

Bhavesh Narsai

TECHNICAL CONSULTANT · AI-ASSISTED BUILDER · PROBLEM SOLVER

I spent nine years taking other people's software apart to prove it could be rebuilt. Now I use AI to help me build my own.

My background is IT support, software escrow verification and technical consulting, not a computer science degree and a decade of shipping features. What I already had was the habit of picking up unfamiliar systems, working out how they really fit together, and taking problems through to an answer with the people involved. AI-assisted development turned that into the ability to build working software myself. This page is what that looks like in practice.

contact@bavtechsolutions.com · 07769 028475 · [linkedin.com/in/bhavesh-narsai-944aa9381](https://www.linkedin.com/in/bhavesh-narsai-944aa9381) · bhaveshnarsai.com

Interactive version of this document: bhaveshnarsai.com/ai



Bhavesh Narsai

Bolton, Greater Manchester · remote UK

background	IT support, software escrow verification, technical consulting
ncc_group	9 years, Dec 2014 to Dec 2023
now	BAV Tech Solutions Ltd, independent since 2024
builds_with	Python, TypeScript, JavaScript, SQL
runs_on	Docker on a VPS, Vercel, Cloudflare Workers
ai_tools	Claude & Claude Code, Cline, Codex
shipped	SmartReply, ReviewPing, Local Enquiry Audit, Site Monitor, First Impression Audit

THE STORY

An extension, not a reinvention

AI did not turn me into a different person. It gave the person I already was a way to build things. Each stage below left something behind that I still use every day.

01

2013 TO 2014

IT Support

Antler, Vita Cellular
Foams

GAVE ME

Troubleshooting and systems thinking.
Someone has a problem now, on a system you did not build.

02

2014 TO 2019

Software Verification

NCC Group

GAVE ME

Reading unfamiliar code, testing it, and finding out on client sites what a build really depends on rather than what the documentation claims.

03

2019 TO 2023

Technical Consulting

NCC Group

GAVE ME

Clients, requirements and delivery. Helping turn around a service line and grow its team from two people to five.

04

2024 ONWARDS

Independent Consulting

BAV Tech Solutions

GAVE ME

Ownership. Pricing the work, selling it, supporting it, and solving problems that have to make commercial sense.

05

NOW

AI-Assisted Building

SaaS, automation, tools

GIVES ME

The ability to turn an idea into working, deployed software myself, quickly, and then keep it running.

Things I have built and run

Small products, built for real local businesses, with payments, logins and other people's phone numbers involved.

SmartReply

In production

Missed-call auto-SMS for small businesses. When nobody can answer, the owner gets an alert, the caller gets a text back, and the lead is recorded.

FastAPI

Supabase

Postgres

RLS

Twilio

Stripe

Docker

Nginx

pytest

THE PROBLEM

A tradesperson up a ladder cannot answer the phone, and the person ringing usually tries the next number on the list. A missed call is often a missed job.

ARCHITECTURE

Caller → Twilio number → Nginx, HTTPS → FastAPI webhook → Supabase Postgres → SMS to owner and caller

AI'S ROLE

Built with AI coding assistants throughout. The commit history shows the webhook engine, invites, auth and Stripe billing landing over two days of sprints in February 2026. A hardening pass in May went back over all of it, and that is where most of the fixes below came from.

CHALLENGES

- Twilio can deliver the same event more than once. A database function now acquires a per-caller cooldown atomically, so a caller cannot be texted twice in quick succession.
- A route ordering bug where `/contacts/all` was being matched as a phone number by `/contacts/{phone_number}`.
- The admin basic-auth check was not actually rejecting bad credentials. Admin routes are now gated at Nginx instead.

WHAT I WOULD TAKE FORWARD

The first version worked. The second version was safe. Idempotency, log redaction and tests all arrived in a hardening pass, and they belong in the first one.

THE APPROACH

One FastAPI service reacting to Twilio's call status webhook, with Postgres doing the consistency work. Multi-tenant from the start: each business has its own Twilio subaccount and number, so the number that was called identifies the tenant.

MY ROLE

Setting the constraints: one service, no queues or extra infrastructure, tenant isolation on every query, and a webhook that must never fail loudly. I wrote those down as project rules for the AI assistant, so every change starts from them.

SECURITY AND QUALITY

- Twilio signatures validated per tenant, with that tenant's own token.
- Row level security on every table. Anonymous access denied; signed-in users read only their own business's rows.
- Stripe webhook idempotency, a CSRF origin check on checkout, phone numbers masked in logs.
- A pytest suite covering the webhook's short-circuit paths and helpers.

