Mahlab Maniar

Strategy & Operations — building toward AI-native operator roles

Five and a half years across BCG (Doha), Daraz/Alibaba (regional e-commerce, 5 markets) and Trella (YC-backed freight marketplace, Pakistan). No formal ML background.

I am self-teaching AI by taking real operating problems I have lived through and rebuilding them as governed, judgment-aware systems: a deterministic layer for the predictable majority, an AI layer only for genuine exceptions, and a human owner for anything with money or risk attached.

This is not a portfolio of demos. Each piece is a real problem, scoped honestly, with stated limitations.

Targeting: GTM/RevOps S&O · BizOps · Founder's Associate / Chief of Staff · Corporate Strategy — UK & EU
Build tools: Claude Code · Cursor · Make | Anthropic certified: Claude Cowork, Agent Skills

The three pieces

Three problems I have actually lived, spanning three domains and two build architectures — deliberately, so the set proves judgment, not one trick repeated.

01 Carrier Onboarding & Qualification

● Live

Source: Trella

Deterministic vendor checks + agent qualification + routing. Live, no login wall.

carrier-onboarding.vercel.app

02 Deal & Pricing Governance Reviewer

● Live

Source: Trella

Rate-card policy checks + AI triage on exceptions only. ~77% cleared with zero model calls; 24 flagged deals routed by owner.

deal-pricing-reviewer.vercel.app

03 Driver-Based Budgeting Tree

● Live

Source: BCG

Tree decomposition maths + a messy-ledger classification agent. Different domain, different architecture — breaks the Trella pattern.

01

Carrier onboarding & qualification

● **Live**

Source: Trella

Deterministic vendor checks, agent qualification, owner routing. A governed onboarding pipeline, live with no login wall.

Carrier onboarding ran on WhatsApp messages, hand-tiered by whoever read them — inconsistent, non-compliant, and capped at 20 a week

Grounded in a manual carrier onboarding framework I designed and ran at Trella (Pakistan) — synthetic data, real qualification and tiering logic

Inconsistency

No enforced criteria

Two reps tiered the same carrier differently. No scoring framework and no audit trail — a carrier one rep called strong, another deprioritised.

Compliance

Tax exposure was invisible

Carriers were onboarded without a valid 7-digit tax number — the trigger for withholding-tax penalties under Pakistan's FBR rules.

Operations

No view of who matters

No systematic read on which carriers were network-critical or which lanes had supply gaps. Load allocation ran on instinct.

What this tool does: reads a raw submission, enforces the compliance gate deterministically, then scores and tiers the carrier with an agent — producing an auditable decision and a specific action plan, with no manual ops effort on the standard path.

Built with: Next.js · TypeScript · Tailwind · Anthropic Claude SDK · Vercel

The design decision

Not everything here uses AI. A deterministic gate owns compliance; the agent reasons only on the call that genuinely needs judgment.

01 — Workflow · No agent

Compliance gate

Extract fields from the blob, validate the 7-digit tax number and bank details, run five binary qualification checks. Deterministic and auditable — a rejection points to an exact rule, not a model's reasoning.

→

02 — Agent (Claude)

Performance call

Runs only on carriers that clear the gate. Looks up PKR lane benchmarks, reasons over pricing and supply gaps, and returns five scores, a tier, and a rationale a human wants to read.

→

03 — Human owner

Owns the risk

Tier 3 plans and high-severity flags route to an ops lead before action. New carriers run a 90-day monitoring track. The agent never finalises a money-at-risk decision alone.

Why this split, not a model on everything: compliance has to be defensible to an auditor, so it stays deterministic — the agent never has to be right on a rule it can't bend. That leaves the model to do the one thing rules can't: turn five messy signals into a tiering decision with a rationale, on only the carriers that clear the gate.

One carrier, end to end: a near-illegible WhatsApp blob becomes an auditable Tier 1 decision with a reasoned rationale

1 · Raw submission

"Hi, registering Al-Amin Transport. Tax ID 1234567, bank MCB 0012345678. We cover City A–B and B–C. Fleet: 3x 40ft and 2x flatbed. App signup done, first load completed last week. On time 92%, fulfilment 95%."

