When the Ground Shifts: Leading Engineering Teams Through the Anxiety We All Feel

Engineering leadership guide for navigating skills obsolescence and team anxiety. Learn strategies to build psychological safety and adaptive teams through rapid technological change. For emerging tech leaders.

15 min read · Oct 14, 2025

Reza Rezvani Following

The message arrived during those quiet hours when introspection tends to surface most honestly:

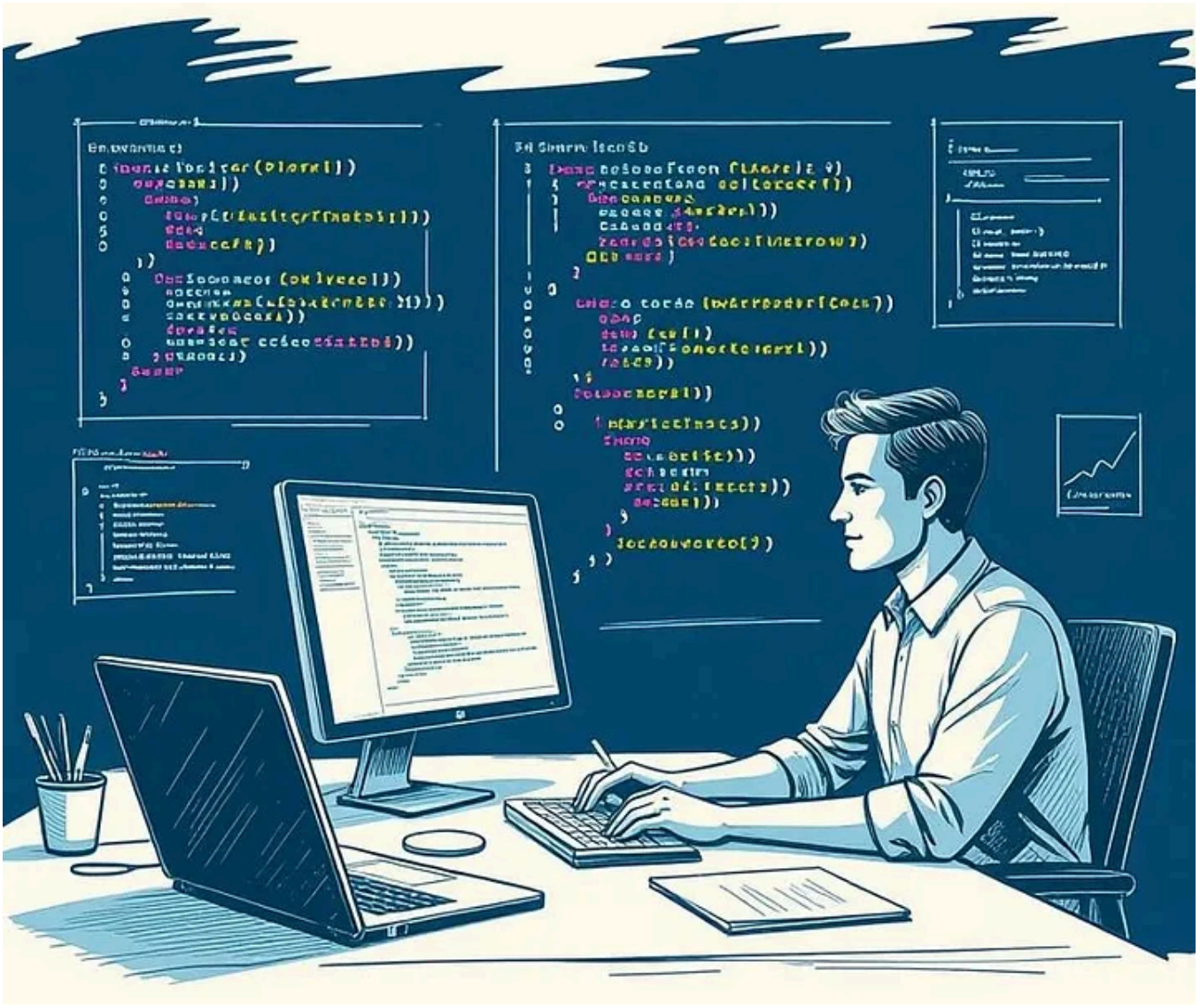
> "Can we talk tomorrow? I need advice."

