



Vintage Marantz receivers from the 1970s are highly prized pieces for audio enthusiasts. We do many refurbishments on these units and customers are thrilled.

Norman's Electronics, Suwanee GA

Geoffrey Cestaro · Chief Bottleneck Eliminator

October 12, 2026 · Originally published on LinkedIn

My dad was an audiophile. For the uninitiated, this means he spent hours in the pursuit of the perfect sound system. First Receivers and records, then CDs and amplifiers, Electrostatic speakers, and finally, tube amps. Perfection means different things to different people, and for my dad, he was looking for realistic reproduction of the human singing voice. #TheAbsoluteSound

I got my love of high-end audio from dad, as well as my analytical approach to problem solving. When I lost my job to DOGE in 2025, I was looking for a challenge and to have more control over my future. Enter Norman's Electronics, a storied Atlanta audio repair shop.

Norman's has been in business for 71 years. Margins in repair shops are thin, so I was looking at the job as an opportunity to bring my technical knowledge to the table to create efficiencies that would help the company remain profitable as audio equipment and electronics move toward disposable status.

One of my first targets for efficiency gains was the CRM.

“AI is very good at doing what you ask. It is not good at knowing what you meant, but it will fake it to save face.”

The repair business is overwhelming.

If you’ve ever picked up an audio magazine in a bookstore or, sadly, flipped through one on Amazon, the array of equipment options is staggering. Each one promises something different: the warm sound of tubes, the sound of solid-state amps that swear they’re as warm as tubes. Vinyl vs digital, baffled speakers vs open baffle. There’s a dizzying array of choices and people spend their lives in search of the perfect sound. #Stereophile

Audio repair businesses mimic the complexity of the equipment we repair. There’s warranty work, paid at lower rates with the promise of bringing repair volume and full-paying jobs, a claim process that’s different for each manufacturer, endless emails of invoices and credit memos—literally hundreds each week.

Customers love their equipment and are anxious for information on the status. They email, text, call. They call when you didn’t reply to a text they sent, and one guy even sent snail mail when he got frustrated trying to get in touch with us. That one stung.

This is the perfect job for a CRM! Managing texts, emails, and phone calls is stuff most CRMs can do, to varying degrees. What I needed was more specific: everything about a customer and a repair living in one place, on the repair ticket and the parent customer record.

AI To The Rescue?

The trap of the audiophile is that once you think you’ve found the perfect combination of equipment, the natural progression is to question that and start looking for something a little better. Maybe speaker cables made of woven strands of silver will improve the sound. (Spoiler: they will #KimberKable.)

It was clear to me that Norman’s legacy CRM was an old database pushed to the “cloud” using Azure, but it was still very DOS 5.0 looking, with 4:3 screens that could either be maximized to take up the whole screen or were too small to see anything. The fix had been dual monitors at every desk, so the CRM could be full screen on one, and other work could be done on the second screen.

I embarked on the journey of looking for a new CRM.

I had worked with Salesforce at Emory University, before the DOGE Incident. As product owner I translated business requirements from customer to developer and

back. While Salesforce had initially been marketed as an end-user customizable CRM, companies had started to use it for everything. As a result, the functionality grew to the point where finding a single checkbox to turn on functionality meant clicking through multiple levels of settings menus. Not really a dis on Salesforce—they were growing organically to meet a need, and the resulting complexity is typical for any fast growth company.

Even with that, I knew Salesforce had a lot of potential to solve my issues, so I took a meeting. I'll cut to the chase: costs were too high for Norman's revenue to support. At the time, we were a 10-person company, and required customizations would have a budget of \$10k on top of monthly per-seat costs. Once I understood the costs, it honestly surprised me that Salesforce was still marketing itself as a small business solution. Again, not a dis—Salesforce is super helpful for large businesses with a big budget and staff to maintain the system. But I was dealing with their small business team, and the disconnect on budget was frustrating.

