PRODUCT ENGINEERING CASE STUDY

Share Expenses

An automatic shared-expense ledger for two roommates — built end to end, from parsing raw bank SMS to a working app people use every month.

Cloudflare Workers | D1 (SQLite) | SMS Parsing Engine | 31 Automated Tests

Live app → pg-expenses.jayyadav-work.workers.dev

Splitting rent-adjacent expenses shouldn't need a spreadsheet

Two roommates, one running tab

Groceries, milk, bills — whoever's turn it is pays. Keeping track of who's ahead relies on memory and goodwill.

The data already exists — in bank SMS

Every UPI debit triggers an SMS with the amount, merchant and time. It's all there — just buried in the inbox.

Manual tracking doesn't survive real life

Spreadsheets get forgotten, WhatsApp tallies drift, and “who owes what” quietly becomes a monthly argument.



SCATTERED ACROSS DOZENS OF MESSAGES A MONTH

From bank SMS to settled balance — automatically



SMS arrives

A debit alert lands and is forwarded the instant it's received.



Parsed in ms

Amount, merchant, date & UPI reference extracted; category guessed.



One tap to confirm

Shared or personal — the shop is remembered automatically next time.



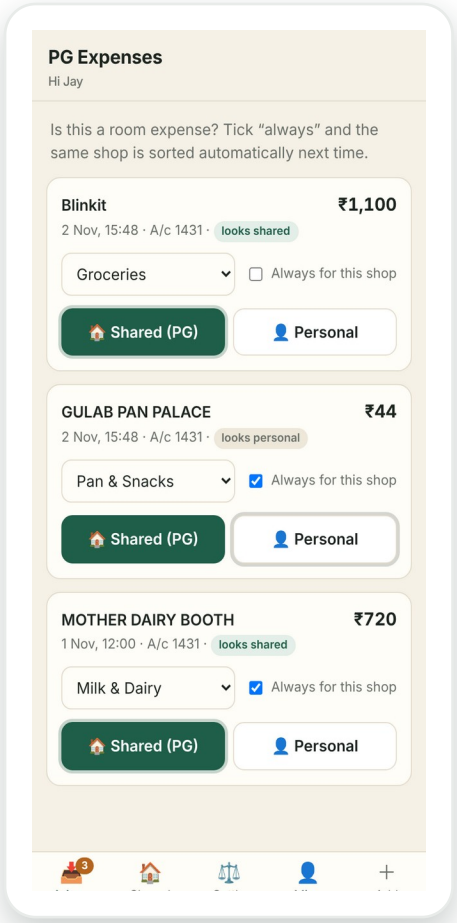
Balance updates

The running total and who-owes-whom recalculate instantly.

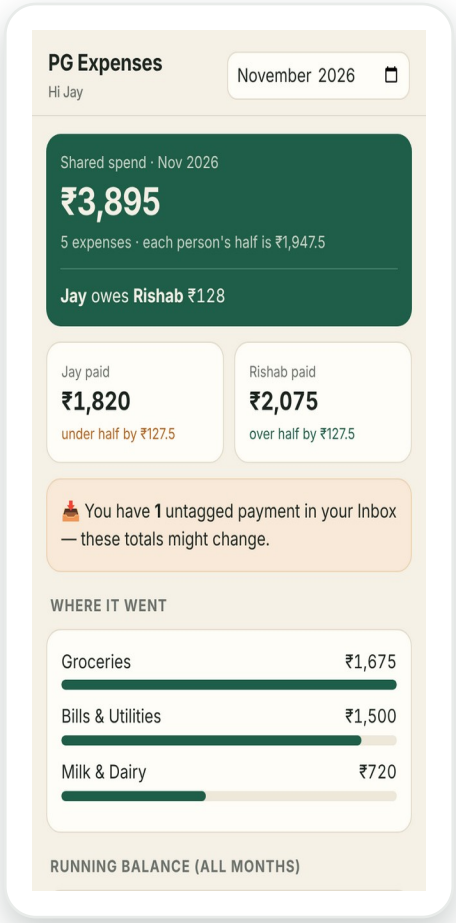


Rejected SMS never pollute the ledger. OTPs, credits, refunds, failed or declined transactions and mandates are recognized and filtered before they ever reach the database.

