PRODUCT-MINDED FULL-STACK DEVELOPER

Kihwan Lee

Product-minded full-stack developer building reservation, payment, CRM, and on-site POS systems as one coherent product.

Busan-based full-stack developer with over 10 years of experience building responsive, user-friendly web products with modern technologies like React, Node.js, Django, and cloud platforms. Interests span web development, game development, artificial intelligence, and robotics, with a strength in rapid prototyping that turns ideas into concrete proof of concept quickly.

Django / DRF | React / TypeScript | Expo POS | Payment Flow | CRM Operations | GCP

Download PDF | GitHub | Email

KH

Kihwan Lee
Full-stack Developer

Email	kifhan@gmail.com
GitHub	github.com/kifhan
Focus	Product systems

Focused on product systems that connect backend contracts, customer-facing web, operator CRM, field POS, and cloud operations.

PRODUCT SUMMARY

Connecting customer, operator, and field workflows through one product state

The core is not the number of screens. It is a structure where different product surfaces consistently share the same order, reservation, payment, permission, and site-scoped business state.

Customer Web

Purchase and Reservation

Customers can browse products, pay, reserve, and review purchase history in one connected flow.

CRM

Operations Management

Operators can handle reservations, orders, payment errors, statistics, and site-scoped permissions quickly.

POS

On-site Sales

On-site payments, order state, terminal settings, and recovery flows are designed for tablet and web operation.

Operations

Verification and Operations

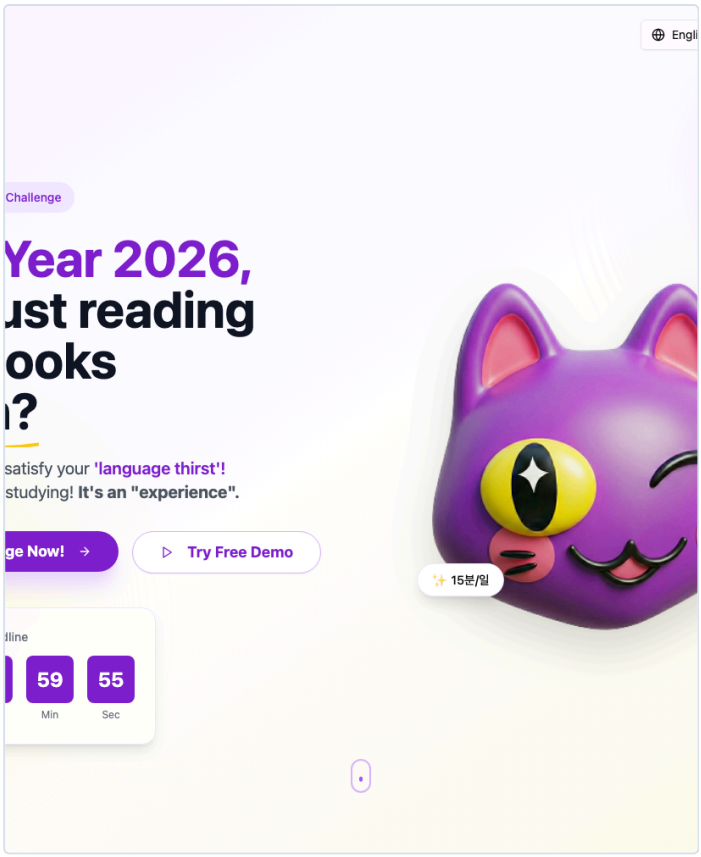
Operational risk is managed through GCP, Cloud SQL, Storage, deployment scripts, and durable work records.