2 · Extracted [Workflow]

Tax ID **1234567**

Bank **MCB · 0012345678**

Lanes **A–B, B–C**

Trucks **40ft, flatbed (5)**

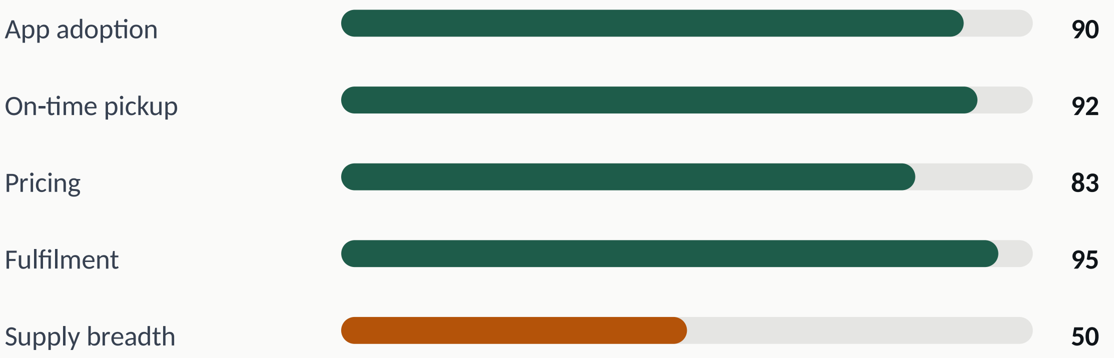
First load **Completed**

App signup **Yes**

3 · Compliance gate [Workflow] PASS

All five criteria met — routed to performance assessment.

4 · Performance assessment [Agent]



Weighted **82 / 100** Tier 1

5 · Decision

A clear Tier 1 carrier — 95% fulfilment and 92% on-time reflect reliable operations. A–B pricing is competitive; the B–C rate sits 17% above benchmark but the fulfilment record justifies Top-tier allocation. Recommend adding them to the A–D corridor — the flatbed fleet suits long-haul.

LOW B–C pricing 17% above benchmark — monitor at next renewal.

One run: 3,459 tokens (extract 612 · agent 2,847)

The system holds across the full range of real submission types — and the cases it can't resolve are the ones it hands off by design

Expected results across 20 synthetic carrier profiles I authored and labelled — reasoned projections by archetype, not a live measured sweep, since the build ran on synthetic data without billed API calls. The individual runs on the previous slide are real.

Case type	Expected result	What it tests
Clean, established carrier	**Tier 1**	Baseline scoring path
Missing / wrong-format tax number	**Rejected at gate**	Deterministic compliance catch
Pricing above benchmark, single-lane	**Tier 3 + action plan**	Agent judgment + plan generation
Urdu transliteration, heavy abbreviations	**Extracted correctly**	Parser resilience
New carrier, no history	**Provisional**	90-day monitoring path

Two limitations, stated plainly

The tax number can't be verified as registered

The gate confirms the number is 7 digits and well-formed. It cannot confirm the number is actually registered with FBR — that needs a tax-authority API integration, a deployment addition, not a prompt fix.

The mixed-carrier blob is untested

WhatsApp forwards sometimes carry two carriers in one message. The extractor assumes one carrier per submission; this edge case hasn't been stress-tested and would need a disambiguation step.

02

Deal & pricing governance reviewer

Live

Source: Trella

Rate-card policy checks with AI triage on exceptions only. ~77% of the book clears with zero model calls; the rest is routed by owner.

Pricing, collections and capacity risk lived in three different inboxes — nobody had one view of which shippers needed action

Grounded in real freight-pricing governance at Trella (Pakistan) — synthetic data, real rate-card and exception logic

- Commercial

No price floor enforcement

Deals got quoted and confirmed below carrier cost. Margin went negative before anyone noticed it.

- Finance & collections

Collections risk was invisible

Shippers with high AR days and high overdue ratios kept getting new deals and extended credit.

