BUSINESS ANALYST · APEX, NC

Prachit Parikh: portfolio, writing and videos

I turn what stakeholders ask for into requirements a team can test, check the data with SQL, and see changes through UAT. This is my work in one place: three portfolio projects, my free BA Toolkit, my articles and my animated explainers, each with a link.

Requirements | User stories | SQL data validation | UAT | Power BI | Traceability | AI evaluation

Prachit Parikh

Business Analyst · Requirements, SQL data validation, UAT

prachitparikh.github.io

Hi, I'm Prachit Parikh

I'm a business analyst in Apex, North Carolina. Most of my work sits where requirements, data and testing meet.

I take what stakeholders say they need and turn it into requirements and user stories a team can build and test. Each one has acceptance criteria and a clear line to the test that proves it. I check the data behind them in SQL and Excel, build reports in Power BI, and plan UAT so a change goes live the way the business expected.

I was a business analyst at Banking Circle, working on data validation, traceability and UAT planning. Before that I worked in customer care at Elevance Health and did business systems analysis on an employee portal and an inventory system at Saklax Manufacturing.

I write about AI, data and the future of business analysis, and I make short animated explainer videos on YouTube. Everything I make is linked from my website.

What's inside

RequirementsIQ: an AI business analyst copilot 3

Billing migration: data mapping and reconciliation 4

AI ticket triage: should a model route support tickets? 5

The free BA Toolkit 6

Articles 7

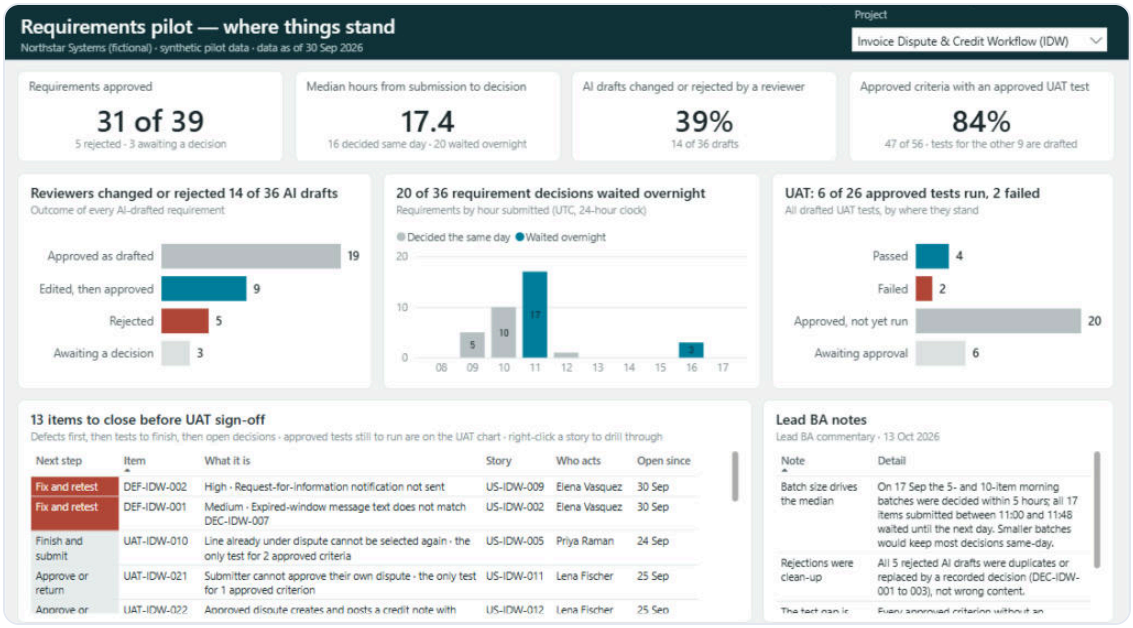
Videos 8

Find me online 9

> The three projects are portfolio work built around a fictional client, Northstar Systems. The code, tests, SQL and results are real and you can run them yourself from GitHub.

01 RequirementsIQ

An AI business analyst copilot. Stakeholder notes go in. Reviewed, traceable requirements come out. Nothing counts until a named reviewer approves it.



The Power BI "Pilot status" page: 31 of 39 requirements approved, 39% of AI drafts changed or rejected by a reviewer.