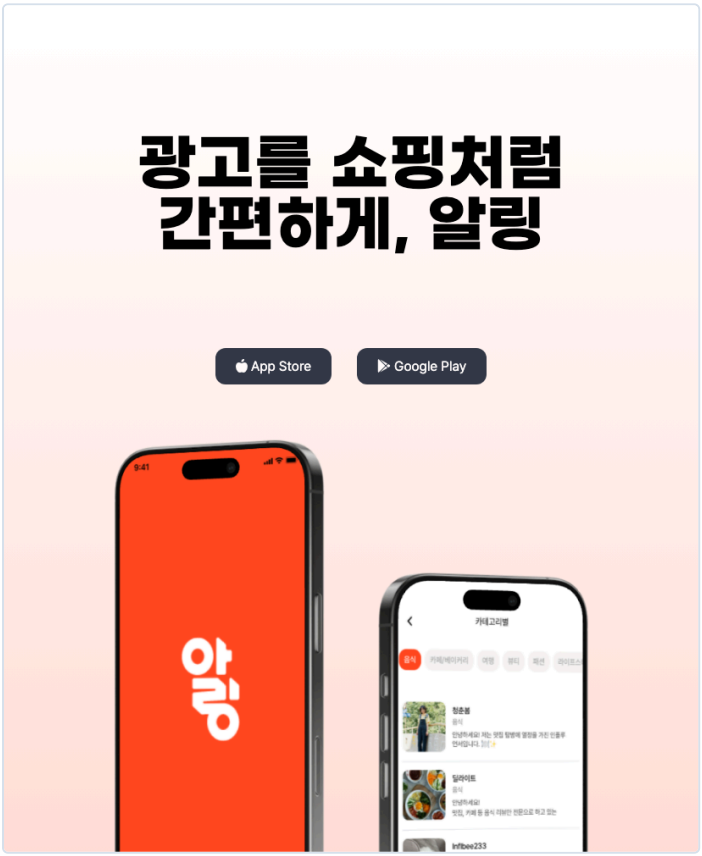
BUILT PROJECTS

Prototypes and services I built

Web services built to quickly validate product hypotheses or demonstrate real user flows.


Your Own
Tours
Arch
CA EDITION
리 사파리, 초원의 숨결을 따라가는 첫 시즌
드라이브, 럭셔리 텐트 로지, 응고롱고로 분화구까지. 실제 판매 패키지 기준의 5박 6일
화면에서 바로 읽을 수 있게 정리했습니다.


Challenge
Year 2026,
ust reading
ooks
?
satisfy your 'language thirst'!
studying! It's an "experience".
ge Now!
Try Free Demo
15분/일
dline
59
55
Min
Sec


광고를 쇼핑처럼
간편하게, 알링
App Store
Google Play
알링
카테고리별

PRODUCT SCREENS

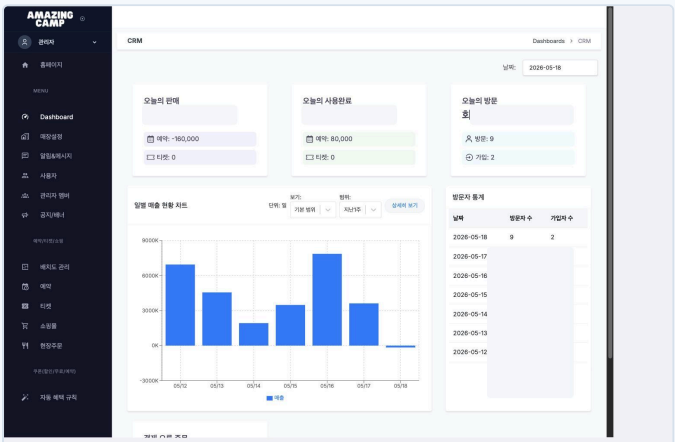
Product scope through real service screens

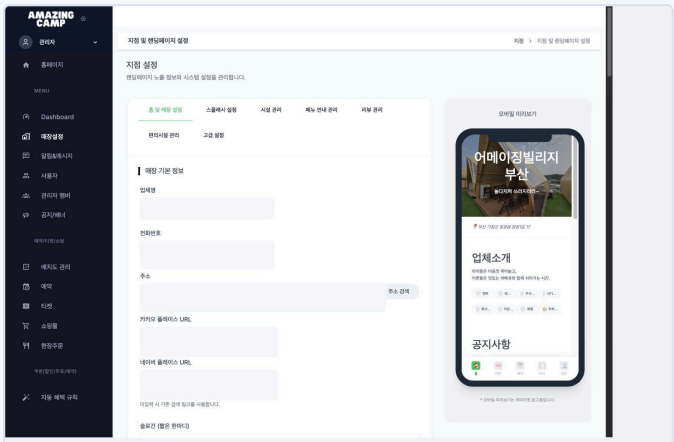
Customer reservation, CRM operations, and field POS screens are connected within the same product flow. Screenshots are based on staging screens, with sensitive account and reservation information masked.

CRM operations board: dashboard and store settings

CRM Board 1

Operators can review operating status and manage branch landing/store information in the same back office.