The actual product — not a mockup



Inbox — one tap sorts a payment



Settle — live running balance

Smart suggestions

Each card shows a "looks shared" or "looks personal" hint based on the merchant, before you even tap.

"Always for this shop"

Tick it once and every future payment to that merchant files itself — no repeat decisions.

Private by default

Personal spends and untagged payments are visible only to the person who paid.

Regex that survives messy, inconsistent bank SMS

Tolerant field extraction

Amount, merchant, date/time and UPI reference — across ₹ / Rs / INR, multiple date formats, and SMS split across lines.

Automatic categorization

Merchant name maps to one of 10 categories — Groceries, Bills & Utilities, Milk & Dairy and more.

Per-person merchant memory

One correction teaches the parser; the same shop auto-sorts the same way for that person from then on.

Noise filtered at the door

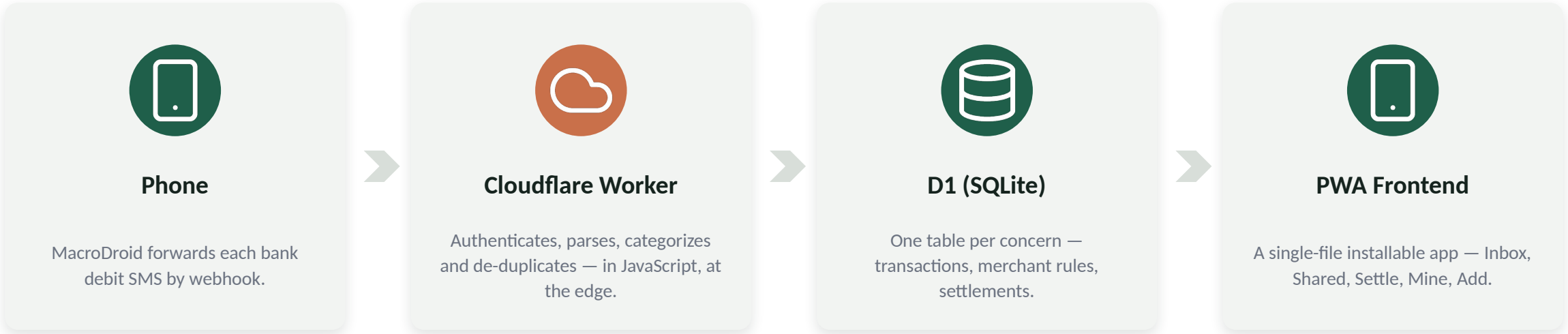
Credits, OTPs, failed/declined transactions and payment mandates are recognized and never logged as expenses.

INCOMING SMS

```
Debited Rs 365.00 from a/c X0433 on 02Nov26
08:32 via UPI to Pappi Genera. Ref
630673645011.Bal Rs 11157.71.
```

amount	**₹365.00**
merchant	**Pappi Genera**
category	**Groceries**
account	**●●●0433**
tag	**pending → shared**

A serverless pipeline, deployed at the edge



No servers to manage — the entire stack runs on Cloudflare's global network, request to response.

Built like production software, not a weekend script



Idempotent by design

Duplicate SMS deliveries and concurrent webhook retries collapse into a single transaction — never a double-charge.



31 automated tests

Parser fixtures for every bank-format edge case, plus full API and database integration tests — all passing.



