

CASE STUDY 01 · IN PRODUCTION · SOLE ENGINEER

Food-ordering and delivery platform

Four native apps, two web portals and a complete serverless backend, live on the App Store and Google Play.

9 campuses in production	15,000+ orders processed	219 merchants onboarded	4 + 2 native apps + web portals
------------------------------------	------------------------------------	-----------------------------------	---

THE SYSTEM

A campus marketplace connecting customers, merchants and riders: discovery, split-tender checkout, dispatch, live tracking and settlement, with an internal admin layer for onboarding, financial policy and oversight.

MY ROLE

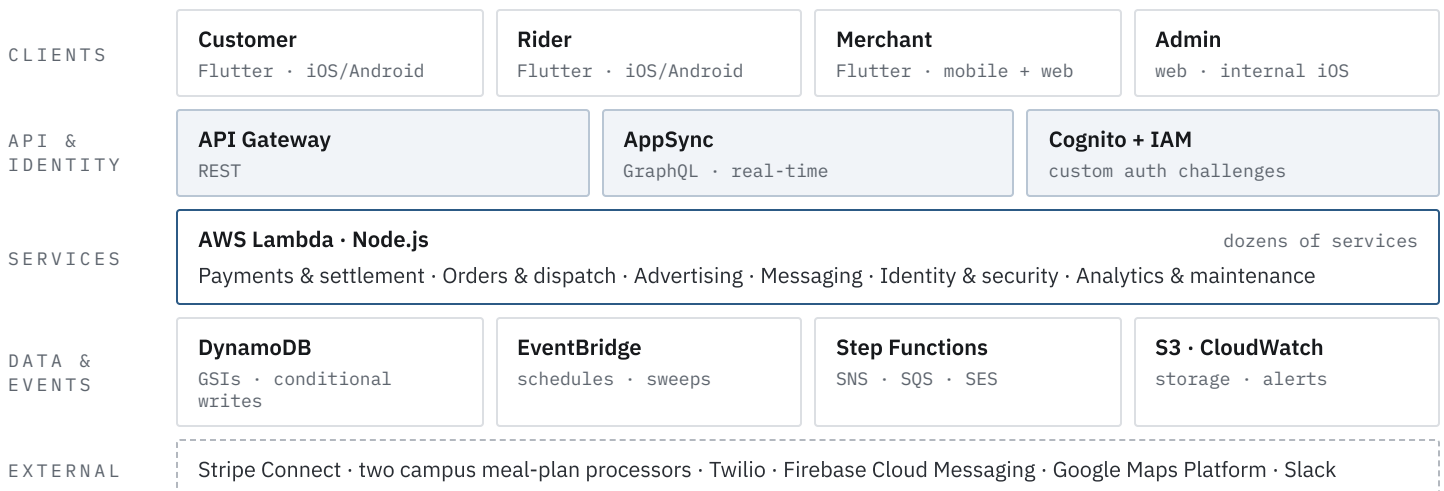
I designed, built, released and operate every part of it: the mobile and web clients, the serverless backend, the payment and settlement flow, and the release pipeline.

PRODUCTS SHIPPED

Customer app	iOS · Android	Discovery and maps, cart, promotions, split-tender checkout, live tracking, chat, rewards.
Rider app	iOS · Android	Job offers with accept-rate scoring, navigation, proof of delivery, masked calling, earnings.
Merchant app + portal	iOS · Android · web	Order queue, menu and inventory, hours, promotions, store ads, payouts.
Admin portal	web · internal iOS	Vendor onboarding, financial policy, delivery oversight, rider suspension, ads, analytics.

ARCHITECTURE

The backend is the largest part of the project and entirely mine: I designed the data model, wrote every service, and run them in production on AWS Lambda behind API Gateway and AppSync GraphQL.



MAJOR SYSTEMS

Payments and money

Stripe Connect authorise-then-capture, integer-cent fees, vendor settlement and payouts, a platform payout ledger reconciled by sweeps, two campus meal-plan processors, split tender, double-refund protection.