ReviewPing

• In production, invite only

Next.js

TypeScript

Clerk

Turso

Drizzle

Stripe

Twilio

Upstash

Vercel

SMS review requests for trades and local businesses. Send one from the dashboard, or let a job system trigger it through a private webhook.

THE PROBLEM

Small businesses get reviews when they remember to ask, which is not often. A text straight after the job works, but only if it happens every time.

ARCHITECTURE

Dashboard or webhook → Clerk auth or secret → Rate limit → Opt-out check → Atomic quota reserve → Twilio → Log

AI'S ROLE

Built in short sprints with AI assistance. Auth, settings, the webhook engine and the full Stripe subscription lifecycle came together in the first week of March 2026. Later work was mostly refactoring and review: pulling duplicated send logic out of two route handlers into one tested path.

CHALLENGES

- The quota check had a race: two concurrent sends could both slip under the limit. Replaced with a single conditional **UPDATE** that reserves the slot before sending and releases it if Twilio fails.
- The plan limit was defined in two places with two different values. Consolidated to one source of truth.
- UK alphanumeric sender IDs are one way, so "Reply STOP" cannot work. Branded messages now link to a web opt-out page instead, backed by a platform-wide suppression list checked before every send.

THE APPROACH

Two ways in: a manual send from the dashboard, or a per-business secret webhook URL that tools like Zapier or Make can post to. A Stripe subscription sets the monthly SMS allowance. Serverless on Vercel, so nothing assumes a long-running process.

MY ROLE

Product and compliance decisions. Who can send, to whom, how often, and what happens when someone says stop. The project rules tell the assistant which parts are off limits: auth, signature checks, rate limits and the opt-out path.

SECURITY AND QUALITY

- Clerk middleware on the dashboard, settings and admin routes.
- Rate limiting on the webhook and send API, in case a secret URL leaks.
- Stripe and Twilio signature verification; zod validation on input.
- Critical auth-bypass CVEs in the auth and framework packages patched, and key dependencies pinned, with a written dependency update routine.

WHAT I WOULD TAKE FORWARD

SMS is regulated territory. Opt-outs, sender ID ownership and consent took as much thought as the product itself, and they are not things an assistant will raise unless you ask.

Local Enquiry Audit

• localenquiryaudit.co.uk 

Next.js

TypeScript

Prisma

Postgres

Stripe

Search APIs

Vercel cron

A scored audit of how a local business shows up online, with a free preview and a paid full report that reads the business's own website.

THE PROBLEM

Local businesses lose enquiries without knowing why. No Google listing, too few reviews, no tap-to-call button, a website that never says which town they work in.

ARCHITECTURE

Audit form → Discovery: search and places → Scoring engine → Postgres → Free preview → Stripe checkout → Full report and crawl

THE APPROACH

Find the business online, score what exists, and explain it in plain English. A readiness crawl reads the homepage plus up to four key pages, runs checks over what it actually found, and rolls them up into five questions a business owner would ask.

AI'S ROLE

AI helped build it. It is deliberately not inside it. Every finding is a deterministic check over pages that were actually fetched, and the checks stay quiet when unsure rather than guess. A language model writing the findings would have been quicker to build and harder to trust.

CHALLENGES

- Discovery accuracy: matching the right business among similar names took several rounds of name matching, URL normalisation and validating local listings.
- Discovery moved search providers part way through, and scoring moved onto real Google review data.
- The discovery step costs money per call, so it is rate limited per IP in Postgres.

WHAT I WOULD TAKE FORWARD

If AI goes into the report later, it is for polishing the wording of findings a check has already proved, not for producing the findings.

MY ROLE

The scoring model and the honesty rules. Findability and contact count for most because those decide whether an enquiry happens. The report never implies it tested live rankings or what an AI assistant says about the business.

SECURITY AND QUALITY

- No customer accounts: the full report needs both an unguessable token and a paid status.
- Admin by Google sign-in, restricted to one account, signed httpOnly session cookie.
- Machine endpoints behind separate secrets.
- A daily job deletes audits after 12 months, plus per-record deletion for erasure requests.

Site Monitor

• [websitemonitor.app](#) 

Watches a website for the quiet breakages nobody notices, and emails the owner only when something that was working stops.