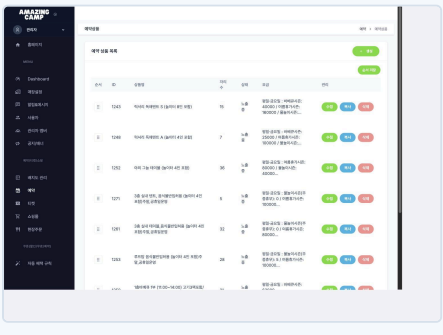


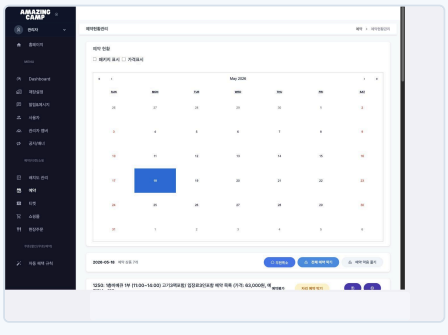
Product value Sales, visits, reservation metrics, and store landing settings stay connected so operators can quickly check daily operating status.

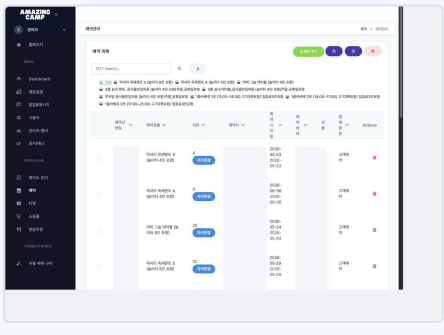
CRM reservation board: products, calendar, reservation list

CRM Board 2

Reservation products, daily status, and reservation handling lists are connected as one operating flow.





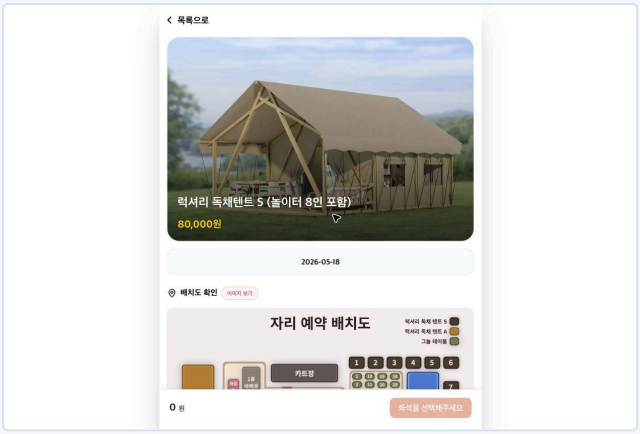


Product value Customer-selected products and operator-managed reservation states move on the same API contract.

Customer web: reservation detail

Web

Date, seat, details, and reservation CTA are connected in one conversion-oriented screen.

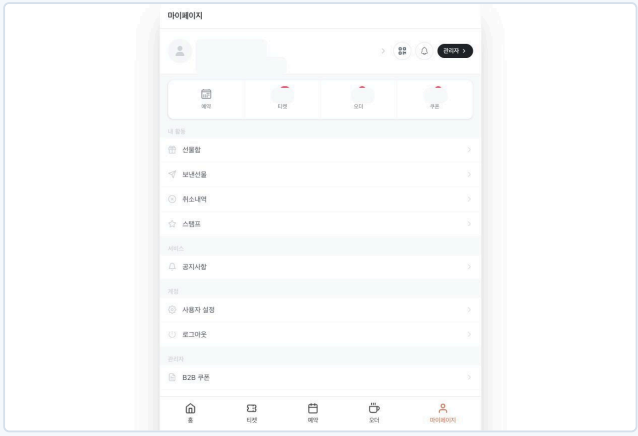


Product value Customer selection UI and back-office product/seat rules share the same reservation standard.

Customer web: my page

Web

A self-service screen where customers can review reservation, ticket, order, and coupon status.

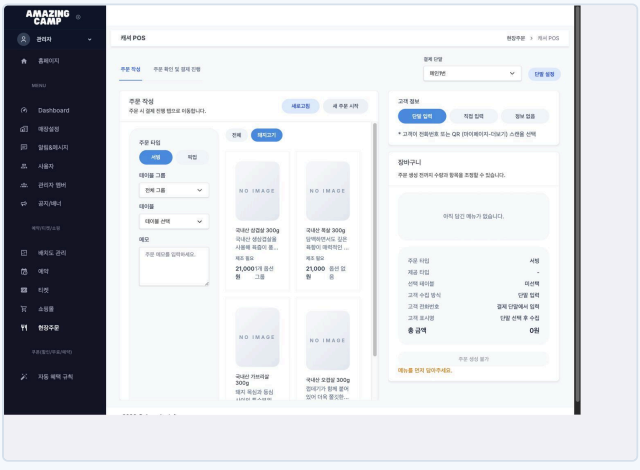


Product value Post-payment state and customer-owned assets are summarized in product language customers can understand.