It came from one of my senior engineers — someone I deeply respect, someone whose technical judgment has shaped our architecture in profound ways. The next morning, over coffee, they shared something I wasn't expecting:

"I watched our new hire solve in hours what would have taken me weeks, using tools I don't even understand yet. And I felt… scared."

We sat there for a moment, the weight of that admission hanging between us. Both of us recognizing this conversation transcended any individual fear — it spoke to something fundamental about engineering leadership in this moment of unprecedented technological acceleration.

What I realized then has shaped my entire approach since: they weren't coming to me for answers. They needed to know they weren't alone in feeling afraid. That the ground shifting beneath their expertise was something their leader felt too.



Building trust in engineering teams through shared learning experiences

The Particular Weight of Engineering Leadership in Times of Transformation

If you're leading an engineering team through this period of rapid change, you're carrying something that transcends typical technical responsibility. You're watching the foundation shift beneath not just your team's capabilities, but your own.

The skills and knowledge that established your credibility — that gave you confidence, that made you valuable — are evolving faster than any of us anticipated.

Here's the dimension of engineering leadership we seldom acknowledge in professional circles: you're expected to project certainty while internally questioning whether what you mastered last year will remain relevant next quarter. You're supposed to guide others through territory you haven't navigated yourself.

I've been in various engineering leadership roles for years now, and I want to share something vulnerable that speaks to the authentic challenge we face: I don't possess

perfect foresight about which skills will matter in three years. I don't wake up confident every morning that my decisions will prove prescient. The roadmap isn't as clear as I wish it were.

What I've come to understand, however, is this: our teams observe us not for flawless answers, but for how we navigate uncertainty itself. For whether we create environments where they can acknowledge their own fears without judgment. For whether we build cultures where people can grow authentically, or places where they must perform infallible expertise.

Industry research from organizations like LinkedIn Learning and professional development firms suggests technical skills now demonstrate relevance periods of approximately two to three years, compared to five or more years a decade ago. But what keeps me awake isn't the technology itself — it's the human beings who are quietly terrified they won't be able to sustain the pace of adaptation required.

The Conversations About Psychological Safety We're Not Having

Last month, during a leadership meeting focused on strategic planning, we discussed *"reskilling initiatives"* and *"capability gaps"* with clinical precision. The conversation remained safely abstract, comfortably strategic. Then someone asked a question that shifted everything:

"Have any of you actually talked to your teams about how they're experiencing all this emotionally?"

The room fell silent — not from lack of caring, but from sudden recognition of what we'd been avoiding.

We invest considerable energy discussing skills matrices and technology frameworks and architectural patterns. We allocate far less attention to acknowledging that change at this velocity, this fundamental in nature, exacts genuine emotional and cognitive costs.

That learning used to energize people, and increasingly it feels like survival. That your team members go home wondering whether they'll remain professionally relevant by next year.

These conversations revealed something deeper that extends beyond tactics or tools. When we talk about *"skills becoming obsolete,"* we're not really discussing code or

frameworks in isolation. We're confronting questions of identity, professional worth, and the fear that if accumulated knowledge no longer carries value, perhaps we ourselves don't either.

I started facilitating different kinds of one-on-ones after that meeting — conversations that didn't lead with sprint velocity or performance metrics.

I simply asked:

"How are you genuinely experiencing all these changes? Not just professionally, but as a person?"

The responses revealed depths we'd been collectively ignoring. People admitted feeling like they were perpetually falling behind despite working harder.

That they were afraid to acknowledge gaps in understanding new tools, particularly in environments where expertise is currency. That the exhaustion from constant learning was eroding the joy that initially drew them to engineering.

One engineer articulated something that has stayed with me:

"I used to love learning new things. Now I'm just tired."

That sentence crystallizes what happens when we prioritize adaptation without equally prioritizing the psychological safety required to sustain it.

What We're Really Confronting Beneath the Technical Surface

I've watched brilliant engineers — people who architected systems serving millions of users, who solved problems that seemed intractable — begin doubting their fundamental value because they haven't mastered the latest AI-assisted development tools.

I've observed junior developers afraid to ask questions because they assume they should already possess certain knowledge. I've seen managers perform understanding they don't actually have because we've created cultures where leaders aren't permitted uncertainty.

This pattern isn't sustainable. More importantly, it doesn't represent authentic leadership.