Cloudflare Workers

D1

Cron

HTMLRewriter

Resend

Stripe

Zero npm deps

THE PROBLEM

Nobody re-checks a website once it is built. A plugin update strips the structured data, a redesign drops the tap-to-call link, and enquiries leak for months before anyone notices.

ARCHITECTURE

Hourly cron → Due sites from D1 → Guarded fetch → Parse and run rules → Snapshot → Diff → Email on change

AI'S ROLE

Built with AI from a written specification rather than a loose prompt. The spec covers cadence, cost per check, what an alert says and what the product must never do, and the assistant works inside it.

CHALLENGES

- Scheduling without a queue: one hourly cron drains whichever sites are due, with uptime checked more often than full content.
- Deciding what counts as a change worth an email, which is a product question more than a code one.

WHAT I WOULD TAKE FORWARD

A clear spec is the best prompt. The more of the thinking happens before the assistant starts, the less of its output needs undoing.

THE APPROACH

"The product is silence until something breaks." Each check is diffed against the last snapshot and only a change sends an email. The first check is a silent baseline, and a site has to fail three times in a row before it counts as down.

MY ROLE

Product boundaries. It reuses the check engine behind the free website tools instead of forking it, it never crawls, never stores page HTML, and treats alert fatigue as the biggest product risk: any new check defaults to quiet.

SECURITY AND QUALITY

- Fetching arbitrary URLs safely: timeouts, size caps, SSRF guards and re-validation of every redirect.
- Admin behind Cloudflare Access; subscribers use magic links.
- Zero-dependency Node test scripts covering the checks, the diffing and the domain logic.

First Impression Audit

• bavtechsolutions.com 

Cloudflare Workers

Browser Rendering

Puppeteer

Queues

R2

D1

Stripe

THE PROBLEM

Website checkers, including my own free one, read what is written on a page. They cannot tell you that the main button does nothing when tapped, or that the cookie banner covers half the screen on a phone. Only an actual visit finds that.

ARCHITECTURE

Free scan → Stripe checkout → Signed webhook → Queue → Headless browser visit → Screenshots to R2, facts to D1 → Findings → Private report

AI'S ROLE

Built with AI assistance on top of the existing scanner. The commit history shows the audit, its landing page, the print view, refunds and promotion codes landing over two days in August 2026, followed by a pass to make the report consistent and credible.

CHALLENGES

- Keeping the paid audit and the free scan from ever disagreeing. Both now run the same scoring code over the same parse of the page.
- Deciding when a buyer gets their money back. Automatic refund when the audit failed to collect evidence; never because the page turned out to be broken, since finding that is the product.
- Cloudflare's rate limiting is per location and approximate. Documented and proven with a diagnostic rather than assumed.

THE APPROACH

A £15 one-off audit, no account. A headless mobile browser lands on the page, screenshots it before touching anything, deals with the cookie banner, scrolls like a person, taps the main button, records the outcome and opens the enquiry form. The report is the screenshots in the order a visitor meets them, each with a plain English note.

MY ROLE

Separating evidence from judgement. One phase drives the browser and records raw facts, judging nothing. A second phase turns stored facts into findings and can re-run without a browser. Every finding traces back to a measurement, a screenshot or a recorded outcome, and the report never estimates what a fix would earn, because that number would be invented.

SECURITY AND QUALITY

- Stripe webhook signatures verified; jobs deduplicated on the checkout session, so a redelivered event cannot run twice.
- Reports and screenshots served only behind unguessable tokens.
- Hard caps on every run: a 120 second deadline with the browser always closed, plus limits on steps, screenshots and screenshot size.
- Checkout rate limited, and only reachable from a valid free-scan token.

WHAT I WOULD TAKE FORWARD

Measure before paying for infrastructure. Cold browser launches measured under two seconds, so there is no warm pool and no fixed monthly cost, and because jobs only exist behind a payment, spend scales with sales.

ALSO BUILT

Everything, live and in development: [Software & tools](https://bavtechsolutions.com/tools) · [Free tools](https://bavtechsolutions.com/free-tools/)

Landing page conversion scanner

Scores a page out of 100 on how well it turns visitors into enquiries. Quotes the actual headline, counts the actual form fields and names what is missing, with no predicted conversion rates. Runs on the shared Cloudflare Worker check engine that Site Monitor and the paid audit reuse.

