PORTFOLIO

Hanifsyah Solehuddin Nazir

Backend Engineer (Go) · Jakarta, Indonesia

hanifsyah-portfolio.my.id | github.com/hanifsyahsn | linkedin.com/in/hanifsyahsn | wakatime.com/@hanifsyahsn | hanifsyahsn@gmail.com

First the case studies: what each system had to solve, what I built and the decisions that mattered. Then the work log, with everything else. Internal systems are described without screenshots, internal addresses or confidential figures.

CONTENTS

CEMINDO

MemoFlow — Delivered · continuous improvement · 2026

Company KPI Dashboard — Delivered · continuous improvement · 2026

P2H: Heavy-Equipment Inspection — Delivered · continuous improvement · 2025 – 2026

TPS Online: Customs Integration — Delivered · continuous improvement · 2026

COP: Permit to Work — In progress · 2026

Single Sign-On & Identity Provider — In progress · 2025 – 2026

STYLE THEORY

Janio Logistics Integration — Nov 2023 – Jan 2024

Percentage Discounts on Stripe & Xendit — Jun – Aug 2024

GCP → AWS Migration — Jan – Mar 2024, Aug 2025

OTP Checkout & Performance — 2024 – 2025

Logistics Webhooks out of the Monolith — 2021 – 2022

FREELANCE

Dumons Coating Website — Delivered · continuous improvement · Oct 2024 – Present

Hörnet Lineup Website — Delivered · continuous improvement · Sep 2026 – Present

PERSONAL

Go Vertical-Slice Boilerplate — Continuous improvement · 2026

This site (React SSR Boilerplate) — Continuous improvement · 2026

MORE WORK, ENTRY BY ENTRY

Style Theory · 16 entries — Feb 2023 – Oct 2025

Style Theory (via Padepokan 79) · 7 entries — Aug 2021 – Aug 2022

Padepokan 79 · 5 entries — Aug 2020 – Feb 2023

CEMINDO

Cemindo · 2026 · **Delivered · continuous improvement**

MemoFlow

Sole engineer: Go backend and React SSR front end

Company memos written in a rich-text editor, approved along a chain, e-signed through Mekari and published as PDFs on the letterhead.

THE PROBLEM

A company memo needs a number in the register, a multi-step approval, a legally valid signature and a clear record of who approved which version, and when.

WHAT I BUILT

- A TipTap editor feeding a Go service that renders HTML to an A4 PDF with headless Chrome (chromedp), stamps a watermark and uploads it to S3-compatible storage.
- Approval chains with delegation, recall, cancel and expiry, plus registrars who keep the memo register.
- Mekari e-signature integration with HMAC-signed requests. Signature positions are found automatically by reading marker coordinates from the PDF's text layer.
- Notifications in the app (SSE), by debounced email and digest, and on Telegram, each with per-user preferences.
- Publishing and signing run through outboxes with retries, bounce handling and a manual-publish fallback.

KEY DECISIONS

- When CSS backgrounds didn't survive Chrome's PDF output, the watermark moved to a separate stamping step on the finished PDF.
- Each submit freezes a memo version, so approvers always sign exactly the text they reviewed.
- Business rules live in the backend, and the front end follows its JSON:API contract instead of copying the rules.
- 59 feature slices, 80 migrations and 540 test files, with the domain rules written down per topic.

Tech: Go, Gin, PostgreSQL, chromedp, Mekari e-Sign, S3 (Wasabi), SSE, React SSR, TipTap

On the site: hanifsyah-portfolio.my.id/portfolio#memoflow

Cemindo · 2026 · **Delivered · continuous improvement**

Company KPI Dashboard

Rewrote the backend and the web app

Clinker and cement production, domestic sales and safety KPIs shown on office wall screens, with a separate confidential view for the board.

THE PROBLEM

The office screens must never show confidential tonnages. The first version had one API for every screen, synced every source hourly whether or not it had changed, and returned 0 when a value couldn't be computed.

WHAT I BUILT

- A Go API with two faces: a public, rate-limited API that only ever returns percentages, and a management API (behind an API key) with the actual figures.
- Scheduled and manual syncs from the ERP, Excel files on an SMB share, and a Google Drive folder of overlay videos, so the dashboard only ever reads its own Postgres.
- A React SSR wall-screen app with rotating banners, a BMKG weather theme per site, and an auto-reload when a new version is deployed.