I also met with industry-specific SaaS partners specializing in appliance repair to see if they could adapt to our audio business. However, during the product demo, the salesperson ran through a generic sales script and entirely ignored my pre-submitted list of specific requirements. To me, a vendor hiding behind a rigid “dog and pony show” instead of addressing core requirements is a major red flag—it suggests the platform lacks native support for our workflow and would force us into costly custom development or workarounds.

A separate vendor initially showed great promise, but we had to back out during onboarding due to severe architectural red flags. They informed us that we would need to duplicate equipment types for every entry point (such as phone and web intake). This requirement strongly implies an un-normalized database structure that portends a maintenance nightmare for our product catalog. Unwilling to build our business on fundamentally flawed database architecture, I made the decision to terminate the relationship.

At this point, my frustration growing, I decided that I might be able to more quickly meet the needs of Norman's on my own. I would start small, with a credit card capture mechanism that would allow us to store a customer credit card on file and charge as the job progresses. So, I started to think back to my days with SQL, Postgres, MySQL, Ruby on Rails, etc. That scared me a bit, because it really wouldn't save any time to build from scratch.

On a literal whim, I searched for AI coding and came across Lovable. There are others, it's just the one I saw, clicked on, did some work, and fell in love (no pun) with the speed with which I stood up a basic CRM to capture customer credit cards using the Square SDK to tokenize card numbers, store the tokens, and charge customers'

deposits, parts orders, then balance. This was the automation I needed and when I tell you it was simple and fast, it was simple and fast. It was also cheap: about \$60 worth of AI credits. I was shocked at first. And then I leaned into it.

“Any sufficiently advanced technology is indistinguishable from magic.”

My AI Team

A good audiophile stereo system is built with multiple parts. There's an input, like a turntable or CD player, sometimes an intermediary, like a DAC that decodes a CD and makes it amplifiable in analog format, sometimes a pre-amp and amp or just an integrated amp, which in turn feeds the speakers. Breaking the musical reproduction chain apart allows each piece to be best at just one thing. #AudiophileSeparates

While I started with just Lovable, I realized that AI might work best with multiple parts, too. I use three AI systems, depending on what I'm trying to accomplish.

Chat GPT was my first AI. We got along great, until it thought it knew me and triggered my avoidant attachment style, so I dumped it. I'm not joking. I'm the type that if I visit a restaurant multiple times and they say, "Hey! The usual today?" I never go back again. Chat GPT was super pushy back then, and it really thought it knew what I wanted and was repeatedly wrong. I eventually went back and now use it for help tweaking my writing and to create illustrations.

Claude is my coding partner. I sometimes take Lovable's response and paste it into Claude to ask, "Before we code, is there anything I or Lovable missed?" Claude has access to my Git repo and my Supabase, so it can look at tables and migrations to cross-check our work. It's a strong pair programming partner.

Lovable has integrations with Supabase, Square, Github, and myriad others that made it super easy to fire up the initial basic customer credit card capture that eventually grew into my CRM.

Infrastructure

Audiophiles obsess over the gear and forget about the wall outlet. But every component in the system runs on the same electricity, and dirty power means a noisy floor under the music. Serious listeners run a dedicated 20-amp line to the listening room, add a power conditioner, and suddenly the black between the notes gets blacker. Nobody brags about their outlets, but get the power wrong and you may be searching for solutions in expensive equipment that will never fix the issue. #DedicatedLine

Phones & Voicemail

We were paying \$392/month for VoIP. I couldn't fathom that cost for a small business, and from a monolith company that markets itself to small businesses. That seemed a little out-of-touch to me. I decided to find a way out of that trap.

“We went from \$392 a month for phones to about \$45.”

I ported my numbers to Twilio and used Lovable to set up a phone system, with a voicemail IVR tree, transcription, call transfer, and the added benefit of requiring callers to press “3” to get through, which stopped spam robocall recordings. We can also answer a voicemail with a text message, and once the AI proves itself worthy, those replies will go out automatically.

