Hector Chavez

Application Answers — real answers written for real applications

ElevenLabs · https://elevenlabs.io

tell us about a hard problem in your past

I built monsterhouse, a real time pipeline that watches livestreams and automatically detects clip worthy moments from chat, audio, and visual signals. It shipped as a small SaaS. The project was constrained by hardware: eight models packed into a little box with 8GB VRAM. The machine cannot hold every model in VRAM at once, but the interesting discovery was that it did not need to.

I was forced to build a scheduler that keeps every model idle until a signal from the livestream is detected and wakes the correct model to interpret it. I did not design the orchestration layer up front, it emerged out of necessity when quantized or weaker models regressed the output or crippled production time. The same GPU also had to be shared by the rendering pipeline that produces portrait clips from the landscape source.

As a result, I used the limited box as a local reference for a cost-effective cloud VM with the same compute budget to make the product work at scale and remain profitable. One instance of monsterhouse can produce between 600 and 800 clips a day 24/7 and upload the output automatically to TikTok and YouTube. The core pipeline is live and generating clips, but auto upload at scale is bottlenecked by a review and approval process from Google and ByteDance to use their commercial APIs.

What makes you excited about the ElevenLabs mission?

As a user, I align with the goal of making media and content more accessible to audiences worldwide. ElevenLabs helps creators connect with audiences that otherwise would not be able to consume their content because of a language barrier. The same tools also help individuals with medical conditions that degrade or affect their voice to communicate and share their stories.

The technology can revolutionize day-to-day operations for enterprise clients, spanning lighthearted projects like the Taco Bell drive through powered by Omilia, to more serious roles like automated first responders and virtual debt collectors.

Hims & Hers · https://www.hims.com

What has been your typical split between frontend and backend work throughout your career?

60/40 backend-leaning overall. My paid work has been almost entirely backend and infrastructure: I ran the IT function for a construction holding company with 1,000+ endpoints across seven subsidiaries. My independent work is an even split between backend architecture planning and careful frontend design that works on any device.

What languages, libraries, and frameworks are you most proficient with?

Python and FastAPI built monsterhouse, an AI/ML-powered short-form video pipeline for livestreams, and heinrich, a debugging toolkit for LLMs. TypeScript and JavaScript built handlingtheloop.com, a browser DJ engine running on a Cloudflare Worker. Rust built rotten-apple, a Xen hypervisor toolkit that lifts a bare-metal machine into dom0 and manages guest VMs. Infra is usually PostgreSQL and SQLite for storage, GitHub Actions for CI/CD, and Docker as the backbone of testing and deployment in some cases. PowerShell and Bash come from automating IT work. MCP (Model Context Protocol) server development is where most of my recent hours have gone.

Have you used AI tools to improve productivity? If so, share a specific example and the difference it made.

Yes, constantly.

I built Osiris because most AI coding tools become unproductive when you're running more than one session at a time. The default state is a chat log with a search bar. Every new session starts blind, and time is wasted re-deriving decisions and context from previous instances.

Osiris is an event-sourced Postgres ledger that gives a fleet of agents a shared memory graph, and an entity ontology with provenance on every fact.

The concrete difference: my day-to-day development workflow mostly consists of orchestrating concurrent AI coding sessions through Osiris. Context from one agent becomes immediately available to every other agent working the same codebase, instead of disappearing after compaction.

In essence, it is a memory prosthesis for myself and the agents.

RunSybil · https://www.runsybil.com/

What does a great demo look like to you? What separates a good one from a forgettable one?

A great demo meets each stakeholder at their own level instead of trying to impress everyone with the same pitch. When I bought Abnormal Security for Stewart Holdings Group, the COO cared about stopping phishing, account compromise, and spam. I cared about integration with Entra, how it behaved in prod, and the learning curve for my users. Abnormal's demo answered both, respectfully and without wasting our time.

We signed a demo agreement that gave them Microsoft Graph API access, scoped for mail, identity, and behavioral signals across our tenant to train their model live, with me as the

main operator flagging false positives and low-confidence hits. Setup complexity was abstracted behind a consent screen, and it plugged in without disturbing our existing email security product Mimecast. The experience was competent and felt trustworthy from day one, because it let us run a real side-by-side comparison in production without disturbing day-to-day operations.