[Scan a page](https://bavtechsolutions.com/free-tools/page-score)

Contact checker Chrome extension

Checks whether a customer can actually reach the business from the page you are on. Reads the fully rendered page, compares it with the raw code crawlers see, and measures the phone-screen view. Collects nothing, with the two narrowest permissions Chrome offers.

[About the extension](https://bavtechsolutions.com/free-tools/contact-checker-extension)

OrderFlow

Multi-tenant takeaway ordering: a storefront per shop, collection orders, an owner dashboard and a live kitchen display. Next.js and Firebase.

[See it with the other apps](https://bavtechsolutions.com/tools)

Transcript CLI with an LLM fallback

Python tool that pulls YouTube captions first and only falls back to the Gemini API when there are none, parsing its timestamped output back into text, SRT or VTT.

Agent-ready repositories

Every repo I maintain carries an `AGENTS.md` : stack, commands, layout and hard rules. Any coding agent starts with the same constraints a new engineer would get on day one.

Website build tooling

Templates first, prompts only where no template fits, and a validator that fails a build containing an invented testimonial or an unsupported claim, because a model will happily write both.

AI-first, not AI-only

AI speeds up the middle of the process a great deal. It does not change who is responsible for the start and the end.

01 UNDERSTAND

Understand the problem before the tool

Who has the problem, what it costs them, what they have already tried, and what done looks like. Most of the useful decisions are made here, and none of them need code.

Business problem Users Constraints Desired outcome

FROM THE BUILDS

Site Monitor's spec opens with the problem and the customer, and has one rule that overrides everything else in it: the product must complement the consulting work, never compete with it. Every feature is checked against that before it is built.

02 DESIGN

Design the shape on purpose

How the system is structured, where the data lives, which integrations it depends on, and who can see what. An assistant will happily produce any architecture you ask for, so it matters that you ask for the right one.

Architecture Data model APIs Integrations Security

FROM THE BUILDS

SmartReply is one FastAPI service with Postgres handling concurrency, and no queue. Local Enquiry Audit has no customer accounts at all; a token plus a paid status is enough. Both were choices, not defaults.

03 BUILD WITH AI

Build quickly, inside agreed rules

This is where AI changes the pace. Implementation, scaffolding and prototypes that would have taken me weeks now take days. The assistant works from written project rules, so speed does not come at the cost of the things that must not change.

Claude & Claude Code Cline Codex for a second opinion
Rapid prototyping

FROM THE BUILDS

Each production repo has project rules for the assistant: never weaken auth or signature checks, migrations are additive only, prefer the smallest safe change, never log a full phone number.

04 CHALLENGE THE OUTPUT

Challenge the output

Generated code gets read, not trusted. Less "write this for me", more "tell me what is wrong with this". Security reviews, edge cases, database permissions and how an API behaves when something upstream misbehaves.

Code review Debugging Security Edge cases
Database permissions API behaviour Maintainability

FROM THE BUILDS

ReviewPing's quota check had a race where two concurrent sends could both get under the limit. SmartReply's admin auth looked right and let bad credentials through. Neither shows up in a quick demo. Both needed someone reading the code critically.

Ship it, then keep it running

Shipping is the start of the useful feedback, not the end of the job. Real users find what tests do not, and software that takes payments and sends texts needs looking after.

Deployment Real-world testing Feedback Monitoring

Iteration

FROM THE BUILDS

Health endpoints for uptime monitoring, a runbook, a weekly dependency update routine, and alerts when someone signs up or pays. Deployed on a Docker VPS, Vercel and Cloudflare depending on what each product needed.

CONSULTING + BUILDING

Two halves of the same job

Anyone can prompt a coding tool now. The harder part is knowing what to ask it for, and then explaining what was built to the person paying for it. I am most useful where the two overlap.

Consulting

Understanding ambiguous problems
Client and stakeholder communication
Translating technical concepts
Requirements
Investigation
Managing issues through to resolution
Working across technical and non-technical teams

WHERE I WORK BEST

Discovery that ends in working software
Getting to grips with an unfamiliar system fast
Explaining trade-offs to whoever signs them off
Taking a problem from report to fix
Helping people use AI tools well

Building

AI-assisted development
APIs and webhooks
Databases and permissions
Integrations and automation
Web applications
Docker, Linux and VPS
Debugging, testing, security awareness

One missed call, step by step

This is SmartReply's actual request path, in the order the code runs it. It is a public endpoint that spends money every time it sends a text, so most of it is about deciding when *not* to.