If a caller ID matches a number in our database, a link pops up to open the customer ticket for that person, just like modern CRMs.

AT&T's Office at Hand is priced per-seat and phone number, while Twilio, because you must build your own interface, is per-minute. There's no need for multiple phone lines with Twilio—it can handle unlimited connections with a single phone number. Cost is around \$0.0085 per minute. The funny thing is, we're not on the phone for hours a day. We have mostly quick calls and a lot of voicemails to return. The resulting cost using Twilio is about \$45/month instead of \$392. In addition, Office at Hand required VoIP phones at each desk, each with an assigned number. All that is gone—we have the CRM open on our computer and the phone rings there, anywhere in the country.

Texting

After a laborious process of getting text approval from Twilio, which is a nightmare regardless of the phone vendor, we integrated texting into the CRM. On our old CRM this was very expensive and didn't fully integrate. The old system needed a desktop app open to see that a customer replied to a text. Then we had to copy the phone number from the app and paste it into the CRM to find the customer. This was a limitation strictly of our CRM, not the text provider.

With Twilio, built with Lovable, we created a notification system within the CRM that alerts the tech and case manager when a customer replies to a text or inquires about ticket status. Texting went from \$0.03 (3 cents) per message segment (~240 chars) to \$0.0083 (.83 cents).

Once texting lived in the CRM, we started sending customers a text whenever their repair status changes. Phone calls dropped literally the week we turned it on. It was funny—we all looked at one another, like, “What happened to the phones?”

To keep customers from coming to the office on days we’re closed, Lovable created a text auto reply. Customers can text 411 to a number and receive a single text reply with our hours for the week. As a small shop, this allows us flexibility, because with one main office worker and flu season around the corner, we may have to adapt our hours on the fly. The information also pops up on our website.

Email

The manufacturers we do warranty work for are old companies. Yamaha started making reed organs in Japan in 1887. Denon’s roots go back to about 1910. Marantz started in Queens in 1953, and McIntosh started in 1949 and has been building amplifiers in Binghamton, New York, since 1951. Over the decades they’ve been bought, merged, and sold across Japan, Italy, and the US. Denon and Marantz now belong to Samsung. Companies with that much history carry a lot of legacy process, and ours arrives by email: invoices, credit memos, and claim confirmations, one PDF at a time. Email was supposed to be the upgrade from paper. Now it’s mostly noise floor, with the signal buried somewhere underneath.

Lovable connects to AI models, so I used it to build, in the CRM, a way to dig us out of the email deluge. The CRM reads our billing inbox in Gmail and sorts through both the email text and the attachments, looking for ticket numbers, PO numbers (the ticket number plus a suffix), parts costs, invoices, and credits.

When it finds a parts order for a ticket, it attaches the PDF invoice, pulls the costs, and enters them on the parts line where the tech originally ordered the part. This works because we deal with a small number of suppliers whose invoices are consistent. A bigger operation might not get the same results, but for us it’s surprisingly accurate. When it isn’t sure it pulled the right number, it flags the parts line for review. We ran it manually for a while before flipping the switch to automatic. This is AI at its best: not flashy, just practical, and it saves us hours.

The AI also watches for credit memos. On warranty jobs, we’re charged for the parts we order and then credited once we file the repair claim. That means two documents per job: the invoice, which updates the ticket so we can submit the claim, and the credit memo confirming the refund. Matching the two is the first step toward

calculating revenue per job, and it's how we'll finally stop paying for QuickBooks, which doesn't come close to automating any of this for our use case.

Receipts get the same treatment. When a customer emails a receipt with the ticket number in the subject line, the CRM attaches it to the ticket and notifies the tech and case manager. Paper receipts that arrive with a shipment get scanned to email at our printer, tagged with the ticket number and #Receipt or #Damage_Photo, and land on the ticket labeled "Receipt" or "Damage Photo."