The forgettable counterexample was a DLP vendor that promised "automatic, frictionless integration." The first thing I actually touched in their demo was manual configuration of data types, essentially Purview with worse UX. The gap between what was promised and the product they delivered killed all my trust in their company and pitch.

A great product only makes half a great demo. The other half is the person running it. Abnormal's FDE was upfront when he didn't know an answer and followed up by email instead of guessing. He was the one consistent point of contact who fanned questions out to the right person internally. His warm engagement and continuity mattered more than encyclopedic knowledge of the product.

How do you handle a technical stakeholder who is skeptical of what you are showing them?

Being the technical stakeholder is a game of ego and power. I can say no, and I'm probably juggling three other demos in the same category, so the question in the back of my head is "what makes this vendor different?" The challenge is not technical, it's political and emotional: humbling me politely while respecting the power dynamic, and earning my trust as the person gatekeeping a blank check.

The hardest I've ever been to convince was during our purchase of Netskope's SD-WAN products. I inherited a legacy stack (RADIUS, Cisco DUO, and Fortinet VPN) that predated me, so I was blind on two sides: half not knowing exactly what was already running, and half not knowing what was about to replace it. My skepticism did not come from expertise, it came from genuine ignorance and paranoia of being upsold into something I didn't fully understand, while also being responsible for translating our actual business needs back to their engineers. There was real embarrassment in asking for clarity and making them repeat explanations. What actually broke the tension was their patience and willingness to go at my pace.

The pattern that worked every time started with a pushback. I would say "We don't need this," and instead of steamrolling me with jargon, the engineer would lay out exactly why we did, in context of our scale and needs, and how saving a few dollars would actually hurt our day-to-day experience with their product once it was live. There was art and finesse in negotiating my legacy assumptions against where their stack needed to go, carefully, without breaking anything in production.

From where I sat, I really could not tell whether the underlying product was better or worse than a competitor's, because I did not have the technical depth to make that judgment. What tipped the scales was the relationship they built with me: an engineer willing to step down from heaven to earth, warmly, and without condescension, dissolved my tension and gave me peace in a way that a feature comparison could never.

What is the hardest part of scoping an engagement accurately and how do you manage it?

The hardest part is separating what the environment actually needs versus what the vendor offers, especially when the product sounds like a great fit in isolation, but overlaps or conflicts with existing deployments.

Our experience with Arctic Wolf is a great demonstration of that conflict. We had real scale and mess: three physical offices, a fleet of job trailers, thousands of endpoints, a mix of Linux and Windows servers, an aging device fleet, and a workforce who did not know or care about our security posture.

Their engineers wanted us to bite on their full vertical stack from year one: We only wanted their core network detection and endpoint telemetry product. They pushed for "Aurora" to replace Carbon Black while we were already transitioning to Defender and add on their email "Security Awareness" product which overlapped with Mimecast.

I had to push back hardest on the EDR piece, because we were already halfway migrated to Defender, and ripping it out in favor of an untested alternative with marginal advantages was a real operational risk that I was not willing to take. The process of replacing a security agent on a Windows machine is agonizing and time consuming, even with a flagship MDM, and their infrastructure was not even rolled out in our tenant yet, so their claims of enhancement and synergy with their core product could not be verified or A/B tested first hand.

The lesson: map what's already running and where a new pitch might conflict with it before I ever bring it to the table, so the customer isn't the one who has to catch it and push back the way I had to, saving us both the embarrassment of buyer's remorse.

What security product or service do you think does the best job selling to enterprise security teams and why?

Abnormal. Selling security to an enterprise client is an uphill battle, and they win at that game by removing the reasons to say no before they ever ask for the sale. Instead of pitching every feature, they let the product prove itself against your own tenant's real mail, identity, and behavioral data, positioned as a complement to whatever is already deployed rather than a replacement for it.

That's the whole strategy: don't ask an enterprise security team to trust a slideshow, let them watch the product work against their own environment while everything they already trust stays untouched.

For a stakeholder protecting a budget and a reputation, the combination of provable results plus a trustworthy human sells harder than any feature list.