking for a purple squirrel and acting surprised when it asks for more money.

AI has made this fragmentation sharper. LinkedIn's AI Labor Market Update (https://economicgraph.linkedin.com/content/dam/me/economicgraph/en-us/PDF/ai-labor-market-update-header-sept-2025.pdf) found that U.S. hiring of AI engineering talent grew more than 25% year over year in 2025, while AI engineering postings approached 7% of technical job postings even though AI talent made up less than 1% of LinkedIn members. If you casually add large language model work, retrieval systems, and strong backend fundamentals into one role, you may not be making the search slightly narrower. You may be moving it into a different labor market.

LinkedIn's 2025 Skills Signal Report (https://economicgraph.linkedin.com/content/dam/me/business/en-us/talent-solutions/resources/pdfs/the-skills-signal-report-2025.pdf) also says companies can expand their AI talent pipeline 8.2x globally by focusing on skills over degrees or job titles. That is not just a strategy slogan. In technical recruiting, it is often the difference between a real market and an imaginary one.

How requirement scope changes the search

ROLE DESIGN	EXAMPLE BRIEF	LIKELY SEARCH DIFFICULTY
Broad	Backend engineer; strong fundamentals; any modern cloud	Largest pool; most adjacencies
Adjacent	Backend engineer; data-heavy systems; AWS or GCP; Kafka helpful	Manageable pool; targeted sourcing
Narrow	Backend engineer; AWS; Kubernetes; Kafka; healthcare data; LLM evaluation	Small pool; longer search or tradeoffs needed

When a search gets tight, you usually have three levers: wait longer, pay more, or relax something. There is no secret fourth lever hidden in the applicant tracking system. That conversation is easier if you translate requirements into tradeoffs instead of just saying the manager wants too much.

> ☝️ **A better intake question**
>
> Ask which requirements predict ramp time, and which ones simply reduce interviewer anxiety. That clears a surprising amount of fog.

Your job is not to pretend the market is broader than it is. It is to help the team design a role that matches the talent market they actually want to hire from.

Interviewer quality is part of the evaluation instrument

In technical hiring, interviewer quality matters more than many recruiting teams are told.

In some functions, a weak interviewer is inconvenient. In engineering hiring, a weak interviewer can distort the result. The same candidate can look excellent in one round and mediocre in another because the process is not measuring consistently.

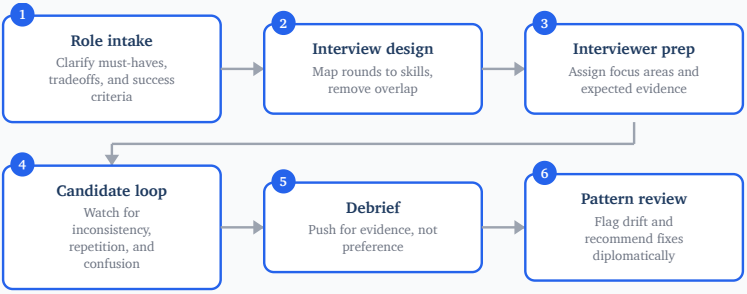
This usually happens for three reasons. First, interviewers do not share a clear definition of what good looks like. Second, they assess through personal preference instead of role requirements. Third, technical interviews are very easy to make performative. A difficult question bank can look rigorous while measuring very little.

Candidates feel this acutely. HackerRank's 2025 Developer Skills Report (https://pages.hackerrank.com/hubfs/PDFs/HackerRank%202025%20Developer%20Skills%20Report.pdf) found that 78% of developers say hiring assessments do not align with real-world tasks, and 56% say algorithm-heavy questions are irrelevant to their jobs. Algorithm-heavy questions, for non-engineers, are the puzzle-style coding exercises many teams still overuse because they are easy to administer and easy to mistake for quality.

AI is another reason interview design needs updating. HackerRank's 2025 Developer Skills Report (https://pages.hackerrank.com/hubfs/PDFs/HackerRank%202025%20Developer%20Skills%20Report.pdf) argues that assessments should evolve away from pure coding output and toward how developers think, debug, and build in real environments. It also reports that nearly a third of code in developer workflows is now AI-generated. If that is true, the question is not whether someone can produce code quickly in a vacuum. It is whether they can use tools with judgment, verify what they produce, and make good decisions around it.