Web Site

Lovable uses a standard web stack: React, Supabase with Postgres, and Tailwind for styles. I separated the public web functions from the CRM to minimize exposure to PII. The public part holds the web form where customers submit a request for service, the service catalog where we publish the models and brands we service, and a repair or replace tool.

The website submission form takes data entered by the customer, including SMS opt-in/out, and passes it through to the CRM, where a customer record is created, if not already there, and a ticket is added as a child record of the customer.

Our service catalog is fed from past repairs and updates nightly as we complete jobs. It has three categories: we don't repair it, we might repair it, we do repair it. Data is broken down by brand, model, and equipment type. The system calculated average costs for different repair types and provides those as well. Cost is always a question from customers, so we get right out with it, which in turn reduces phone calls. We're clear that these are averages and the individual repair may be completely different. There is of course risk there, but the risk of not doing it kills productivity.

Our Repair or Replace tool is one of my favorite uses of AI. It answers another common customer question—is repair worth it? The tool lets the customer provide info on their repair: brand, model, are they shipping it? From what zip code? What's the problem? The AI ingests that info and makes a call to our CRM, where repair history lives. The repair history is updated nightly and recalculates average costs for any newly completed repair tickets. This allows the AI on the front end to look for similar problems for a brand and model, pull average repair costs, search the internet for comparables available for purchase used, and provide the customer with a recommendation. This is the same thing we do when people call the office, which happens about eight times a day. Each call is a 20-minute conversation I love having, but it generates 10 voicemails while I'm on the phone. Now AI handles that first conversation, with the same honesty we provide on the phone. If something is not worth repairing, we tell the customer that.

What I Learned Along the Way

“The algorithm owns the decision. AI owns the interpretation.”

AI loses the thread

AI is not good at following long threads. The longer a conversation runs, the more it drifts, forgets what you agreed on, and starts “fixing” things that weren’t broken. When a thread starts to wander, don’t keep iterating. Revert to the last good version, start fresh, and reframe the request.

Iterating with AI is dangerous. The same thing happens logically when you don’t repeat reminders and prompt AI to revisit prior decisions and refer to memory.

Tell it what matters, and make it remember

AI doesn’t know which of your instructions are passing comments and which are rules. Tell it. I use prompts like “This is a fundamental principle and should be saved to memory” and “Check memory before doing anything to be sure you’re following our guiding principles.” Then remind it before any important request.

Separate questions from commands

Ask a coding AI a question and there’s a good chance it will start changing code to answer it. When I only want an answer, I say so: “Don’t make any changes. I’m asking a question and only want an answer.”

Say the obvious

An inexperienced person using AI won’t necessarily get the results I did. I have 20 years in tech and two technology-focused degrees, one with a minor in database technologies. This was not my first rodeo—and this is important. I started my CRM without telling Lovable to follow relational database best practices, including primary and foreign keys. It hadn’t occurred to me that I needed to. But I have a strong tendency toward pattern recognition, so when one of Lovable’s responses seemed suspect, I dug deeper. It had used my list values as keys, so updating a list label was impossible without changing every entry. AI is very good at doing what you ask. It is not good at knowing what you meant, but it will fake it to save face. If something matters, spell it out, even if any professional would take it for granted. And if you don’t know what a professional would take for granted, find someone who does before you build anything you depend on.

Get a second opinion

Before Lovable builds anything significant, I paste its plan into Claude and ask, “Before we code, is there anything I or Lovable missed?” Running one AI’s work past another catches a surprising number of mistakes. Some are in the logic, some are in the copy, and some are human, like forgetting to push my updates to Git.

Give AI a framework

A good, old-fashioned algorithm will guide AI’s decisions. When I first tested our Repair or Replace tool, I described a 1978 Marantz stereo receiver with a dead channel, and it answered like I’d asked about a three-year-old 5.1 surround receiver. It kept mixing up vintage and modern equipment, and once it decided it didn’t have enough information, it dodged the one thing customers wanted: a price range. The answers sounded confident, polite, and helpful while saying almost nothing. It was the conversational version of a speaker with a blown tweeter: all the bass and none of the detail.

