Booking and service platform: one failure, how I fixed it, and how I confirmed it

Redacted walkthrough for the Full-Stack Engineer | Search & Data Applications post. No live data, no credentials, no customer or staff names.

1. What the project is

A booking and service-operations web application for a small family aircon business, built with FastAPI and PostgreSQL (14 database migrations). Staff review each request, assign teams and track jobs with an audit history. A background worker delivers events to n8n, which writes reports to Google Sheets. It is a pilot used by the owner with real bookings; customer messaging and payments are switched off. It is not a large-scale production system.

2. The part I built myself, and the part the AI wrote

I work with AI coding tools (Codex in VS Code and Claude Code). I describe the workflow, approve each phase, test the result, and go back and forth until it meets the goal. It is never one prompt. I can explain the code, but I did not hand-write most of it.

Area	What I did	What the AI did
Requirements and rules	Defined the workflow, who approves what, and that PostgreSQL is the source of truth.	Helped write them down as phase plans.
Delivery worker (claim, send, record, retry)	Set the goal: safe error handling and human control. Approved each phase. Tested failure and retry myself.	Proposed and wrote the claim, retry and event-key design.
n8n workflow and Google Sheets	Tested the workflow, spotted the stuck events, applied and republished the fix.	Helped me check the workflow and find the cause.
Deployment and checks	Ran the app on a server, ran the worker on a one-minute timer, read the results.	Wrote the deployment files from my instructions.

3. The failure I resolved: events stuck as pending

- Symptom: automatic deliveries to n8n showed as pending on the Automation page and never became recorded.
- Cause: the worker marks an event recorded only when n8n replies with the exact result 'processed'. The n8n workflow did the work but did not send that reply, so the worker correctly kept the events pending and would retry. The fault was the missing final step in the workflow, not the worker.
- Fix: I stopped the worker timer, added a 'Return Processed Response' step at the end of the n8n workflow, republished it, and restarted the worker.
- Evidence it was fixed: the Automation page changed to Pending 0, Recorded 4, Failed 0 for the four test events. Earlier I also tested failure on purpose: with n8n not listening, attempt 1 returned HTTP 404 and stayed pending with the error saved; attempt 2 was recorded. A second send of an already recorded event is refused.

4. Design points behind it (read from the code)

- Each event has a stable key (outbox-N) and a status limited to pending, recorded or failed by a database rule.
- The worker claims one eligible event at a time with a 5-minute lease and row locking, so two runs do not take the same event.

- n8n gets a 10-second timeout. 'Processed' means recorded; 'rejected' means failed; no reply, an error, or anything else means pending. Retries back off 1, 2, 4 minutes up to 30.
- Automatic delivery has an on/off switch, off by default.

5. What I have observed and what I have not

Observed	Not observed, or not built
Failed send, then retry that recorded (HTTP 404 then success).	A lost reply after the sheet was already written (read from code, not tested).
Stuck pending events fixed: Pending 0, Recorded 4, Failed 0.	Service History tab is append-only: a repeat send could add a second row. The fix (check for an already logged event id) is a proposal, not built.
Duplicate bookings from the same test data approved twice by staff (two jobs, two events).	A database unique rule against duplicate bookings; today the human approver is the only check.
Test suite recorded as 50 passed, 5 skipped on 2026-10-08.	Push alerts (there is a status page and worker log lines only). Search, ranking and language-model accuracy tests are not part of this project.

6. How this maps to your post

You ask for	My honest position
APIs, integrations, SQL, duplicates and freshness	Direct: API integrations and PostgreSQL loads in earlier work; stable event keys, retries and a status page here.
Keep it running: retries, logging, deployment, background processing	Direct: timer-driven worker, retries, logs, migrations, server deployment.
Search, filters, matching, ranking with reasons	Partial: matching pipelines with human review in earlier work; not in this project.
Language-model extraction and accuracy checks	Partial: AI-assisted categorization with human review; I have not run a formal accuracy evaluation.
Authentication, cost controls, monitoring	Partial: authentication is not demonstrated in this project; cost limits learned from OpenAI rate limits in earlier work; a status page and logs, no push alerts yet.

7. Availability

Full time, 40 hours a week, with full overlap with New York working hours (about 9 pm to 5 am Manila time).