KEY DECISIONS

- Confidential values never leave the server for the public screen. The API key is only added by the server-side proxy.
- Each KPI card is compared with its target to date, prorated per day, and a value that can't be computed is null instead of 0.

Tech: Go, PostgreSQL, SMB, Google Drive API, React SSR, Redux, Vitest

On the site: hanifsyah-portfolio.my.id/portfolio#kpi-dashboard

---

Cemindo · 2025 – 2026 · **Delivered · continuous improvement**

P2H: Heavy-Equipment Inspection

Main developer of the backend and the CMS

Daily pre-use inspections of heavy equipment, follow-up investigations and fitness checklists, across several business units and sites.

THE PROBLEM

Operators check every unit before a shift. Failed checkpoints have to become investigations, units have to move between sites, and safety teams need reports per site.

WHAT I BUILT

- The Strapi backend: checkpoints, investigations with attachments, unit availability history and site moves, Bugar Selamat fitness checklists, Excel exports and email to the safety team.
- A redevelopment to support multiple business units, each with multiple sites.
- A responsive React CMS for units, groups, sites, departments, users and shifts.

KEY DECISIONS

- Moved the access policies into middleware, and a user's token is revoked when their account changes, so a role change takes effect immediately.

Tech: Strapi, Node.js, MySQL, Redis, React, Vite, Docker

On the site: hanifsyah-portfolio.my.id/portfolio#p2h

Cemindo · 2026 · **Delivered · continuous improvement**

TPS Online: Customs Integration

Sole engineer

Bonded-warehouse documents (BC 1.1 inbound, SPPB/PIB outbound) drafted, validated and submitted to Indonesian Customs over SOAP/XML.

THE PROBLEM

Every vessel arrival and cargo movement at the bonded warehouse (TPS) has to be reported to Bea Cukai. The documents depend on each other, and a wrong quantity is a compliance problem.

WHAT I BUILT

- A Go REST API that follows the domain: header (vessel) → bill of lading → inbound and outbound documents, each drafted and then sent.
- A SOAP client that builds the XML payload, wraps it in an envelope and reads Bea Cukai's result.
- Sign-in through the company SSO (JWT verified against JWKS, session allowlist in Redis), plus an activity log of who did what.

KEY DECISIONS

- Sending validates the whole document chain first. Available weight is computed inside a transaction with row locks, so two outbound documents can't take the same cargo.
- A feature flag turns the live customs call on or off, so the whole flow could be tested without filing anything real.

Tech: Go, Gin, PostgreSQL, SOAP/XML, Redis, sqlc, testcontainers

On the site: hanifsyah-portfolio.my.id/portfolio#tps-online

Cemindo · 2026 · **In progress**

COP: Permit to Work

Sole engineer

Job Safety & Environmental Analysis and Permit to Work for the safety (SHE) team, with registration approved per plant.

THE PROBLEM

Contractors and staff register at a plant, admins of that plant approve them, and every plant must always keep at least one admin.

WHAT I BUILT

- The Go backend foundation: JSON:API 1.1 (content negotiation, include, sparse fieldsets, sort, filter), ES256 auth with rotating refresh tokens, and an admin user directory.
- The domain rules written down before any code: who approves whom, how companies are tied to plants, and the last-admin guard.

KEY DECISIONS

- Presenting a refresh token that was already used revokes the whole login, so a stolen token stops working as soon as both parties use it.
- The last-admin check locks the plant row first, so two revokes at the same moment can't both pass.

Tech: Go, Gin, PostgreSQL, sqlc, JSON:API, testcontainers

On the site: hanifsyah-portfolio.my.id/portfolio#cop

Cemindo · 2025 – 2026 · **In progress**

Single Sign-On & Identity Provider

Sole engineer: protocol design, Go service, React SSR web

One login for the company's internal apps, with per-app sessions, device limits and a logout that reaches every app at once.

THE PROBLEM

The company's internal apps needed one account per person, one place for admins to approve who may use which app, and a logout or a disabled account that takes effect in every app immediately.

WHAT I BUILT

- An authorization-token flow where the client exchanges an opaque, single-use, Redis-backed token for a session-bound access and refresh token pair.
- Apps verify ES256 JWTs against the SSO's JWKS and keep their own session allowlist in Redis.
- A global logout over Kafka (SASL): every app drops the user's access tokens right away instead of waiting for them to expire.
- Email verification through a transactional outbox with exponential backoff, and a user and app-access approval dashboard.