Honest status codes

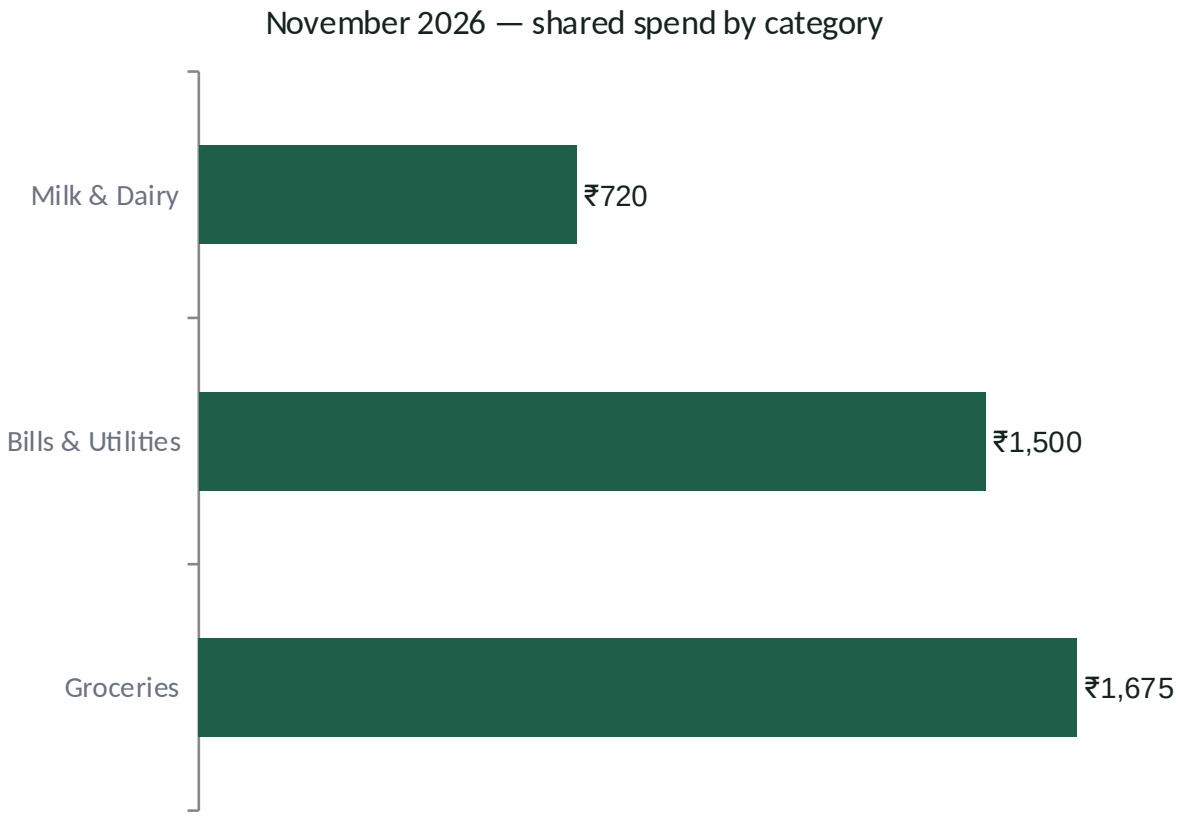
200 for success, 422 for an unparseable SMS, 500 for a real failure — never a silent false-positive.



Privacy & ownership checks

Personal spends stay invisible to the other person; only the original payer can edit or delete a transaction.

Running on real data, right now



5

expenses auto-categorized this month — zero manual entries

2

bank SMS formats supported out of the box, and extensible to more

₹3,895

tracked and reconciled automatically between two roommates in November

The stack



Cloudflare Workers

Serverless JavaScript, deployed at the edge



Cloudflare D1

SQLite, globally replicated



Vanilla JS PWA

Zero-framework, installable frontend



MacroDroid

Android SMS-to-webhook automation



Node Test Runner

31 unit + integration tests

CASE STUDY

From a messy inbox to an automatic ledger



Designed, built, debugged and shipped end to end — problem discovery, a parsing engine, a serverless backend, and a product two roommates actually use every month.

Cloudflare Workers | D1 (SQLite) | PWA Frontend | 31 Automated Tests

Live app → pg-expenses.jayyadav-work.workers.dev