- Management

No shared owner or priority

Pricing issues, collections issues and strategic exceptions all landed in the same inbox — no ranked list of what to fix, by whom, by when.

What this tool does: checks every deal and every shipper against policy automatically, and produces one ranked action queue — routed by owner — that commercial, finance and management can all read off the same source.

The design decision

Most AI tools send every case to the model. This one doesn't — rules handle the predictable majority, and the model only sees what genuinely needs judgment.

01 — Deterministic

No model call

Every deal checked against the rate card; every shipper checked against margin, AR-days and overdue thresholds. ~77% clear here — instantly, at zero marginal cost.

→

02 — Agent (Claude)

Exceptions only

Runs only on the ~23% the rules flag. Looks up the rate card, reasons over account context, returns a typed verdict — approve exception, reject, or escalate — with a written rationale.

→

03 — Human owner

Final say on risk

Escalations route to the right owner by breach type — Vertical Lead, Collections Owner, or Commercial Manager. The agent never auto-approves a deal with money at risk.

Why this split, not a model on everything: without the deterministic layer, an AI-only tool would need ~95%+ accuracy to be trusted on commercial pricing decisions. With it, the agent only has to be accurate on the genuinely hard cases — and the escalation path means the system can tolerate being wrong, because a human always has final say.

Output & numbers

77% of the book clears automatically. The other 24 deals get a triaged, owner-routed action queue — not a spreadsheet someone has to interpret

77.1%	**24**	**5**	**12**
In-policy	**Flagged deals**	**Shippers at risk**	**Shippers tracked**
81 / 105 deals	sent to AI triage	one or more red lens	9-month trend each

What each team sees

Portfolio view	All 12 shippers, 3 health lenses (pricing / collection / capacity), sorted by risk.
Shipper drill-down	9-month trend per shipper, full deal table, metric cards against each threshold.
Action queue	P0/P1 items by owner, status cycle, AI rationale visible on every entry.
Weekly report	Month-on-month health deltas by shipper, actions grouped by owner — sent every Monday via Make.

Expected accuracy is highest on clear violations and lowest on genuinely ambiguous calls — which is exactly where the architecture puts a human, not a coincidence

Expected accuracy, not measured — no live API calls were run in this build. Estimated from case structure; running POST /api/review against deals.json would produce real numbers against this same table.

Case type	Expected accuracy	Why
Clear violation	~95%	Rules and AI agree — unambiguous
Strategic + sub-floor	~75%	Volume logic vs. margin discipline is judgment-dependent
Borderline take rate (5–7%)	~65%	Right answer depends on commercial priorities, not just data
Missing rate card cell	~50%	Tool returns null — agent has no anchor and is guessing

Two failure modes, by design

Strategic account over-tolerance

The agent sees "strategic: true" and a rationale for thin pricing, and approves an exception a human CM would reject. The flag is binary — it can't tell genuine strategic accounts from opportunistically flagged ones.

Rate card hallucination

If the agent reasons from memory instead of calling the lookup tool, it anchors on a fabricated number. Mitigation: null tool output auto-escalates rather than producing a verdict.

03

Driver-based budgeting tree

Live

Source: BCG

Tree decomposition maths with a messy-ledger classification agent. A different domain and a different architecture — the set's breadth play.

Finance teams spend two to three weeks on data plumbing before any analysis can start

Grounded in a real BCG cost-decomposition engagement (Qatar) — synthetic data, real tree and benchmark logic

Messy GL exports

1,111 SAP transactions over five years

Ambiguous narrations, inconsistent cost-centre codes, chart-of-accounts drift, and mixed-service lines that cover two cost nodes at once.

Contracts in prose

Rate schedules buried at page 17

A 25-page catering or accommodation PDF holds the tiered rates that drive the forecast. No structured table, nothing a lookup can read.

A budget that won't reconcile

An Excel file that disagrees with the GL

Someone has to find and close the gap between budget and ledger before a single forecast number can be trusted.