KEY DECISIONS

- Login takes the same time whether or not the account exists, so response timing can't reveal which emails are registered.
- Passwords are SHA-256 pre-hashed before bcrypt, so long passwords are never silently cut at bcrypt's 72 bytes.
- Cleanup workers lock rows with SELECT … FOR UPDATE, so two instances can't race on the same session.
- A rate limiter guards the auth endpoints, and the number of concurrent device sessions per user is capped.

Tech: Go, Gin, PostgreSQL, Redis, Kafka, sqlc, React SSR, testcontainers

On the site: hanifsyah-portfolio.my.id/portfolio#sso

STYLE THEORY

Style Theory · Nov 2023 – Jan 2024

Janio Logistics Integration

Full stack, end to end

A new third-party logistics provider for Singapore, from the database to the ops tooling.

THE PROBLEM

Style Theory needed Janio for Singapore apparel delivery, return and swap orders without disturbing the other providers already running.

WHAT I BUILT

- Provider rows, timeslots and coverage areas, plus a Janio bridge service for create, cancel, timeslot and webhook calls.
- A webhook listener that maps Janio statuses to internal states with idempotent transitions.
- Ops tooling in the internal web app, return-swap label exports, and alerts to Google Chat.

KEY DECISIONS

- Ran a pilot with operations (one delivery, one swap, one return) before going live.
- Delivered and Cancelled became terminal states, so a late webhook from any provider can't undo a finished shipment.

Tech: Scala (Lagom), PostgreSQL, Kafka, GraphQL, React

On the site: hanifsyah-portfolio.my.id/portfolio#janio

Style Theory · Jun – Aug 2024

Percentage Discounts on Stripe & Xendit

Backend

The voucher system extended from fixed amounts to percentages, across two payment providers.

THE PROBLEM

Marketing wanted percentage vouchers limited to particular plans, verticals and user statuses, but the system only knew fixed amounts tied to a single plan.

WHAT I BUILT

- A discount type (absolute or percentage) with validation from 0.01 to 100, plus Xendit-specific rounding.
- Multi-plan and vertical eligibility, user-status targeting, and no vouchers across regions.
- The Stripe webhook now applies vouchers per subscription item, and the invoice pipeline handles both discount types.

KEY DECISIONS

- Moved vouchers out of a loose meta field into a proper relationship, so they could be queried and reported on.

Tech: Scala (Lagom), PostgreSQL, Stripe, Xendit, Kafka

On the site: hanifsyah-portfolio.my.id/portfolio#vouchers

Style Theory · Jan – Mar 2024, Aug 2025

GCP → AWS Migration

DevOps

Backend and frontend services moved from GCP to AWS, and later an EKS upgrade.

THE PROBLEM

Every service, database and deploy pipeline had to move clouds with minimal downtime.

WHAT I BUILT

- Replaced the GCP-specific Helm configuration with AWS-native settings, and routed DNS through Route 53.
- Migrated PostgreSQL to RDS, and moved auto-deploys from Jenkins to CircleCI with artifacts in JFrog.
- Upgraded EKS from 1.31 to 1.32 in every environment, after removing deprecated APIs.

KEY DECISIONS

- Checked that every pod was ready and every service still worked before switching traffic over.

Tech: AWS (EKS, RDS, Route 53), Kubernetes, Helm, Terraform, CircleCI

On the site: hanifsyah-portfolio.my.id/portfolio#aws-migration

---

Style Theory · 2024 – 2025

OTP Checkout & Performance

Frontend (React SSR)

Checkout and sign-in rebuilt around OTP verification, then made faster.

THE PROBLEM

Sign-in, sign-up and checkout were separate flows that behaved differently by region, and the checkout page made duplicate API calls.

WHAT I BUILT

- One checkout flow with OTP that sends each user to the right next step: new, subscriber, ex-subscriber or window shopper.
- Vouchers applied automatically from the URL, for guests too, plus saved-card reuse across verticals.
- Analytics events to Mixpanel, Meta, MoEngage and Braze, including abandoned checkout.

KEY DECISIONS

- The voucher validation API is now called once per page load, and UI render time dropped below 100 ms.

