PLATFORM ENGINEERING CASE STUDY

LifeStar

A recruiting and outreach engine for life sciences manufacturing — where every rule about who may be contacted, and how, is enforced by the database rather than the interface.

Supabase · Postgres | Cloudflare Workers | 105 Migrations | 328 Locked-Rule Checks

Role → Data model · matching engine · send-gate architecture · delivery

Outreach rules that live in people’s heads get broken

The rules are real, and so are the consequences

Who may be approached, at which address, how often and in whose time zone. Break one and you have burned a candidate or a sending domain.

Every path is another chance to skip the check

A sequence, a one-off reply, a bulk action and an overnight job all end in the same place — a message leaving the building.

A guard in the interface only guards the interface

Rules written in the front end are bypassed by the next script, the next integration and the next person to touch the code.

11 candidate emails delivered to work inboxes

27 managers sent a follow-up with no first email

31 cold emails from a one-to-one mailbox

Two mailboxes, zero replies — from the sends, not the filters

FOUR FAILURES IN ONE WEEK, EVERY ONE A KNOWN RULE

From a job requisition to a contact that is allowed to happen



Requisition in

The role, its discipline, the site and the must-haves.

Ranked, then read

Everyone scored, then checked against the role, person by person.

Gated at send

One function decides whether this message may leave, on every road.

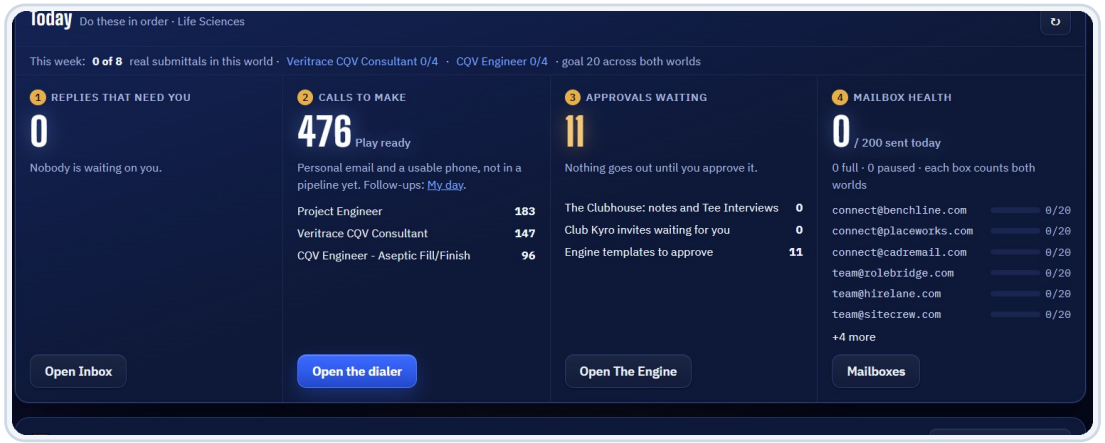
Logged and reversible

Who sent what, to whom, by which road — and how to undo it.



No code path can opt out. The gate runs inside the step that claims a message for sending, so a sequence, a one-off reply, a bulk action and an overnight job all meet the same rules — and a new feature inherits them without being told to.

The actual product, running on real data



Home · the day in four steps, in the order a recruiter works them



A requisition · 335 ranked, 65 a match, 55 reachable today

One number, one definition

Every count on every screen comes from one place, so two screens can never disagree about the same thing.

Nothing goes out until it is approved

The approvals step is on the home screen, not buried. The owner sees what is waiting the moment he signs in.

The screen says what it cannot do

"No manager on file", "no mobile yet" — gaps are shown plainly rather than hidden behind an empty result.

Ranking people by who will be offered the job, and accept it

Two scores, not one

Fit says who matches the role. Will close multiplies the chance of being offered by the chance of saying yes.

Every point opens to its reason

Titles held, years in the role, distance from the site, a contract running out — each one visible behind the number.

Elimination filters start switched off

A new filter only reports who it would remove. An owner turns it on, and a single call undoes every removal it made.

Never from a school, a name or an origin

Scoring reads the work history and the location, and nothing that stands in for where a person is from.



One card · Fit, Offered, Accepts and the meters behind each



Thirty-odd ways to narrow a list; filters hide, they never re-order

One gate, every road

Address class is enforced, not advised

Candidates at personal addresses, clients at work addresses. A message that breaks it is held with a reason, never quietly sent.

Spacing and ordering

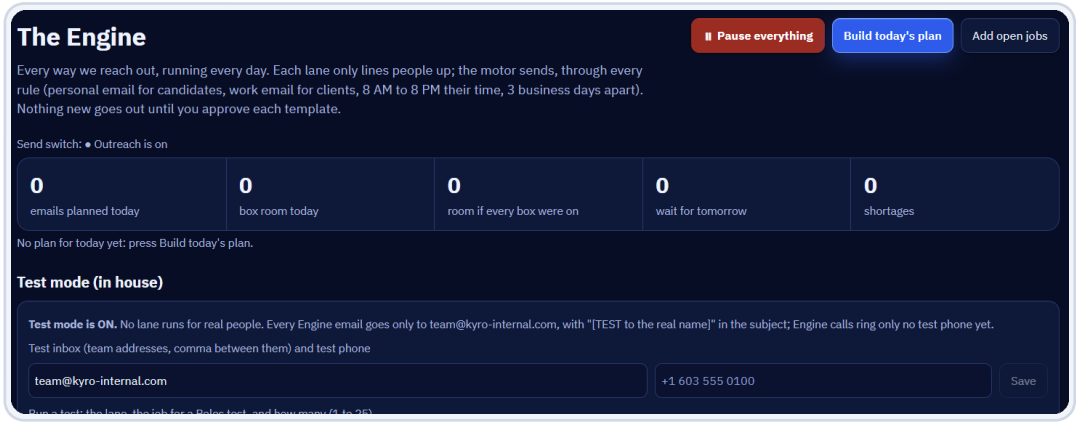
One message per person per three business days, across every mailbox and template — and step two never lands before step one did.