Real engineering leadership — the kind that transforms organizations rather than simply managing them — begins with acknowledging what's true. And what's true is this: we're all navigating unprecedented territory together. The senior engineers, the junior developers, the CTOs, the emerging tech leads — all of us are learning as we go.

What changes when we simply say that out loud? When we normalize the reality that none of us has encountered this particular moment before?

All these practices — the protected learning time, the transparent sharing, the normalized struggle — they point to something fundamental about building trust in teams. Trust doesn't emerge purely from demonstrated competence. It grows from vulnerability, from leaders who acknowledge uncertainty when it exists, from environments where seeking help is recognized as intellectual courage rather than professional weakness.

Building Teams Capable of Authentic Breathing Room

I think extensively about what differentiates teams that thrive during uncertainty from those that fragment under similar pressures. The distinction isn't about having the most sophisticated technical talent or the largest budgets or access to cutting-edge tools. It centers on trust and psychological safety — the foundation that enables everything else.

When people trust their leaders and each other genuinely, they can admit knowledge gaps without career consequences. They can experiment without fear that failure becomes career-defining. They can collaborate rather than compete. They can learn visibly instead of pretending to already understand.

But trust doesn't originate from competence demonstrations alone. It emerges from vulnerability — from leaders who say "I don't know yet" when that's true, from environments where asking for help signals strength, from teams where learning is celebrated with equal weight to results.

I began doing something simple but initially uncomfortable in my approach to tech leadership development:

I share what I'm actively learning with my team. Not in a pedagogical *"let me teach you"* framework, but in a collaborative *"look, I'm figuring this out alongside you"* spirit.

When I struggle with a new tool or approach, I mention it in team contexts. When I make mistakes, I discuss them openly. When I'm uncertain about technical direction, I say so explicitly.

Initially, this vulnerability felt like undermining my authority — like leaders were supposed to project unwavering confidence. But something unexpected happened: my team started reciprocating that honesty. People began admitting struggles earlier. Asking for help before problems compounded. Sharing nascent learning with each other proactively.

The team didn't become less productive through this cultural shift. They became more resilient, more adaptive, and paradoxically more innovative because they weren't expending cognitive energy on performing certainty they didn't feel.

The Questions That Generate Better Thinking Than Premature Answers

I used to conceptualize my role as providing answers — knowing which technologies warranted investment, which skills required development, which strategic direction we should pursue. I'd spend hours researching, planning, attempting to predict futures that remained stubbornly uncertain.

My understanding of engineering leadership has evolved significantly. My role isn't primarily about having answers. It's about facilitating better collective thinking through more generative questions.

Questions like: *What are we ultimately trying to accomplish, and why does it matter beyond technical elegance? What do our users genuinely need from us that technology should serve? What would we need to learn to serve them more effectively? How can we support each other's growth rather than compete? What would we attempt if we weren't constrained by fear of failure?*

These questions don't yield simple answers or quick resolution. But they create space for authentic dialogue, for collective intelligence that exceeds individual knowledge, for teams that own their direction rather than passively waiting for instructions.

I've noticed something about younger engineers on my team — people earlier in their careers who bring fresh perspectives to tech leadership development. They often perceive possibilities that my experience blinds me to. They're less invested in *"the way we've always approached problems."* They're genuinely curious about new

tools not from obligation, but from authentic interest. They ask *"why not?"* when I'm thinking *"that seems risky."*

What if our fundamental responsibility as leaders isn't protecting teams from change, but creating environments where change becomes an adventure we undertake together rather than a threat we defend against?

The Human Cost of Velocity Without Care

But I've also learned this crucial dimension: inspiration without substantive support is merely pressure disguised as empowerment.

It's one thing to talk enthusiastically about embracing change and continuous learning. It's another to acknowledge that constant adaptation exacts genuine costs. That people have lives, relationships, and responsibilities beyond work. That learning new skills while shipping products while managing everything else represents a legitimately challenging juggling act.

I made a significant mistake once that taught me this lesson. Excited about new AI-assisted development tools, I enthusiastically encouraged my team to *"experiment"* and *"explore."* I believed I was empowering them, creating opportunity for growth. What I failed to recognize was that they heard "figure this out in whatever time you can find or fall behind."