Tech: React (SSR), GraphQL, GTM, Mixpanel, MoEngage

On the site: hanifsyah-portfolio.my.id/portfolio#checkout

Style Theory (via Padepokan 79) · 2021 – 2022

Logistics Webhooks out of the Monolith

Backend

Gojek and Versafleet webhooks moved from the monolith into the logistic service, with Kafka events.

THE PROBLEM

Delivery webhooks were processed inside the monolith, where duplicate and late updates made shipment history messy.

WHAT I BUILT

- New webhook endpoints with request and response kept identical to the monolith's, checked by side-by-side comparisons.
- Events published to Kafka and consumed by the shipment service: pickup and delivery dates, auto-unbook, courier details.
- Duplicate updates are filtered out, and failures are reported to Google Chat.

KEY DECISIONS

- If either leg of a swap shipment fails, the whole swap is cancelled, so no customer is left with half a swap.

Tech: Scala (Lagom), Kafka, PostgreSQL

On the site: hanifsyah-portfolio.my.id/portfolio#logistics-webhooks

...and many more Style Theory was a long chapter: checkout and payments, analytics events, logistics providers, internal ops tools, infrastructure, and the system changes for the business wind-down. **All of it is in "More work, entry by entry" at the end.**

FREELANCE

Freelance · Oct 2024 – Present · **Delivered · continuous improvement**

Dumons Coating Website

Frontend and deployment

The company site of an automotive coating brand, built from a PRD and maintained ever since.

THE PROBLEM

The brand needed a site that shows who they are and what their products do, and makes it easy for customers to get in touch.

WHAT I BUILT

- A responsive site covering the vision, mission and brand identity.
- Product highlights, best-sellers, and detail pages with specs, application steps, ingredients and purpose.
- Sections for promotions, before-and-after photos and social media content, plus contact by email, WhatsApp and map.
- Ongoing updates as new products and machines arrive, deployed through Git and Netlify.

KEY DECISIONS

- Started as a Vue.js SPA. In Nov 2026 I ported it 1:1 to React SSR prerendered to static HTML: same URLs, pixel-identical pages, now fast plain HTML with tests.

Tech: Vue.js, React SSR, TypeScript, Netlify, Vitest

On the site: hanifsyah-portfolio.my.id/portfolio#dumons · Live site: dumonscoating.com/home

---

Freelance · Sep 2026 – Present · **Delivered · continuous improvement**

Hörnet Lineup Website

Design, frontend and deployment

The site for Dümons' Hörnet product lineup: an Indonesian landing page and product catalogue built to win customers and partner shops.

THE PROBLEM

Hörnet launched as its own lineup under Dümons. It needed a site with its own character that presents every product and turns workshops and shops into partners.

WHAT I BUILT

- A landing page with promo posters, real customer reviews, an FAQ with shop logos, and a partner section ("Bengkel & Toko Mitra") that opens a ready-written WhatsApp message.
- A product catalogue with a poster-style heading, a die-cut product lineup, product cards and detail pages with a colour switch.
- A hornet-painter mascot that follows the mouse, and its own brand type and colours.
- Fast images: an image CDN with srcset, versioned asset URLs cached for a year, and photo and link prefetch.

KEY DECISIONS

- Built on my React SSR boilerplate and prerendered, so visitors get plain HTML and there's no server to pay for.
- Built AI-assisted with Claude (80 of its 117 commits), with every change reviewed by me and released through development and production branches.

Tech: React SSR, TypeScript, Tailwind CSS, Webpack, Netlify

On the site: hanifsyah-portfolio.my.id/portfolio#hornet · Live site: hornet.dumonscoating.com/home

PERSONAL

Personal · 2026 · **Continuous improvement**

Go Vertical-Slice Boilerplate

Author

The foundation I start every Go service from: vertical slices, JSON:API 1.1, ES256 auth and tests against a real Postgres, all ready on day one.

THE PROBLEM

Every new backend needed the same groundwork. Rebuilding it each time wasted days and let the conventions drift from one service to the next.

WHAT I BUILT

- Vertical slices: each endpoint is its own package (route → handler → service → store → sqlc queries → tests).
- Gin and Postgres with sqlc and golang-migrate, plus JSON:API 1.1 (content negotiation, include, sparse fieldsets, sort, pagination, filter).
- ES256 JWT auth: register, login, refresh-token rotation, logout and /me.
- gomock unit tests and testcontainers integration tests, structured slog logs with request IDs, and build and deploy scripts with a graceful stop.
- A working example to copy from: auth, a notes CRUD domain and an admin user directory.