This is where recruiters often get blamed for problems they do not control. You can deliver strong candidates into a weak process and still get bad outcomes. So part of the job is noticing variance early and making it discussable.

Where interviewer variance enters, and where you can intervene



The most useful intervention is a cleaner scorecard. LinkedIn's Interview Scorecards Guide (https://business.linkedin.com/content/dam/me/business/en-us/talent-solutions/talent-hub/pdfs/lth-interview-scorecards-guide.pdf) recommends role-specific scorecards and weighting the competencies that matter most. Good. A backend engineer, a machine learning engineer, and a Site Reliability Engineer (SRE), the engineer responsible for keeping production systems reliable in production, should not all be judged with the same generic template.

The second fix is tighter interviewer handoff. Each interviewer should know what they own. If one round is meant to assess system design, another interviewer should not run a second accidental system design round because they like asking those questions. That is how loops become repetitive and contradictory.

The third fix is a better debrief. I am a fan of a simple rule: no strong opinion without evidence. "I did not love the answer" is not evidence. "The candidate explained tradeoffs clearly but missed monitoring concerns" is evidence. Debriefs get shorter and better when people have to say what they actually observed.

> ⚠ **A practical red flag**
>
> If feedback is detailed only when a candidate does badly, and vague when they do well, your process may be rewarding confidence and familiarity more than evidence.

If you do not control the panel formally, you can still help. Track patterns by interviewer and round. Bring examples instead of complaints. Propose small fixes: a rewritten scorecard, one removed round, a short interviewer brief, or a debrief template with evidence fields. Small fixes get adopted far more often than speeches about process maturity.

In tech hiring, interviewer quality is not a side issue. It is part of the measurement tool.

Market dynamics change the search and the close

The final difference is market dynamics. In technical hiring, the market is not just background context. It changes how you source, how fast you move, and how you close.

A lot of engineers are employed, reasonably paid, and selective. They are not evaluating only compensation. They are comparing your role with their current one across scope, manager quality, technical challenge, work setup, and whether your process feels competent.

This is another reason not to treat tech as one market. HackerRank's 2025 Developer Skills Report (https://pages.hackerrank.com/hubfs/PDFs/HackerRank%202025%20Developer%20Skills%20Report.pdf) found that hiring recovery favored senior talent, with year-over-year activity up 22% for lead developers and 19% for senior developers, versus 9% for junior developers, while entry-level hiring was nearly flat at 7%. That means sourcing strategy and candidate expectations change materially by seniority alone.

AI demand is also reshaping ordinary software hiring, not just niche machine learning roles. LinkedIn's U.S. AI Labor Market Update (https://economicgraph.linkedin.com/content/dam/me/economicgraph/en-us/PDF/ai-labor-market-update-header-sept-2025.pdf) says AI engineering hiring grew more than 25% year over year in 2025, and GitHub's Octoverse 2025 (https://github.blog/news-insights/octovers

e/octoverse-a-new-developer-joins-github-every-second-as-ai-leads-typescript-to-1/) describes generative AI as standard in development. It also reports that more than 1.1 million public repositories now use an LLM software development kit (SDK), with 693,867 of those projects created in the prior 12 months. In plain English, AI-related product work is no longer sitting in a neat corner of the market.

That means outreach has to be specific. Generic messages underperform because the candidate's first question is sensible: why this role, on this team, now? If your note cannot answer that, it is another polite stranger asking for thirty minutes.

Speed matters too. In technical hiring, delay is often read as disorganization or weak decision-making. Strong candidates usually have options, even in softer markets. Move slowly and they start inferring things about the team. Some of those things may even be true.

Close strategy starts early. HackerRank's 2025 Developer Skills Report (https://pages.hackerrank.com/hubfs/PDFs/HackerRank%202025%20Developer%20Skills%20Report.pdf) says 79% of developers prefer hybrid or remote work, and 40% plan to leave their current company within a year, while only 27% expect to stay beyond two years. Openness to move does not mean ease of close. It means candidates are willing to look, then highly selective about what they choose.