One team member finally told me directly: *"I have young children. I don't have spare time that isn't already allocated. When you talk about all this learning we need to do, I just feel more inadequate."*

That conversation fundamentally changed my approach to building trust in teams. We can't ask people to continuously reinvent themselves without simultaneously asking:

What are we removing from their responsibilities to create capacity for that growth? How are we genuinely supporting them? What are we protecting them from so they have energy for development?

Growth isn't solely about accumulation. Sometimes it requires letting go — deciding what we don't need to do anymore so we have capacity for what matters now.

What Actually Supports Sustainable Adaptation (From Someone Still Learning)

I don't claim to have perfected this. But several practices are working for us — imperfectly, honestly, as we navigate this together:

We protect time with genuine conviction. Not theoretically, but actually. Every week, dedicated hours exist where people can explore, learn, experiment — without tying it to immediate deliverables or justifying its value. Initially, this felt almost irresponsible, like we couldn't *"afford"* it. The truth is we couldn't afford not to create this breathing room. Sustainable growth requires space that isn't consumed by immediate production pressures.

We practice learning out loud as a team norm. We created a communication channel — nothing formal or structured — where people share what they're learning in real time. Small discoveries. *"I just figured out how to…"* or *"I struggled with this and here's what eventually worked…"* It's not performative or curated. It's people genuinely helping each other navigate new territory as they encounter it.

We normalize struggle as part of learning. When someone encounters difficulty with something new, we don't rush to fix it or gloss over it with toxic positivity. We acknowledge the challenge explicitly. *"Yeah, that tool has a genuinely steep learning curve"* or *"I found that aspect confusing initially too."* We work to make struggling a normal dimension of growth rather than something requiring concealment.

We create partnership opportunities organically. Not through formal mentorship programs that can feel forced, but by creating natural opportunities for collaboration. Someone comfortable with a new tool partnering with someone learning it. Not in hierarchical teacher-student dynamics, but as collaborators. The person learning benefits obviously, but teaching deepens understanding for the person sharing knowledge — it's mutually beneficial.

We celebrate different dimensions of contribution. We still celebrate shipping features and solving complex technical problems. But we also explicitly recognize asking for help, admitting mistakes constructively, trying something new even when it doesn't work, and sharing learning that benefits others. We're deliberately building a culture where growth matters as much as immediate results, where psychological safety enables risk-taking that drives innovation.

The Fundamental Shift That Changes Everything About Engineering Leadership

What I believe deeply, based on both experience and observation: people don't inherently resist change. They resist being changed without their involvement, without their voice mattering, without acknowledgment of what change costs them personally.

When we involve people authentically in determining what to learn and why it matters to their work and growth, they're not resistant — they're curious and engaged. When we acknowledge that change is genuinely difficult and provide real support rather than platitudes, they're not overwhelmed — they're willing to try. When we create genuine safety to fail and learn from those failures, they're not afraid — they're creative and innovative.

The teams I've observed thriving aren't those with the most aggressive learning mandates or the newest technologies. They're teams where people genuinely care about each other as human beings. Where senior engineers help junior developers not because organizational charts require it, but because they remember being junior and someone helped them. Where leaders admit uncertainty. Where everyone's genuinely figuring this out together.

I think about that engineer who initially told me they felt scared. We meet for coffee regularly now — it's become a ritual I value. Sometimes we discuss technical challenges. But mostly we talk about how we're both navigating uncertainty in our respective roles. How we're attempting to learn while simultaneously doing our jobs effectively. How we're balancing confidence with humility, conviction with openness.

They told me recently: *"I'm still scared sometimes. But I'm not scared alone anymore. That makes all the difference."*

That sentence captures something essential about tech leadership development in this era. The fear doesn't necessarily disappear. But isolation doesn't have to amplify it.

What If We Reframed the Entire Question?

We've been asking: *"How do we prevent our skills from becoming obsolete?"*

What if we asked instead: *"How do we build engineering teams where people can grow together, support each other through inevitable change, and find genuine meaning in the work even as the tools keep evolving?"*

The technology will continue changing at this accelerated pace. The tools will keep improving and proliferating. The skills we need will keep shifting in sometimes unpredictable ways. That velocity isn't going to slow down, and we can't control that fundamental reality.

But we absolutely can control the kind of environment we create. Whether we build cultures characterized by fear or cultures that enable growth. Whether people face uncertainty isolated or supported. Whether learning feels like falling behind or moving forward together.

An Invitation to Shared Learning, Not Prescription

I'm not prescribing what you should do. I'm sharing what I'm learning, in case it resonates with what you're experiencing in your own engineering leadership journey.