KEY DECISIONS

- Vertical slices instead of layers: each feature lives in one folder, so it can be found, copied and tested on its own.
- Recurring slice patterns (owned resource, lock-before-write, partial PATCH, list) are written down as rules that teammates and the AI both follow.
- KPI Dashboard API and COP API both started from it, which is why my services all share one shape.

Tech: Go, Gin, PostgreSQL, sqlc, golang-migrate, JSON:API, testcontainers, gomock

On the site: hanifsyah-portfolio.my.id/portfolio#go-boilerplate · Source: github.com/hanifsyahsn/go-boilerplate-vs

---

Personal · 2026 · **Continuous improvement**

This site (React SSR Boilerplate)

Author

Server-side rendering for React with Express, without a framework, prerendered to static HTML for Netlify, with nine switchable themes.

THE PROBLEM

I wanted a portfolio that loads instantly as plain HTML and costs nothing to host.

WHAT I BUILT

- One render function shared by the dev server and the static build, so both produce the same HTML.
- Behaviour tests for every page, the static build and the theme switcher.

KEY DECISIONS

- A tiny script picks your saved theme before the page paints, so the default theme never flashes first.

Tech: React, Express, Webpack, TypeScript, Tailwind CSS

On the site: hanifsyah-portfolio.my.id/portfolio#this-site · Source: github.com/hanifsyahsn/react-ssr-boilerplate

MORE WORK, ENTRY BY ENTRY

Everything else from Style Theory and the Padepokan years, newest first: the features, integrations, fixes and tests that didn't get a full case study.

Style Theory · Software Engineer II · Feb 2023 – Oct 2025

Business wind-down — Sep – Oct 2025 · Backend

System changes for the company's liquidation: no new shipments after the final day, no new subscriptions, returns kept open.

- Cut delivery timeslots from three to one, with 26 September as the last day for new or swapped shipments.
- Disabled new subscriptions and re-subscriptions, while existing shipments and subscriptions ran to completion.
- Then disabled delivery and swap shipments (and partner delivery services) but kept returns and their timeslots available.

Scala, PostgreSQL, Helm, Kubernetes, AWS

Shopify storefront: Meta tracking and Join Now — Jun – Jul 2025 · Frontend

Tracking and conversion work on the Shopify site.

- Meta events for plan, collection and product views and clicks, per vertical (apparel and bags), and domain verification so ads and the pixel work.
- A Join Now button only on the apparel and bag landing, collection and product pages, which opens the plan page with the right vertical selected.
- The cart icon is hidden everywhere except the One Time Rental pages.

Shopify, Liquid

Inventory data: missing boxes and product sync — May – Jun 2025 · Full stack

Ops can release a missing box, and product data keeps itself in sync.

- Extended the unblock API so ops can release a missing box; customers can then create return and swap shipments for it again.
- Restored each box item's original source and dropped the obsolete missing-box columns.
- A PostgreSQL trigger keeps the products table in sync with product_additional_info on every insert, update and delete.

React (SSR), Scala, PostgreSQL

Indonesia sunset — Apr – May 2025 · Frontend

Closed the Indonesian business on the web: Indonesia-only features turned off and Indonesian users can no longer subscribe.

React (SSR)

Checkout and voucher fixes — Mar – Apr 2025 · Frontend

A stabilisation round on checkout after the OTP rebuild.

- Fixed endless loading after a failed payment, the city selector, OTP submission and number formatting.
- Fixed voucher error messages, rounding and Mixpanel parameter mismatches, and removed outdated auto-applied vouchers.
- Subscribing to a second vertical now requires the valid card from the first subscription.

React (SSR), GraphQL

Analytics and CRM events — Nov 2024 – Jun 2025 · Frontend

The events the customer web sends to MoEngage, Mixpanel and Meta, so marketing and product can follow the subscription journey.

- MoEngage SDK: users are signed in and out with their account, profile attributes are synced, and subscribe, resubscribe and unsubscribe are tracked for every vertical and region.
- Mixpanel and Meta events for vouchers, OTP, registration and subscription, with mandatory parameters in one lowercase schema.
- Abandoned-checkout events when a registered user signs out, navigates away or closes the tab.
- The Mixpanel project is chosen per environment through GTM, so development data never mixes with production.