The recipient’s clock, not ours

The sending window closes in the evening where the recipient is, not where the sender happens to be sitting.

Copy guard

Internal job codes, unresolved merge fields and placeholder text are caught before a real person ever sees them.



The rules in the product’s own words, with the switch that pauses everything



Sixty-four mailboxes, each with its own road, owner and daily room

Serverless, with the rules in the one place everything shares

Cloudflare

The app, the public pages and the inbound mail catcher, served at the edge.

Postgres

Tables, row-level security, triggers — and the send gate itself. The rules live here.

Edge functions

27 services: sending on three roads, inbound polling, enrichment and scoring.

Scheduled jobs

Nightly rebuilds, scans and refreshes — each one switchable, none of them able to bypass a rule.

Three sending roads, chosen per mailbox at send time. SMTP, a mail provider's API and a transactional service each carry mail for the boxes they suit. Adding a fourth changes one function — not the thirty places that ask for a message to go out.

Built like production software, not an internal tool

328 locked-rule checks

Every rule the owner has set is written down and tested, so a later change cannot quietly undo one. Nothing ships until they all pass.

Row-level security on every table

Plus an explicit list of the only functions a signed-out visitor may call — re-verified against the live database on every change.

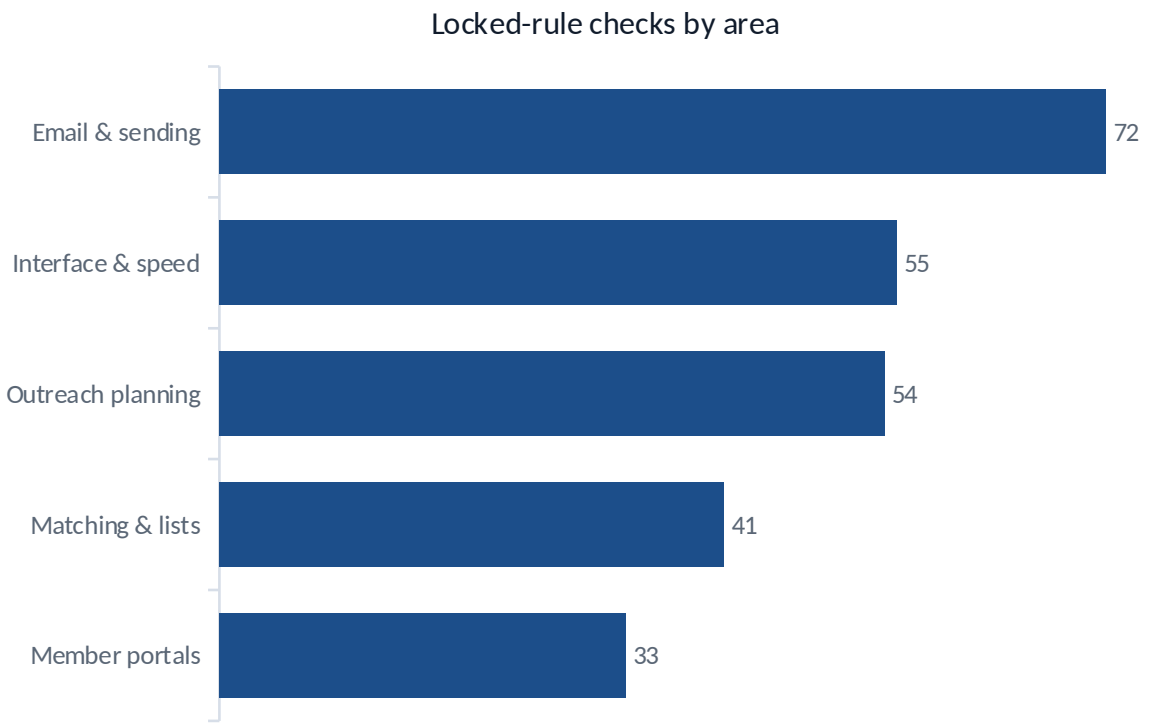
Reversible by design

Every one-time data fix records the old and new value of each row it touched, and every migration ships with its own rollback note.

Secrets no user can read

API keys sit in a vault the server alone can open. The browser may save a key and see that one exists — it can never read one back.

What holds it together, in numbers



The five largest areas of 328 checks in total

105

database migrations, each one shipped with its own rollback note

52

automated test suites, run end to end before anything is pushed

0

send paths that can reach an inbox without passing the gate

The stack



Supabase Postgres

RLS, triggers, scheduled jobs



Cloudflare

Pages, Workers, DNS and mail catcher



Edge Functions

27 Deno services on three mail roads



Vanilla JS

156 modules, no framework, split by screen



Node + Python tests

52 suites gate every push

CASE STUDY

Rules that cannot be quietly undone

Designed and built the data model, the matching engine and the send gate — then wrote the checks that keep every rule in place as the product keeps changing.

Postgres & RLS | Edge Functions | Reversible Migrations | 328 Locked-Rule Checks