1

Was the call actually missed?
Twilio reports every call status. Only `no-answer` continues.

`ignored_status`

2

Which business owns this number?
The called number identifies the tenant. Unknown numbers stop here.

`no_client`

3

Did this really come from Twilio?
Signature checked with that business's own subaccount token, not a shared one.

`bad_signature`

4

Is the service switched on?
A per-business switch, so it can be paused without a deploy.

`bot_inactive`

5

Under this month's SMS limit?
Usage is capped per business, so a bad day cannot run up a bill.

`sms_limit_reached`

6

Record the lead
Logged for the dashboard. A failure here is logged and does not stop the texts.

`insert lead`

7

Alert the owner
The owner always hears about a missed call, whatever happens next.

`sms owner`

8

Is the caller a saved contact?
Uploaded contacts never get the auto-reply. Your supplier does not need a "sorry we missed you".

`known_contact_suppressed`

9

Outside the cooldown window?
Acquired atomically in a Postgres function, so retries and repeat calls cannot double-text.

`cooldown_blocked`

10

Text the caller back, count the usage
Sent from the business's own number on its own subaccount.

`ok`

Every exit returns HTTP 200 with a reason code. An error response invites a retry, and a retry here can mean a second text to the same person. The reason goes to the logs, with the phone number masked.

COMPONENTS	
Caller	A customer ringing a business. They never see any of this. They just get a text back instead of silence.
Twilio subaccount number	Each business has its own Twilio subaccount and number. The number that was called is how the system knows which business this is, and each subaccount has its own credentials, so one tenant cannot sign requests for another.
Nginx on a VPS	Terminates HTTPS in front of the app, and gates the admin routes with its own authentication so the application does not have to get that right on its own.
FastAPI in Docker	One service doing one job. The webhook handler runs the checks in order, and never returns an error, because an error invites a retry. Deliberately no queue or extra moving parts: a small business tool should be cheap and boring to run.
Supabase Postgres	Holds tenants, leads, contacts, usage and subscriptions. The cooldown is acquired inside a database function so two near-simultaneous events cannot both win. Row level security on every table means a signed-in user can only ever read their own business's data.
Stripe	Subscriptions and checkout. Its webhook is idempotent: each event is recorded once, so a redelivered event cannot apply twice. Checkout also checks the request origin.
Dashboard	Where the owner sees missed calls, usage and status. Contacts who should never get the auto-reply can be loaded from CSV, VCF or plain text. Sessions follow Supabase Auth token lifetimes, with refresh rotation.

AI can generate the implementation in minutes. The harder questions are still there.

01 Should this exist at all?

03 How should the system be structured?

05 What happens when something fails?

07 Can another person understand it?

02 What problem are we actually solving?

04 What data should it be able to touch?

06 Is the generated code secure?

08 How do we know it actually works?

None of that is generated for you. It is judgement, and judgement is built rather than prompted.

I wrote more about this: AI has not replaced learning, it has accelerated it

CAREER FOUNDATION

The building sits on ten years of the other bit

Nine of those years were at NCC Group in software escrow verification: taking client software I had never seen, working out what it really needed to build, and proving it could be rebuilt from source in an independent lab without the original developers.

That meant working through each client's own codebase and build toolchain, visiting development teams in the UK and abroad, and writing it all down so somebody else could repeat it. Before that, first and third line IT support.

10+

years in software and IT

9

years at NCC Group

2→5

build team grown

2024 to now	Independent Technical Consultant	BAV Tech Solutions Ltd
2019 to 2023	Software Build Consultant	NCC Group
2015 to 2019	Verification Consultant	NCC Group
2014 to 2015	Integrity Test Analyst	NCC Group
2014	IT Analyst / Developer, contract	Vita Cellular Foams
2013	IT Support Assistant, contract	Antler

BSc (Hons) Business Information Technology, University of Salford. Azure Administrator Associate (AZ-104), AWS Cloud Practitioner, Google IT Automation with Python, ISTC-accredited technical writing.

Have an interesting problem?

Let's talk.

Email: contact@bavtechsolutions.com

Phone: 07769 028475

LinkedIn: [linkedin.com/in/bhaves-h-narsai-944aa9381](https://www.linkedin.com/in/bhaves-h-narsai-944aa9381)

GitHub: github.com/bnarsai

Full career history: bhaveshnarsai.com/#experience

Interactive version: bhaveshnarsai.com/ai

Software and tools: bavtechsolutions.com/tools