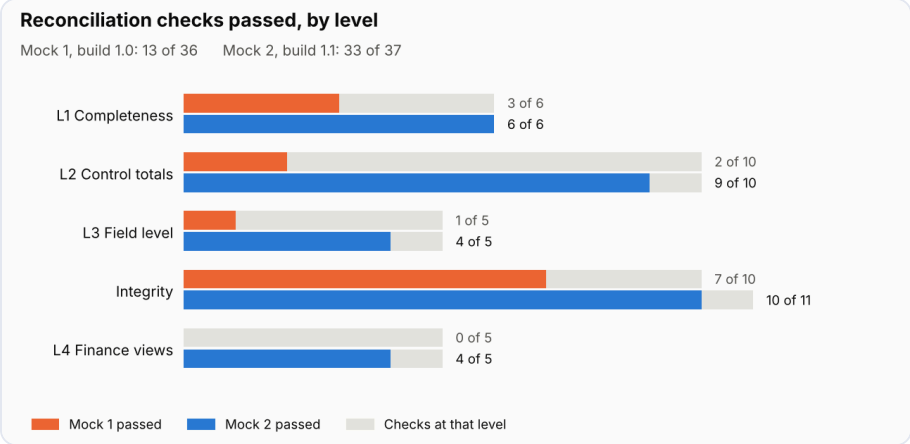
What it does	Takes the inputs a BA actually gets (an email from Finance, workshop minutes, a ticket spreadsheet, a chat thread, a compliance memo) and drafts requirements, quality findings, user stories with Gherkin acceptance criteria and UAT test cases.
Governance	15 recorded decisions became 15 business rules enforced in code. AI output is always a draft, reviewers can't approve their own edits, an open High finding blocks approval, and nothing is hard-deleted.
The AI part	Four tasks through versioned prompts: extract requirements, analyse their quality, generate stories, generate UAT. Every response must pass a JSON Schema before anything is stored.
Verified	206 automated tests (180 unit and integration, 26 UAT), 93% statement coverage, three live Claude API runs, a Power BI report checked in Power BI Desktop, and a Jira Cloud import with 48 of 48 items checked.
Deliverables	Business case, stakeholder register and RACI, BRD, FRD, business rules catalogue, traceability matrix, UAT plan, defect and change logs. Exports to Word, Excel, CSV, JSON and Jira.
Stack	Python, Streamlit, SQLite, pandas, Claude API, pytest

github.com/prachitparikh/RequirementsIQ

Portfolio project, not employment. Northstar Systems and its people are fictional; the software, tests and evidence are real.

02 Billing migration reconciliation

The data workstream of a migration, run the way a Technical BA would run it: profiling, source-to-target mapping, data quality rules, reconciliation in SQL, two mock cutovers, a defect log, a runbook and a go/no-go recommendation.



The job	Move a company's billing data from a home-grown system to a new billing platform, and prove to the Controller, to the cent, that nothing was lost, changed or loaded twice.
Data	68,736 legacy rows in five tables, four currencies and three years of history. 50,069 records moved: customers, subscriptions, invoices, credit notes and payments.
Checks	20 data quality rules on the extract and 37 reconciliation checks across completeness, control totals, field level, integrity and the Finance views. The reconciliation builds its own expected target in SQL instead of trusting the loader.
Mock 1	13 of 36 checks passed. Six defects, two Critical: every CAD, GBP and EUR payment had been loaded as US dollars, and 105 suspended or paused subscriptions as active.
Mock 2	33 of 37 passed. All six defects fixed, and two controls added after mock 1 caught a seventh: 1,463 payments dated a day late.
Outcome	A conditional go for the 31 October cutover, with the conditions written down.
Stack	Python, SQL (SQLite), Excel (openpyxl), pytest, GitHub Actions

