CASE STUDY

A three-stage Claude pipeline that produces and delivers a finished PDF unattended

Built for LaunchQuack · Make.com, Claude API, Typeform, Doppio, Gmail

The problem

LaunchQuack sells a detailed written assessment. Before this system, a specialist had to read the client's whole submission and write the report by hand. That took two to six weeks and burned $500 to $2,000 of billable time. Quality moved with whoever picked up the file, so two clients paying the same price could get very different reports.

The constraint

Four things had to work at once, and each one made the others harder. The report runs long and changes with the input, so one model call can't produce it. Every call is stateless and remembers nothing from the call before. The finished document still has to read like one person wrote it. It also had to arrive as a print-quality branded PDF. And nobody could sit in the middle of it, or the cost lands right back where the manual version started.

What I built

An intake form collects the submission and the details the assessment depends on, then fires a webhook into a Make.com scenario. Three Claude calls run in sequence. Each one carries its own system prompt of several thousand words.

Getting all three to sound alike meant writing the voice rules once and copying them word for word into every prompt. They're versioned together, so changing a rule in one place changes it everywhere.

Before any of the visible text, the model writes a counted inventory into a comment that never renders. It works through the document zone by zone. Whatever it concludes after that has to match the counts. That's what moved the output from confident guessing to something you can defend.

The three outputs assemble into a styled HTML document. That renders to PDF through a third-party API, and Gmail sends it out with the download link mapped in.

```
Typeform webhook → Claude 1A → Claude 1B → Claude 2 → HTML assembly → Doppio /render/pdf/sync → Gmail delivery
```

What broke, and what fixing it took

- **Generated HTML won't survive a raw JSON body.** Model output is full of characters that break JSON strings, and you can't predict which ones turn up. Every escape chain I wrote by hand worked on one payload and failed on the next. The fix was to stop escaping by hand and let the platform build the payload through a typed data structure.
- **The render API checked its schema strictly and the platform's field names didn't match.** It rejected the request for a capitalized property. The next try failed because a property sat one level too high. The one after that failed on an enum value that was off by a single character. Every rejection cost a full test cycle, and those cost real money because each run spends live credits.
- **Characters came through corrupted.** The first line of the rendered document showed up as replacement glyphs on an otherwise blank page. It happened the same way on every run. Corruption that repeats exactly rules out randomness and points at encoding. The request needed its character set declared.

- **Modules that passed on their own came back empty in the full run.** Testing a downstream module alone shows it cached data from the last run instead of live data, so a broken chain looks healthy. Fixing it meant tracing the real output key each module emits rather than the obvious one. Then I sent a sentinel value through the pipeline to find where it went dark.

> None of those failures came from the model. They came from the layer underneath it, where encoding and schema validation live. That's where this kind of project usually stops. Now I check the exact payload shape against a live endpoint with a bare request before I build a single module around it. That's the first thing I do on client work, and I charge for it in discovery.

The result

3	**15+**	**0**
chained model calls, one continuous voice	page branded PDF, generated and delivered	people between submission and inbox

The pipeline runs unattended. A submission comes in and a branded multi-page assessment goes back out. It covers document structure and section-level notes, plus comparables and a closing recommendation. Turnaround runs in minutes where the manual version took weeks.

The system also runs a check the manual version never had. It reads the generated text for the patterns that make someone conclude a document was machine-written. Density and clustering across the whole document decide the call, rather than single instances. That calibration matters, because deliberate repetition would otherwise get counted as machine output.

What this means for your business

Most service businesses have at least one process that runs the same way every time. Something comes in, a person works through it using judgment and rules, and a document goes out. In one business that's a client intake turning into a structured brief. Somewhere else it's a claims file becoming a reviewed summary with the exceptions marked. In another it's data from four systems building itself into a monthly report.

What a system like this costs depends on how many workflows connect and how strict the output has to be.

Waylan Webb

Seqlo Automations · Florida
waylan@seqloautomations.com · seqloautomations.com

Available for direct engagements and white-label delivery under agency brands.