The fix was a table and a decision tree the AI has to work through first. Is it vintage? That means 25 years or older, and it’s in the table. Is it a stereo receiver from Marantz, Pioneer, or Sansui? Also in the table. A full recap and cleaning runs \$850 to \$1,250. Once the decisions are made, **THEN** AI organizes the information and presents it to the customer.

AI works best when its flexibility sits on top of structured logic that reflects how your business makes decisions. If two people with the same equipment and the same problem ask the same question, they should get the same answer, and we should be able to explain why. The algorithm owns the decision, like a doctor’s differential diagnosis: rule things in and out, in a set order, before writing the prescription. AI owns the interpretation. Customers don’t write in decision-tree language. They write things like “it pops when I turn it on and smells kind of like burnt metal,” and AI is excellent at turning that into answers the tree can use.

Run it manually before you automate it

We ran the invoice parsing manually for a while, checking its work, before flipping the switch to automatic. Even now, it flags anything it isn’t sure about for a human to review. Trust gets earned.

Build fast, then rebuild on purpose

AI coders like Lovable make it very easy to build and change a CRM organically. I solved whatever problem was in front of me and moved on. That's a fast way to get something useful and a reliable way to get something that won't scale. For Norman's, that's OK. We're not a big shop, the system handles our volume, and the odds of us outgrowing it are slim. But I'd like other small repair shops to be able to use it, so I'm redesigning it as a multi-tenant system, using the organic version as my spec. Version one taught me what a repair business needs. Version two gets designed on purpose.

Think about security up front

I separated the public website from the CRM to limit exposure of customer data. AI won't think about that unless you tell it to, and it's much harder to split later.

“Now AI is eating the software that ate the world.”

Conclusion

Who should worry about AI? Not coders. Big companies offering overpriced solutions are the first ones who will see AI deflate their business models.

Small businesses have traditionally had two choices: buy software that almost fits and pay to customize it, or buy software that almost fits and live with the gap. Most of us pick door number two, because door number one costs more than our margins can carry. So the shop adapts to the software with printouts, Excel, and sticky notes. Printouts get lost, sticky notes fall off, and the next thing you know you have a job that's been in house for 363 days and a pissed off customer.

AI adds a third door. I built a CRM, a phone system, and a website around the way Norman's actually works, for pennies on the dollar. The expertise that matters most here isn't coding. It's knowing your own operation well enough to explain it precisely. Every small-business owner already has that. Until now, they just didn't have anyone affordable to explain it to.

For the record: our old CRM and phone system cost \$836 a month combined. Today we pay about \$45 a month for Twilio and \$50 a month for Lovable, plus a \$60 credit top-up when I'm building new features or our Repair or Replace tool is especially busy. Even counting three top-ups a month, we're saving about two-thirds; in a typical month, it's closer to 80%.

Big companies that have been charging small businesses thousands of dollars should be very scared of AI. They're still paying the last living COBOL programmer to keep their legacy systems talking to a convoluted website. You can practically watch it happen from the front end: the page sends a smoke signal to some ancient backend

oracle, then sits there waiting for a reply. Meanwhile, AI is letting small businesses replace those products with solutions built for their own workflows, at their own scale, for a fraction of the cost.

In 2011, Marc Andreessen wrote that “software is eating the world.” Now AI is eating the software that ate the world. The real disruption isn’t that AI can write code. It’s that custom software is within reach of businesses that could never afford it before, and that makes expensive, one-size-fits-all products a lot less valuable.

askG(consulting) · Technology and AI for small businesses · askg.consulting ·
[linkedin.com/in/gcestaro](https://www.linkedin.com/in/gcestaro)