React (SSR), GTM, MoEngage, Mixpanel, Meta

MoEngage subscription preferences page Jan – Feb 2025 · Frontend

A page where users choose which emails and WhatsApp messages they get, backed by MoEngage.

- Reads the user's current preferences from MoEngage and shows email categories and WhatsApp marketing as checkboxes.
- Saves through MoEngage's bulk preferences API (email) and track user API (WhatsApp); a failed save keeps the previous state.

React (SSR), MoEngage API

App download banners and store links Jun 2024 – Apr 2025 · Frontend

Getting web visitors into the mobile app.

- AppsFlyer banners injected by script, so marketing designs and runs them without a deploy.
- One download link that sends iOS to the App Store, Android to Google Play and desktop to a landing page.
- Removed the in-house banner from the mobile checkout pages.

React (SSR), AppsFlyer

Shipment statuses and unsubscribing Jun – Aug 2024 · Full stack

A new shipment status, and a clearer view of why customers leave.

- Added the Received By Warehouse shipment status, and allowed unsubscribing at that stage.
- The internal unsubscribe page records several reasons per request, so churn can be analysed.

React (SSR), GraphQL, Scala (Lagom)

CSV exports for operations Jan – May 2024 · Full stack

Downloads that let the warehouse and customer service work ahead.

- Shipments by scheduled date (delivery or return) as CSV for customer service: new Scala endpoints, GraphQL schema and a download dialog.
- Order batching CSV with Janio's delivery zone (mapped from the postcode) and the customer's timeslot.
- Download-all return-swap shipping labels for Janio orders, for smoother driver handovers.

React (SSR), GraphQL, Scala (Lagom), PostgreSQL

Sign-up journey: thank-you pages, Braze and terms Jan – Apr 2024 · Frontend

Small changes around subscribing.

- After payment, users land on Shopify thank-you pages per vertical, where survey tools can run.
- A Braze enter_checkout event, so marketing can remind users who leave checkout.
- A region-specific Terms and Conditions link on the registration page.

React (SSR), Shopify, Braze

Vue.ai personalisation Aug – Nov 2023 · Full stack

Personalised product listings from Vue.ai, without breaking the existing clients.

- The Browse page's Occasion tab shows only banners backed by Vue.ai listings; Vue.ai responses are wrapped in our personalisation service so clients didn't change.
- Vue.ai tags in the inventory service, for filtering across apps.
- An internal tab to create personalised collections (Trending, Recommended for You, …) by slot, with new GraphQL and Go endpoints.

Go (Gorilla Mux, dbr), PostgreSQL, GraphQL, React

Singapore apparel checkout redesign Jun – Jul 2023 · Frontend

A new checkout page: order summary, about you, voucher code and payment, responsive on every device.

- Personal details are pre-filled for existing users and validated before subscribing.
- A Change Plan dialog inside checkout; the chosen plan survives a page refresh.

React (SSR)

Piing integration refactor May – Jun 2023 · Backend

Piing's API calls moved into their own service, so it could have the static IP Piing whitelists.

- Create, get and cancel order calls moved from the logistics service into a standalone time-estimation service.
- New tables for Piing's zip codes and timeslot usage; the logistics service calls the new endpoints, and every existing test still passes.

Go (Gorilla Mux, dbr), Scala (Lagom), PostgreSQL

OpenTelemetry tracing and metrics Feb – Apr 2023 · Backend

Distributed tracing and service metrics for the core Go services.

- OTEL tracing in the user, SMS, returns and search services, and metrics for returns, search and SMS.
- One naming and tagging convention, exported to Grafana with DevOps, so incidents can be followed across services.

Go, OpenTelemetry, Grafana

Smaller fixes 2024 – 2025 · Frontend

Among many others:

- Side-navigation links that wrongly led to a 404, and a checkout logo whose click area was offset.
- GraphQL error logs now include the failing URL and message for chained calls.
- Consignor valuation and submission requests now send the origin header they need.

React (SSR), GraphQL

Style Theory (via Padepokan 79) · Backend developer · Aug 2021 – Aug 2022

Account deletion lifecycle 2021 – 2022 · Backend

Deleting an account from request to anonymised data, with an audit trail.