Identity and security

Cognito with custom auth challenges and risk auditing, OTP, campus-affiliation verification, partner token exchange, server-enforced suspension, consent audit trail, fraud sign-up cleanup.

Advertising

Store-local ad scheduling (dayparts, DST-safe) with an hourly, deduplicated ledger of views, clicks and orders.

Cost control

AWS budget and anomaly alerts, GCP billing and Maps spike alerts to Slack, a tiered Maps cache.

Support

An assistant that answers from a FAQ knowledge base, tickets pre-filled with the chat, ticket tracking, support ratings and support credits.

Orders and fulfilment

Cart-to-settlement lifecycle, rider dispatch, locked rider deposits with a 12-hour sweep, pre-orders, scheduled vendor restocks and “sold out until” times.

Messaging

Email with bounce handling, guarded SMS, push, bulk campaigns, real-time three-way chat, masked numbers.

Data and operations

Funnel analytics, merchant and campus reports, table archiving, maintenance jobs, Slack error alerts.

Audit and cost visibility

Admin and vendor audit logs of account and configuration changes. A Cost & Resources dashboard of AWS Cost Explorer snapshots by resource and period; refresh sits behind IAM auth because each call costs money.

Disputes and chargebacks ROLLING OUT

One case per Stripe dispute, each webhook event applied once; evidence, a responsibility decision, settlement postings and a 6-hourly reconciliation sync. Also manual campus-card claims. Piloting in dev.

SELECTED ENGINEERING

01	PAYMENTS	02	IDEMPOTENCY	03	CONFIGURATION
Split-tender correctness		A double-charge from retries		Server-driven checkout	
One checkout can draw on meal plan, declining balance and a card. Fees are summed in integer cents, so processor, ledger and reconciliation agree to the penny.		Retries were regenerating the idempotency key, which caused a double-charge in production. Traced and eliminated.		Per-merchant policy decides what checkout renders: tax, tips, meal-plan coverage, second-tender rules. Absent settings default to enabled, so merchants need no migration and pricing stays server-side.	
04	RELEASE	05	RELIABILITY	06	CONCURRENCY
Releasing a four-app fleet		Live tracking on unreliable Wi-Fi		Two riders, one order, the wrong card	
One command ships every app to TestFlight, the App Store and Google Play, with dry-runs and recovery.		Subscription sockets could stay open but silent. A watchdog repairs them, and polling and push fallbacks keep tracking live.		Simultaneous rider deposits let a penalty hit the wrong card. A conditional-write lock allows one hold per order; penalties charge only the cancelling rider; suspension is enforced server-side on every clock-in, deposit and accept.	
07	COST	08	COST	09	PAYMENTS
Cutting Maps spend within the terms		A read spike traced to a client loop		Refunds that prove they happened	
A tiered DynamoDB cache holds geocodes for 30 days and routes for 6 hours, inside Google's caching terms.		About 4 million promotion reads a day came from an app loop. Fixed server-side, with no app release.		Campus-processor refunds verify the reply, block repeats and flag unconfirmed results.	

RECENT SYSTEMS

Released to production roughly weekly, Sep–Oct 2026. Cards tagged rolling out are built but not live: they are piloting in dev behind fail-closed release locks.