What this build does: AI reads every GL row, extracts the rate tables from the contract PDFs, and drafts commentary on flagged nodes. The arithmetic is deterministic. The finance team owns the output before any number reaches management.

The design decision

Most finance AI puts a model in the forecast. This one keeps the model out of the arithmetic entirely — it reads the mess, and deterministic code does the maths

01 — AI consolidation

Reads the mess

Claude reads 1,111 GL rows, the contract PDFs and the budget files, proposes the lookup tables, and maps every row with an audit trail. Runs once, offline.

→

02 — Deterministic recalc

Does the maths

Python rolls the mapped rows into the cost tree and writes the Excel workbook; TypeScript runs the same logic in the browser. Same input, same output. No model call at runtime.

→

03 — Human owner

Owns every number

Every AI mapping sits in reviewable JSON and can be overridden. Commentary is flagged AI-drafted on every surface. Finance signs off before management sees it.

Why this split, not a model on everything: an AI-only tool would have to be trusted on five-year cost arithmetic it should never own. The split puts the model only where reading is genuinely hard, lets deterministic code be auditable by a CFO, and keeps a human on every decision that carries money.

One system, three outputs from the same mapped data — and QAR 8.5M a year in cost flags the model surfaced

What gets produced

Excel workbook

13 tabs, live formulas. Flex one Control tab, the forecast and benchmark recalculate.

Web app

No login. Interactive cost tree, three-scenario forecast, volume sliders, flag commentary.

Exec slides

Auto-generated, one slide per flagged node, actuals and forecast with commentary.

QAR 113.2M 2025 modelled · **1,111** rows mapped

Cost flags — where we pay above the market quote

Node	% ABOVE	QAR M/YR
Camp B › Rent/Lease	+11%	2.4
Camp C › Standard Meals	+14%	1.5
Camp D › Standard Meals	+14%	1.5
Camp B › Staff Meals	+10%	1.4
Camp A › Staff Meals	+10%	0.5
Camp C › Rent/Lease	+8%	0.5
Camp D › Rent/Lease	+8%	0.5
Camp A › Executive Dining	+12%	0.1
Total annual saving		**8.5**

Annual saving = volume × (current rate − market quote). Totals from unrounded figures. Human review required before presenting to management.

The eval is built to measure where the AI is weakest — the messy lines a rules engine cannot place

Expected accuracy, not measured — this build runs on a fallback that copies the ground-truth mapping, so no live model accuracy is claimed. Running the consolidation with a live key scores real precision and recall against this same labelled set.

Case type	Expected	Why
Clean direct match	~98%	GL code, cost centre and vendor all agree — deterministic, no judgement
L1 / L2 — segment and camp	>95%	Strong signals survive even on messy rows
L3 — full leaf path	~85–90%	Depends on narration text where the GL code is ambiguous
Mixed-service line	~60%	One row covering two nodes — genuinely ambiguous, flagged for a human split
CC-3099 unallocated	~55%	No cost-centre signal; the model has only the narration to go on

Two hard cases, by design

Mixed-service lines

"misc camp B — pest + laundry" is one row covering two nodes. The harness scores it as a miss unless split, and the audit trail flags it for review rather than guessing.

Chart-of-accounts drift

GL 612300 becomes 612350 across years. The eval set includes drifted codes, so the score reflects real five-year data rather than one clean year.

Limitations: synthetic data · single-engagement taxonomy · benchmark is peer / budget / RFQ, not external industry data.

The skill on display here isn't building AI. It's the judgment about where it belongs.

A strategy and operations operator becoming AI-native: scoping problems I have lived through, deciding where a workflow ends and an agent begins, and keeping a human on anything that carries money or risk.

- ✓ Three problems I lived and ran by hand — real operating work, not toy demos
- ✓ The same architecture judgment in each: deterministic rules for the predictable majority, an agent only for genuine judgment calls, a human owner on money and risk
- ✓ Honest limitations stated upfront in every piece, before anyone has to ask

All three live, no login wall: carrier-onboarding.vercel.app deal-pricing-reviewer.vercel.app camp-cost-model.vercel.app