- A review checklist that saves only on Save, can be edited only while a request is Checking or Approved, and records who changed it and when.
- Approval logs the user out on every device, revokes their tokens and notifies the team on Google Chat.
- Completion sends an SMS, then anonymises shipments, payments and events through a UserDeleted Kafka event.
- Login refuses blocked or deleted accounts with a clear message.

Scala (Lagom, Play), Kafka, PostgreSQL

Multiple addresses per user 2021 – 2022 · Backend

Addresses moved from a settings column to their own model, first in the monolith, then in a service.

- CRUD APIs with validation, default-address reassignment and coordinates stored as numbers instead of text.
- Wired into logistics (Gojek coverage by postcode and location), shipments, the customer apps and the ops tools, with no regressions in subscribe or shipment flows.
- Documented in Stoplight and covered by Go unit tests.

Go, Scala, PostgreSQL, Stoplight

Pickupp for Hong Kong apparel 2022 · Backend

A new logistics provider for Hong Kong: delivery, return and swap orders.

- Custom request payloads, the preferred timeslot in the order notes (with time zones), and responses mapped to the common create-order result.

Scala (Lagom), PostgreSQL

Logistics timeslots, tests and fixes 2021 – 2022 · Backend

Timeslot features, a broad test suite and reliability fixes across three markets.

- Customers choose a delivery timeslot, within a configurable three-day window in Singapore.
- Component tests for every provider: Gojek, Paxel, Ninjavan, SiCepat and self collect in Indonesia, DHL and Pickupp in Hong Kong, Versafleet and self collect in Singapore.
- Fixed Singapore's Collect At Your Convenience orders, missing pickup and delivery times from Versafleet, and Gojek live-tracking links.

Scala (Lagom), PostgreSQL

Inbound warehouse app 2021 – 2022 · Backend

Search, validation and access control for the warehouse's inbound app.

- Search by RFID, barcode, customer, style or designer, size and colour filters, and rack-zone mapping for items at dry cleaning.
- Strict RFID validation, a 400 for unsupported searches instead of a broad query, and forced review of returns unless cancelled.
- Missing boxes: search by user or customer name, confirm and detail endpoints, pagination, and a fix for name search in Singapore.
- Role-based access so Hong Kong partner staff only see Hong Kong data, plus component tests for every major inbound flow.

Scala (Lagom), Go, PostgreSQL

Braze CRM integration 2022 · Backend

Customer activity sent to Braze, so marketing can re-engage users in real time.

- A health check and a user-track endpoint, validated end to end with a test event before going live.

Scala, Braze

Performance and checkout fixes 2021 – 2022 · Backend

A faster user lookup and an on-demand checkout that works again.

- Rewrote the monolith's user lookup from joins to UNION ALL so it uses its indexes, and indexed apple_id.
- Fixed on-demand orders with self pick-up and self return, which couldn't be completed.

Scala (Slick), PostgreSQL

Padepokan 79 · Junior backend developer · Aug 2020 – Feb 2023

Free Trade Zone customs & excise Aug 2022 – Feb 2023 · Backend

New taxation rules for goods traded in a Free Trade Zone.

- Tax logic and validation for the updated customs and excise regulations, worked out with compliance officers.
- Auditable workflows, and less dependence on the legacy Oracle setup.

Java (Spring), PostgreSQL

XBRL platform, Indonesia Stock Exchange May – Jul 2021 · Backend

The exchange's XBRL financial-reporting platform, modernised.

- Migrated a legacy JavaServer Faces application to a modern Java framework.
- Extended the XBRL templates and tightened validation, so issuers make fewer filing errors.

Java (JSF, Spring Boot), PostgreSQL

Internal team Feb – May 2021 · Backend

Backend tasks on client projects, delivered on time and written to SOLID principles.

Java (Spring Boot), PostgreSQL

Student admissions, Loyola school Nov 2020 – Feb 2021 · Backend

The school's student recruitment system: applicant records and clear information for students and parents.

Java (Spring Boot), PostgreSQL

Backend bootcamp Aug 2020 – Feb 2021 · Training

Java and Spring Boot, database design with Spring Data, a first look at Go, Git workflows, Heroku deploys and Scrum.

Java, Spring Boot, Go, PostgreSQL, Heroku