10	RELEASE	11	ROLLING OUT DISPATCH	12	ROLLING OUT PAYMENTS
Shipping dark: release locks that fail closed		A 20-second offer that phones couldn't meet		Disputes as one case, applied once	
Unreleased features sit behind a global setting, an environment allow-list and a per-campus switch. Every lock fails closed, so a feature stays off unless each one says on. Disputes, runner auto-accept, runner-fault deposit recovery and a vendor user-agreement trail are built this way and piloting in dev.		Auto-accept first pushed an offer with a 20-second window. In dev testing, backgrounded apps accepted 47–64 seconds late. Now the server reserves the order for the top-ranked opted-in runner before the campus broadcast, places the deposit, accepts, then pushes “New Order Auto-Accepted” with estimated earnings. Any order or system failure falls back to the normal broadcast. 38 unit tests.		A new Lambda service turns Stripe dispute webhooks into one case. Each event is applied once, and an older event never rolls a case back. Charges are auto-classified as customer order, vendor settlement or runner deposit; evidence is gathered and one responsibility decision recorded. A 6-hourly sync reconciles with Stripe. 67 unit tests.	
13	DATA	14	RELIABILITY	15	OPERATIONS
An audit log that stopped scanning the table		Telling “offline” from “our servers are down”		Restocks that run themselves	
The logs page full-scanned a ~64k-item mutations table and was never usable. Audit entries (who changed which account or setting, and when) now come from a sparse DynamoDB index that holds only audit-worthy rows. TTL handles retention; a one-time cleanup backed up every row before deleting and never touched audit rows.		A connection monitor tells no internet apart from BYPPO servers failing. Pollers pause while offline or when the tab is hidden, never overlap, and resync immediately on reconnect.		Vendors schedule restocks and set “sold out until” times. A scheduled Lambda applies them, sends a Slack notice and writes an audit entry.	

STACK

Mobile	Flutter, Dart, Provider, GetX, BLoC, Google Maps, Stripe SDK, Apple Pay, Google Pay, biometrics, Firebase Cloud Messaging, fastlane
Backend	Node.js, AWS Lambda, API Gateway, AppSync GraphQL, DynamoDB (conditional writes, GSIs), Cognito, S3, SES/SNS, Step Functions, EventBridge, CloudWatch, AWS Budgets and Cost Anomaly Detection, AWS Cost Explorer, Amplify
Integrations	Stripe Connect, campus meal-plan processors, Twilio, Firebase, Google Cloud (Maps Platform, Billing, Monitoring, Pub/Sub), Slack alerting
Released on	App Store · apps.apple.com/us/developer/https-www-byp-po-com/id1755669272 Google Play · play.google.com/store/apps/developer?id=Byppo+Technologies,+LLC

CASE STUDY 02 · BUSINESS PROJECT · PRE-LAUNCH · SOLE ENGINEER

D-Touch Health Platform

A clinic management SaaS for many independent clinics. Four React Native Web apps, one Postgres backend, and every authorization rule enforced in PostgreSQL.

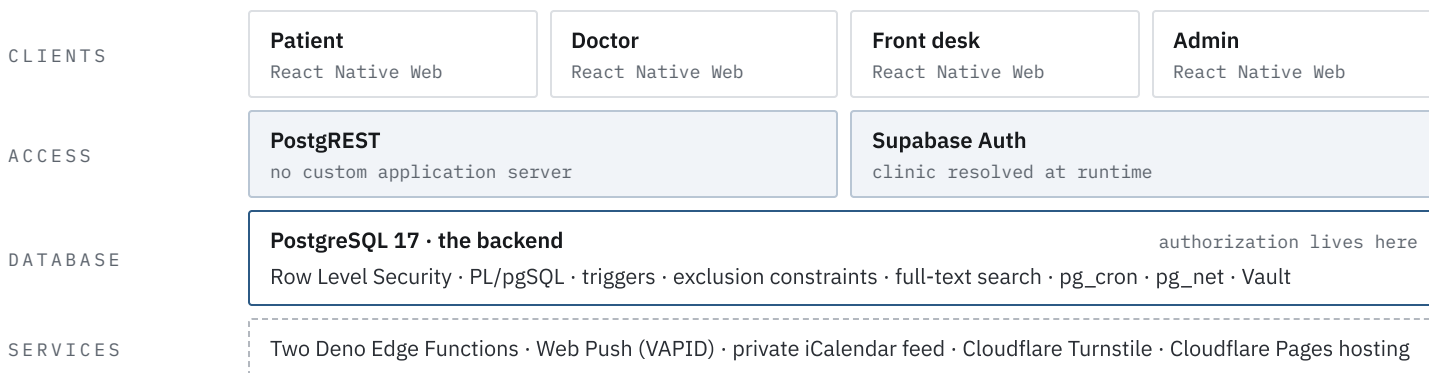
github.com/kdgbduyu28/health-platform-fegithub.com/kdgbduyu28/health-platform-be