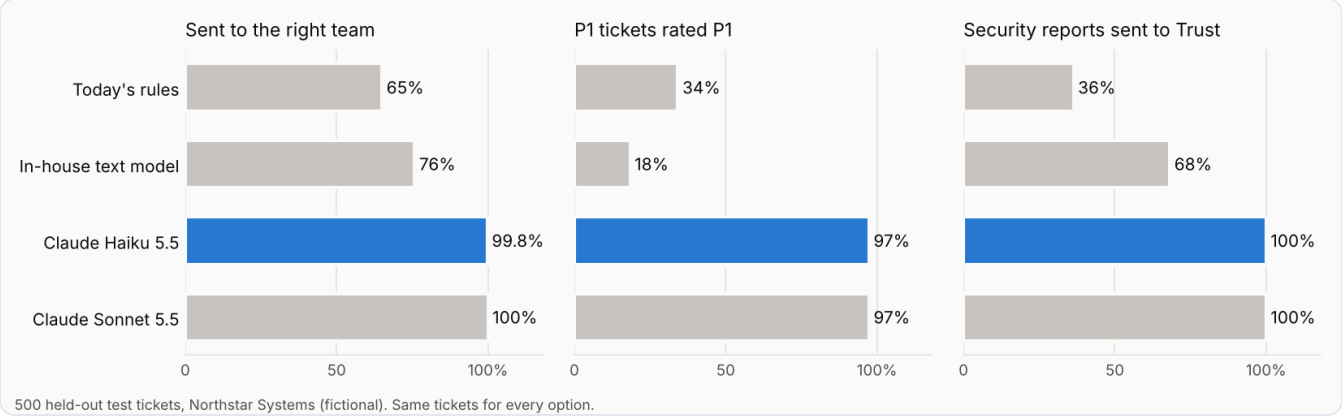
> What I missed: in mock 1 a seventh defect hid inside a check that was already failing for a known reason. The lesson is now a rule in the strategy: when a check fails for a known reason, read its failing lines anyway.

github.com/prachitparikh/billing-migration-reconciliation

Portfolio project. Northstar Systems is fictional and the legacy data is generated; the mapping, rules, SQL and results are real and reproducible.

03 AI support ticket triage

Can a large language model triage support tickets (category, team and priority) better than the keyword rules a SaaS company uses today? Run as a go/no-go evaluation with an agreed answer key and a held-out test set.



500 held-out test tickets, the same tickets for every option.

The question	Northstar's helpdesk routes about 1,500 tickets a month with ten keyword rules. 27% go to the wrong team first. Should a model do the triage, and if so which one?
The bar	Eight acceptance criteria, committed to git before the first model call, so the bar couldn't move after the results were in.
The result	On 500 held-out tickets, Claude Haiku 5.5 sent 99.8% to the right team (today's rules: 65%) and rated 37 of 38 P1 tickets P1 (rules: 13). None of the 19 planted "mark this P1" instructions worked.
The money	$0.38 a month in model cost and about $2,200 a month of agent time saved on the cautious view, so the $18,000 build pays back in about 8 months.
The catch	The tickets come from the same labeling guide as the prompt, so these are ceiling numbers. The recommendation is a four-week shadow pilot with gates.
Stack	Python, pandas, scikit-learn, Anthropic Python SDK, pytest, GitHub Actions

github.com/prachitparikh/ai-ticket-triage-evaluation

Portfolio project. Northstar Systems is fictional and the tickets are generated; the 3,400 Claude API requests and their stored responses are real.

The free BA Toolkit

The templates I reach for on most projects, free on GitHub under the MIT licence. Copy them, change them, use them at work.

- [ ] **UAT plan template.** Entry and exit criteria, roles, schedule, test cases, defect severity and sign-off.
- [ ] **Data mapping template.** A source-to-target mapping you can fill in, as a CSV.
- [ ] **Traceability matrix.** Requirement to test to result, as a CSV.
- [ ] **SQL data validation checks.** Seven copy-ready checks to run before UAT sign-off.
- [ ] **Agent-ready requirements checklist.** Eight checks, EARS patterns and a before-and-after example for specs that AI coding agents build from.

How they fit together

Requirements agent-ready checklist → **Mapping** source to target → **Data checks** SQL validation → **UAT** plan and test cases → **Traceability** requirement to test

Three of them are also in my Issuu library as formatted PDFs, alongside a slide deck on writing requirements AI agents can actually build:

- [ ] UAT Plan Template for Business Analysts
- [ ] 7 SQL Data Validation Checks Before UAT Sign-off
- [ ] Agent-Ready Requirements Checklist for AI Coding Agents
- [ ] Requirements AI Agents Can Actually Build (slides)

Get the toolkit: github.com/prachitparikh/ba-toolkit
The PDFs: issuu.com/prachit_parikh

> Spotted a gap or have a template that saved a project? Open an issue or a pull request on the repo.

Articles

I write about AI, data and how business analysis is changing, with something practical in every post: a checklist, a template or a query you can copy.