✓ **What technical candidates read as signals**

Strong signals: precise outreach, a clear problem to solve, calibrated interviewers, fast follow-up, honest leveling, and realistic compensation. Red flags: generic messaging, unexplained delays, trivia-heavy interviews, surprise changes, and managers who cannot explain why the role matters.

One final point: a softer market does not excuse a sloppy process. Teams sometimes add extra rounds, slow down feedback, or get vague on compensation because they assume candidates will tolerate it. Usually that works exactly until it does not.

Conclusion

If I had to reduce this whole book to one sentence, it would be this: tech hiring is harder because the signals are noisier, not because the process is magical.

Once you see that clearly, the work becomes more practical. You stop over-trusting titles. You push harder on intake. You screen for evidence of depth instead of polish. You treat interviewer quality as part of the system, not as background noise. And you read the market by specialty, seniority, and constraint instead of using one broad story for all engineering roles.

That is the thread running through the other books in this series as well. Better hiring is usually not about heroic effort. It is about better definitions, better judgment, and fewer preventable mistakes.

If this book does its job, you will come away with a simpler mental model for technical recruiting and a sharper eye for where searches go wrong. In my experience, that alone makes the work less mysterious, less frustrating, and a lot more effective.

Glossary

- **AI (Artificial Intelligence)** — Software systems that perform tasks such as generating text, analyzing data, or making predictions.
- **AI assistant** — A tool that helps developers write, explain, or debug code using AI.
- **AI literacy** — Practical comfort working with AI-enabled tools and workflows, even without deep AI specialization.
- **API (Application Programming Interface)** — A structured way for different software systems to communicate with each other.
- **Applicant Tracking System** — Software used by recruiting teams to manage jobs, candidates, and hiring workflows.
- **Backend engineer** — An engineer who works on server-side systems, business logic, databases, and system integrations.
- **Cloud platform** — Online infrastructure used to run software, store data, and scale systems, such as AWS or GCP.
- **Data infrastructure** — The systems used to move, store, and process data inside a company.
- **Data pipeline** — A process for collecting, moving, and transforming data from one system to another.
- **Distributed systems** — Software systems that run across multiple machines and need to coordinate reliably.
- **Full-stack engineer** — An engineer who can work on both the frontend and backend parts of a product.

- **GCP (Google Cloud Platform)** — Google's cloud infrastructure platform for hosting and running software.
- **Generative AI** — AI systems that create content such as text, code, or images.
- **GitHub** — A platform where developers store code, collaborate, and manage software projects.
- **GPT models** — A family of AI language models commonly used for writing, coding, and question answering.
- **Kubernetes** — A system for deploying and managing software across multiple servers.
- **Large language model** — An AI model trained on large amounts of text and code that can generate or analyze language.
- **LLM (Large Language Model)** — A large language model used for tasks such as code generation, summarization, and question answering.
- **Machine learning** — A branch of AI where systems learn patterns from data instead of relying only on fixed rules.
- **Machine learning engineer** — An engineer who builds, deploys, or supports machine learning systems in production.
- **Model work** — Building, training, tuning, or evaluating AI or machine learning models.
- **Production** — The live environment where real users depend on software working correctly.
- **Req** — Recruiter shorthand for an open role or job requisition.

- **SRE (Site Reliability Engineer)** — An engineer focused on keeping production systems reliable, available, and well-operated.
- **SDK (Software Development Kit)** — A package of tools and code that helps developers build against a platform or service.
- **Stack** — The combination of languages, frameworks, tools, and infrastructure used to build a product.
- **System design** — The work of planning how a software system should be structured to meet performance, scale, and reliability needs.
- **Technical lead** — An experienced engineer who provides technical direction and often guides other engineers while still contributing directly.
- **Title matching** — Screening candidates mainly by job title rather than by skills, scope, and evidence of actual work.
- **Vendor tool** — A product bought from an outside company rather than built internally.

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.

Revision 2.0 — 18 March 2026

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.