77 RLS policies across 28 tables	100+ PL/pgSQL functions	50+ triggers	32 SQL migrations	660+ database checks	187 Vitest tests
--	-----------------------------------	------------------------	-----------------------------	--------------------------------	----------------------------

WHAT IT IS

Four apps (patient, doctor, front desk, admin) share one Supabase Postgres backend, deployed as four web builds on Cloudflare Pages for landscape tablets and desktops. Once one dental practice with three hard-coded clinic identities, it is now multi-tenant: clinics are data rows, clinic type switches dental features on per vertical, the user's clinic resolves at runtime, and each clinic has a slug-routed public page. D-Touch Dental is the first clinic, live on test data ahead of launch.

ARCHITECTURE



MULTI-TENANCY AND AUTHORIZATION

The database is the backend. There is no custom application server and no role logic in client queries: every rule is enforced by PostgreSQL.

- 77 Row Level Security policies across 28 of the 30 tables; tables deny by default.
- Clinical notes are clinician-only, in imports and exports too; patients get a separate note.
- Roles scoped per clinic membership: patient, doctor, front desk, admin, branch-locked admin.
- 660+ database checks verify tenant isolation and every role's access; Supabase's linter reports no errors.

DATABASE INTEGRITY

Exclusion constraints

Time-range constraints (btree_gist) make double-booking impossible for any doctor or patient chart; cancelled slots stay bookable.

Service snapshots

Appointments snapshot the service name, so renaming or retiring a service doesn't rewrite past visits.

Composite foreign keys

Keep each appointment's doctor and service inside the appointment's own clinic.

Transactional walk-ins

The walk-in function runs as SECURITY INVOKER in one transaction under RLS, so a lost slot race leaves no orphan chart.

CORE FUNCTIONALITY

Scheduling	Slot availability from working hours, doctor leave and closures; pending → confirmed → arrived → in consultation → completed; rescheduling, walk-ins, month, week and day views, and a drag-and-drop status board.
Clinical	Tooth charting (FDI or Universal), a searchable patient directory with recall, unpaid-balance, missing-contact and follow-up flags, doctor handover, doctors registering patients themselves, and a four-step intake a guardian can complete, after which the tablet locks behind a staff PIN.
Billing	Completed visits add their services to a draft invoice at catalogue price; GCash, QR Ph, bank transfer or cash; printable patient statements; an expense ledger reports gross, pending and net.
Data	CSV import and export of patients, visit history and dental charts, for clinics moving off paper or spreadsheets; a trigger-written audit log.
Access	Single-use staff invite codes with admin approval, reusable patient join codes, password reset by emailed code, a Cloudflare Turnstile check on sign-in.
Privacy	Data Privacy Act (RA 10173): sign-up records which Privacy Policy and Terms version each person was shown; consent to process health information is collected separately, per clinic, as the National Privacy Commission recommends.
Notifications	Web Push, an in-app inbox, per-topic mutes, scheduled visit reminders, and a private iCal feed for any calendar app.

SELECTED ENGINEERING

01

TENANT ISOLATION

A role that leaked across tenants

Roles lived on the global user profile, so one clinic's admin could promote a person at another. Roles now sit on each clinic membership; a trigger restricts branch-locked admins, and a guard keeps every clinic's last full admin.

03

NOTIFICATIONS

Notifications sent from the database

One SQL function writes the inbox row, checks mutes and calls a Deno Edge Function via `pg_net` (secret in Vault) for VAPID Web Push; `pg_cron` sends reminders and prunes. The iCal feed has a resettable token and hides service names.

05

AUDIT

An audit log no client can skip

Database triggers log every insert, update and delete on patient, visit, billing and clinic records, so the log covers every path in: apps, RPCs, imports and cron jobs. Updates store only the changed fields; bookkeeping-only changes are not logged.

02

PRIVACY

Overlap rules that respect privacy

Scheduling clashes are checked per patient chart, not per person; a person-wide rule would let one clinic discover that the same patient has a booking at another.