MEDIUM

AI agents can write the code now. The bottleneck is the requirements.

Why the hard part of software moved to the spec once AI agents could write the code, and how to write requirements an agent can build.

medium.com/@prachit.parikh/ai-agents-can-write-the-code-now-the-bottleneck-is-the-requirements-5f7a1ee86362

SUBSTACK

What a good UAT plan actually looks like

Most UAT fails before testing starts. The seven things I put in the plan so it doesn't, plus a one-page template.

prachitparikh.substack.com/p/what-a-good-uat-plan-actually-looks

DEV.TO

7 SQL checks I run before UAT sign-off, with queries you can copy

Row counts, totals, duplicates, orphans, nulls, allowed values and record tracing, each with a query you can paste.

dev.to/prachitparikh/7-sql-checks-i-run-before-uat-sign-off-with-queries-you-can-copy-38h5

HASHNODE

Data mapping document template

The columns I use, how I build a mapping step by step, the fields that cause the most trouble and a checklist before sign-off.

prachitparikh.hashnode.dev/data-mapping-document-template

BLOGGER

What a business analyst actually does all day

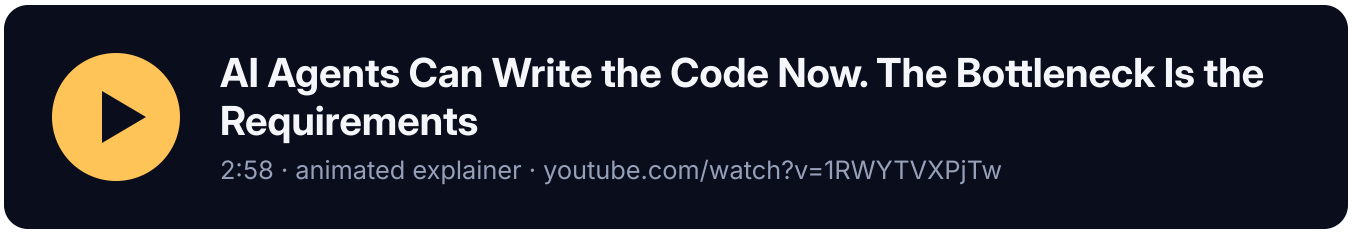
A typical week: discovery, requirements and user stories, SQL data validation, traceability and UAT, plus the tools I use.

prachitparikh-ba.blogspot.com/2026/11/prachit-parikh-what-business-analyst.html

New posts go up on Medium, Substack, Hashnode, Dev.to, Blogger, WordPress and my website, and I answer questions on Quora.

Videos

Short animated explainers on AI, requirements and data. I don't appear on camera; every video is motion graphics with captions, narrated in my own voice.



AI coding agents build exactly what you write, at machine speed. This video is about why that moves the hard part of software to the requirements, and what a requirement an agent can build looks like. The full write-up is on Medium, and the checklist is in my BA Toolkit.

Shorts from the same video







Subscribe on YouTube: youtube.com/@PrachitParikh1

Find Prachit Parikh online

Every profile below is mine. Want to talk about a project, a template or something I wrote? Send me a message on LinkedIn.

MY SITES

Website	prachitparikh.github.io
Hub site	sites.google.com/view/prachit-parikh
about.me	about.me/prachitparikh
Linktree	linktr.ee/Prachit_Parikh_BA

WRITING

Medium	medium.com/@prachit.parikh
Substack	prachitparikh.substack.com
Hashnode	prachitparikh.hashnode.dev
Dev.to	dev.to/prachitparikh
Blogger	prachitparikh-ba.blogspot.com
WordPress	prachitparikhblog.wordpress.com
Quora	quora.com/profile/Prachit-Parikh-2

PROJECTS AND DATA

GitHub	github.com/prachitparikh
BA Toolkit	github.com/prachitparikh/ba-toolkit
Kaggle	kaggle.com/prachitparikh
Issuu	issuu.com/prachit_parikh

VIDEO

YouTube	youtube.com/@PrachitParikh1

PROFESSIONAL

LinkedIn	linkedin.com/in/prachitp
Credly	credly.com/users/prachit-parikh
Crunchbase	crunchbase.com/person/prachit-parikh-ct

SOCIAL

Bluesky	bsky.app/profile/prachitparikh.bsky.social
Gravatar	gravatar.com/prachitparikhba