If you're a newer leader feeling the weight of trying to guide others when you're uncertain of the path yourself — you're not alone in that experience. Engineering leadership isn't about possessing all the answers. It's about creating space for discovering answers together, for collective intelligence that exceeds individual knowledge.

If you're watching your team struggle with change and feeling uncertain how to help — start by acknowledging the struggle explicitly. Sometimes people don't need solutions immediately. They need to know their leader sees them clearly, understands the weight they're carrying, and is genuinely in it with them.

If you're trying to balance moving forward with taking care of your people — that tension is real and probably unavoidable. There's no perfect formula. But leaning toward genuinely caring for people as whole human beings rarely turns out to be the wrong choice for building trust in teams.

A Closing Reflection

That senior engineer I mentioned at the beginning? The one who felt scared watching a junior colleague solve problems so quickly?

They're doing something remarkably brave now. They initiated a weekly session where they learn new tools alongside junior engineers. Not as the expert dispensing wisdom to beginners, but as someone learning publicly and visibly. Modeling that uncertainty is acceptable. That we're all perpetually works in progress.

A junior engineer told me it transformed their entire perspective:

> I thought senior engineers just knew everything instinctively. Seeing someone I respect struggle with the same things I struggle with — it made me realize I'm not falling behind. We're all just learning at different paces.

Perhaps that captures what engineering leadership really means during periods of rapid technological change. Not having the map before we start. But being willing to explore the territory together. Not knowing the answer. But being brave enough to sit with difficult questions. Not being fearless. But being honest about the fear and moving forward anyway.

Your team doesn't need you to have everything figured out perfectly. They need you to be authentically human. To care about them as complete people, not just as resources or capacity. To create an environment where they can do their best work not despite uncertainty, but within it — because you've built the psychological safety that enables experimentation and growth.

The technology will figure itself out through continued innovation. It always does.

The question is: *what kind of leader do you want to be while it does? What kind of team do you want to build together? What kind of culture do you want to create?*

Not what some article or framework tells you to create. Not what you should create according to some external standard. But what feels authentically true to you. What aligns with why you chose to lead in the first place.

. . .

Frequently Asked Questions About Leading Through Skills Obsolescence

How do I address team anxiety about AI replacing their jobs? The most effective approach I've found is direct, honest conversation. Acknowledge the fear explicitly rather than dismissing it. Share how AI tools are augmenting rather than replacing human judgment in your context. Create opportunities for people to experiment

with these tools themselves so they move from abstract fear to concrete understanding.

What if I don't understand the new tools my team needs to learn? Learn alongside them. Your vulnerability as a leader who's also learning creates psychological safety for everyone else to admit what they don't know. The goal isn't being the expert on everything — it's creating an environment where collective learning happens.

How much time should we allocate to learning versus shipping features? This balance varies by organization, but teams that allocate 10–20% of capacity to deliberate learning and experimentation typically see compounding returns. The question isn't whether you can afford that time — it's whether you can afford not to invest in your team's adaptive capacity.

How do I convince leadership to invest in learning time? Frame it in terms of risk and competitive advantage. The risk isn't investing in learning — it's watching your team's skills atrophy while competitors build more adaptive organizations. Share concrete examples of how protected learning time has led to innovations or efficiency gains.

What do I do when senior engineers resist new tools? Understand the resistance often comes from fear of becoming irrelevant, not from stubbornness. Have honest conversations about those fears. Show how new tools can amplify rather than replace their expertise. Pair them with engineers enthusiastic about new tools so learning happens through collaboration rather than mandate.

. . .

Let's Continue This Conversation Together

I've shared what I'm learning and thinking about as I navigate engineering leadership through this period of change. Now I genuinely want to hear from you.

What's the hardest part of leading through all this transformation? What's working in your context? What keeps you up at night? What moment made you realize something important about your approach to leadership?

Share your experience in the comments — not because you have it all figured out, but because your perspective matters to others navigating similar challenges.

Because other leaders face similar uncertainties. Because we're all figuring this out together, and collective wisdom emerges from shared experience.

If this resonated with something you're experiencing, I invite you to follow along. I write regularly about the human dimensions of engineering leadership — the aspects we don't always discuss in technical circles. The fear, the uncertainty, the hope, the humanity beneath the surface.