04

DATA

Imports you can preview, then undo

A dry run sends every row through the real constraints, then rolls the whole call back, so the preview shows exactly what will happen. Real imports run in chunks; each row succeeds or fails on its own, and the whole batch can be undone.

FRONTEND

Rewritten from Flutter to React Native for Web with Expo Router and TypeScript: about 30,600 lines; four apps with 78 routes built from one shared package. Data goes through TanStack Query and supabase-js; UI uses React Native Paper. 187 Vitest tests.

STACK

Client	Expo Router, React Native for Web, React 19, TypeScript, TanStack Query, React Native Paper, supabase-js, date-fns, Vitest
Data	Supabase, PostgreSQL 17, Row Level Security, PL/pgSQL, btree_gist exclusion constraints, full-text search, pg_cron, pg_net, Vault, PostgREST, Supabase Auth, Supabase CLI migrations
Services	Supabase Edge Functions (Deno, TypeScript), Web Push (VAPID), iCalendar feeds, Cloudflare Turnstile
Deploy	Cloudflare Pages (one Pages project per app)

CASE STUDY 03 · LIVE · PERSONAL PRODUCT · SOLE ENGINEER

Presyo

Palengke, pump and household prices near you, built from Philippine government monitoring data. A Python pipeline and an Expo app, with no backend.

github.com/kdgbduyu28/Presyo presyo.pages.dev

11	1,736	39	2× daily	0
data sources · 9 automated	places mapped (PSGC)	wet markets geocoded	automated updates	servers to run

WHAT IT IS

Food, fuel, LPG, electricity, water and fare prices where you are, plus BSP exchange rates. A Python ingest normalizes government and utility reports into history in the repo; a build step precomputes static files, and the Expo app picks the nearest place on-device.

DATA SOURCES

PRICES	SOURCE	DETAIL	UPDATED
Food: rice, meat, seafood, vegetables, fruit, pantry (Metro Manila)	DA Bantay Presyo	per wet market + NCR average	daily
Same categories, rest of the country	PSA OpenSTAT	province / city / region	monthly
Gasoline, diesel, kerosene	DOE pump prices	city/town × brand	weekly
Fuel price adjustments	DOE prior notices	per oil company	weekly
LPG 11 kg	DOE LPG monitoring	city/town × brand	monthly
Electricity	DOE rates	per distribution utility	monthly
Grocery SRPs	DTI SRP bulletin	national	as published
Foreign exchange	BSP reference rates	31 currencies + USD buy/sell	daily
Residential electricity bill	Meralco bill tables	full bill at 50–5,000 kWh	monthly
Water (quarterly)	MWSS concessionaires	Maynilad / Manila Water zones	by hand
Public transport fares	LTRFB	jeepney · bus	by hand

ARCHITECTURE

SOURCES	DA · PSA · DOE (fuel, LPG, power, adjustments) · DTI · BSP · Meralco — PDFs, CSVs, Google Sheets
INGEST	Python 3.14 · uv · httpx · pdfplumber · shapely fetch → parse → normalize → data/prices/<source>/*.json
BUILD	Precomputed per-source static JSON PSGC gazetteer with centroids · headline items + sparklines
CLIENT	Expo / React Native · iOS, Android, web nearest place on-device · Cloudflare Pages
	GitHub Actions · 12:00 and 20:00 Manila · test → fetch → commit → build → deploy

SELECTED ENGINEERING

01	DATA	02	GEOGRAPHY	03	ARCHITECTURE	04	RELIABILITY
	Parsing government PDFs DA and DOE publish prices as PDF tables that differ by region and change format. Per-source parsers with pdfplumber normalize items, units and local/imported origin into one schema, with parsing tests in pytest.		Matching messy place names Reports write "Sta. Cruz", "City of Cebu" or "Taguig". A normalizer folds abbreviations and city prefixes and matches them against 1,736 PSGC places whose centroids are computed from boundary polygons.		No backend, still fast Everything heavy is precomputed, so the app never filters large files: it resolves the nearest place locally and fetches one small file per source. Hosting is static.		One site down doesn't stop the rest Each source runs on its own; failures are logged as GitHub Actions warnings and the build still ships with the data that did arrive. New data is committed only when something changed.

APP FEATURES

Today card with price changes and sparklines · palengke basket priced per nearby market · fuel by city and brand · LPG · electricity by utility · Meralco bill estimator · forex · water · fares · dark mode · desktop sidebar layout · saved preferences.

STACK

Pipeline	Python 3.14, uv, httpx, curl_cffi, pdfplumber, shapely, pytest
App	Expo, React Native, React Native Web, TypeScript, Expo Router, expo-location, Zustand, react-native-svg, AsyncStorage, Vitest
Ops	GitHub Actions (scheduled), Cloudflare Pages, Wrangler

CASE STUDY 04 · CONCEPT BUILD · SOLE ENGINEER

Pabili

A social-commerce app in the style of Philippine marketplaces: shop, check out and track parcels, without real payments. iOS, Android and web from one Expo codebase.

github.com/kdgbduyu28/Pabili pabili.pages.dev

37	3	25	2	3
screens	platforms, one codebase	unit tests	languages (EN / FIL)	EAS build profiles

WHAT IT IS

Pabili is a full shopping experience built to show mobile product engineering end to end: catalog and search, a multi-shop cart, a checkout engine with real promotion rules, order tracking, chat with shops, and the engagement mechanics Filipino shoppers know from big marketplaces. Payment is always ₱0. It runs on Expo with file-based routing, ships as an installable web app on Cloudflare Pages, and has EAS profiles for Android and iOS builds.

ARCHITECTURE

CLIENT	Expo Router · 37 screens iOS, Android, web (PWA)
STATE	Zustand store persisted to AsyncStorage cart, orders, coins, vouchers, settings
DOMAIN	Pure TypeScript modules checkout, orders, notifications, achievements, Wrapped tested with Vitest
PLATFORM	expo-notifications · expo-haptics · expo-audio · expo-image-picker · expo-sensors · react-native-svg
DELIVERY	EAS (development, preview APK, production) · one-command Cloudflare Pages release

FEATURES

Shopping	Catalog with variants, search, voice and photo search, shop pages, likes and follows, Q&A, chat with shops
Checkout	Multi-shop cart, bundle deals, shop and platform vouchers, free-shipping thresholds, express delivery, coins, gift wrap
Orders	Timeline from placed to delivered, parcel map, delays with compensation vouchers, ratings with photos, unboxing
Engagement	Flash deals, mega-sale countdowns, live selling, group buy, Slash It, daily check-in, spin, garden, shake, monthly "Wrapped"

SELECTED ENGINEERING

01	CHECKOUT	02	NOTIFICATIONS	03	RELEASE	04	UX
Promotion math as pure functions		A feed derived from state		Expo web on Cloudflare Pages		Theming and Filipino copy	
Checkout groups lines by shop, then applies bundle discounts, shop vouchers, shipping fees and free-shipping gaps, the platform voucher and coins in a fixed order. It's pure TypeScript, so 19 Vitest cases cover the money paths.		One function builds the notification feed from orders, cart, restock alerts, group buys and check-ins, and schedules local phone notifications that deep-link to the right screen. The schedule has its own tests.		Wrangler never uploads folders named node_modules, so the icon fonts Expo exports there returned 404 and every icon broke. The release script moves them and rewrites the bundle paths before deploying.		Light, dark and system themes through themed() style sheets. Text runs through t(), keyed by the English wording, so any untranslated string falls back to English cleanly.	

STACK

App	Expo SDK 57, React Native 0.86, React 19, TypeScript, Expo Router, Zustand, Reanimated, Gesture Handler, react-native-svg, expo-image
Native	expo-notifications, expo-haptics, expo-audio, expo-image-picker, expo-sensors, AsyncStorage
Quality	Vitest, tsc typecheck
Delivery	EAS Build (dev client, preview APK, production), Cloudflare Pages, Wrangler, PWA manifest

Other work · Side and skills projects are public at github.com/kdgbduyu28. Professional work is under NDA.