We're all learning. Let's do it together.

. . .

About these reflections: These perspectives emerge from years of engineering leadership across different organizations, extensive conversations with other leaders navigating similar challenges, and ongoing learning about what authentic leadership requires. Every story shared is rooted in real experience, though details have been modified to protect privacy. This isn't about claiming to have answers — it's about asking better questions together and building the kind of technical organizations where people can thrive.

Connect on these topics:

#EngineeringLeadership #PsychologicalSafety #LeadershipDevelopment #TeamBuilding #TechLeadership #EmpatheticLeadership #BuildingTrust #GrowthMindset #HumanLeadership #AdaptiveTeams

. . .

About the author: As CTO of a Berlin-based MedTech startup, I lead a talented team of computer vision and data engineers building next-generation mobile and web applications for healthcare.

My work is driven by a deep fascination with where AI and agentic coding are heading. Over the past decade, I've also explored the intersection of search algorithms, information retrieval, and content strategy — understanding how technology shapes the way we find and interpret knowledge.

I publish what I learn because this technology moves too fast for anyone to figure out alone. Connect with me here on Medium (

Reza Rezvani

), or on Twitter.

You can also connect with me at *alirezarezvani.com* for more insights on AI-powered development, architectural patterns, and the future of software engineering.

Looking forward to connecting and seeing your contributions — check out my *open source projects on GitHub*!

✨ Thanks for reading! If you'd like more practical insights on AI and tech, hit **subscribe** to stay updated.

I'd also love to hear your thoughts — drop a comment with your ideas, questions, or even the kind of topics you'd enjoy seeing here next. Your input really helps shape the direction of this channel.

Leadership Development | Tech Leadership | Engineering Management | Team Building | Empathetic Leadership

Following

Written by Reza Rezvani

1.1K followers · 77 following

As CTO of a Berlin AI MedTech startup, I tackle daily challenges in healthcare tech. With 2 decades in tech, I drive innovations in human motion analysis.

No responses yet

Bgerby

What are your thoughts?

More from Reza Rezvani

In nginity by Reza Rezvani

How Cursor and Claude Code Plugins Turned Me Into a 20x Developer – The Agentic Coding Setup That...

My Background Agent just submitted a PR that made our senior architect ask, “Who wrote this?”

Oct 14

Reza Rezvani

Gemini CLI: What Happened When I Replaced My IDE With a Free AI Terminal Agent for 30 Days

I gave Google's new Gemini CLI full access to my development workflow and tested it on real production code. Here's what actually worked...

Oct 10 •••

Reza Rezvani

"7 Steps" How to Stop Claude Code from Building the Wrong Thing (Part 1): The Foundation of...

Learn how to stop Claude Code from rewriting your architecture with vague prompts. This guide introduces Spec-Driven Development...

Sep 17

Reza Rezvani

The AI Agent That Became Our Team's Silent Partner: A Journey from Chaos to Flow

When everything changed, it wasn't the code – it was how we worked

Oct 11

See all from Reza Rezvani

Recommended from Medium

In Entrepreneurship Handbook by Joe Procopio

You Don’t Need AI

As a new wave of AI hype crashes over the working world, you’ll need this knowledge to stand your ground

4d ago

In Career Paths by Tobias Charles

I Can’t Tell You Exactly How to Be a Better Manager. I Only Know What Worked For Me

There’s no single roadmap to excellence—but there are lessons worth stealing

Nov 2

In Towards AI by Teja Kusireddy

We Spent $47,000 Running AI Agents in Production. Here's What Nobody Tells You About A2A and MCP.

Multi-agent systems are the future. Agent-to-Agent (A2A) communication and Anthropic's Model Context Protocol (MCP) are revolutionary. But...

Oct 16

In Level Up Coding by Jacob Bartlett

The Terrible Technical Architecture of my First Startup

A system that could only be built with the confidence of a 25-year-old Deloitte graduate with an AWS certification

Oct 30 •••

In Predict by Tasmia Sharmin

The Man Who Invented AI Just Admitted What Tech CEOs Won't Say!

Geoffrey Hinton: They're spending $420 billion on AI. It only pays off if they fire you

Nov 2 •••

The Secret Developer

You Can’t Fake Passion When You Can’t Use the Product

Imagine this. Working at your desk is in Hyderabad as part of the “feature team” building the checkout flow for a major US retailer. Great...

4d ago