POS: cashier sales screen

POS

Table/pickup selection, menu selection, cart, and terminal payment request are connected in one flow.

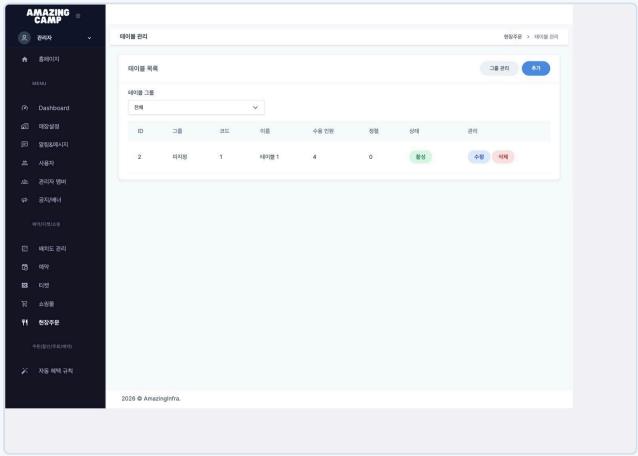


Product value The sales flow is separated by task units so field staff can operate quickly.

POS: table management

POS

Field tables and order context are managed so online reservations and on-site sales share the same operating model.



Product value The POS experience captures store operations beyond a simple payment screen.

PRODUCT CAPABILITIES

Core product capabilities

Feature boundaries and data contracts are aligned so customers, operators, and field staff can work from the same service state.

Reservation/order/payment state flow

The backend contract keeps customer web, CRM, and POS aligned around the same order state.

Tech base Django/DRF API, state transition rules, payment-provider response normalization

Product effect Payment success, failure, and exception handling connect to customer flow and operational response

CRM operations workflow

Dense screens support repeated operator work such as reservation changes, payment errors, and statistics checks.

Tech base React, Vite, server pagination, bulk actions, CSV export

Product effect Task-oriented operations UI that reduces repeated handling cost

Site-scoped permission model

Operational permissions are interpreted by site role and action-level policy, not only global admin status.

Tech base Django permission gate, site-scoped permission helper, targeted tests

Product effect A permission contract that balances security and operational convenience

On-site POS sales flow

Field staff need fast product selection, pending-payment locks, and recoverable order state.

Tech base Expo Router, React Native, React Query, local device context

Product effect Field touch flow and payment stability stay inside the same UX

Customer my page and purchase history

Customers need accurate site-scoped purchase history, reservation information, coupon, and ticket status.

Tech base React customer web, summary endpoint, shared modal contract

Product effect Front-end screens and backend aggregation rules stay aligned

Operating cost and media management

Operating cost is not outside the product. It is part of keeping service quality sustainable.

Tech base GCP Storage/CDN checks, signed URL separation, dry-run based processing scripts

Product effect Operational standards that consider cost, security, and visibility together

ARCHITECTURE

Service architecture

The center is the Django API contract. Each client has a different purpose, but they share the same data state and operating rules.

Customer Web

React, TypeScript, Vite, TossPayments SDK

CRM

React, server pagination, bulk operations, exports

Expo POS

React Native, Expo Router, terminal/device context

Django / DRF API

Order, Reservation, Payment, Permission, Product, Site Scope

PostgreSQL / PostGIS

Core transactional data and geo-enabled product/site data

Payment Provider

Payment approval, cancel, manual refund support path

GCP Operations

Cloud SQL, Storage, deployment scripts, billing/export checks

Customer, operator, and cashier experiences are separated at the UI layer but converge on the same backend contract and operational evidence.

CASE STUDIES

Representative project deep dives

These cases show product-minded development from problem definition to contract design, verification, and operational rollout.

CASE 01

Reservation operations flow

CRM, reservation product, calendar

Problem

When customer-facing reservation products and operator reservation lists/calendars drift, field operations become unstable.

Constraint

Product exposure, seat/date inventory, reservation status, cancellation, and entry handling happen across different screens.

Approach

Reservation products, calendar, and reservation lists are connected through the same Django API contract and site scope.

Effect

A CRM structure where product management, daily operations, and individual reservation handling stay in one flow.

CASE 02

Connecting store settings to customer screens

Site settings, customer web, product detail

Problem

Store information edited by operators must be reflected consistently in customer landing and reservation detail screens.

Constraint

Brand information, product description, dates, seats, and CTA can appear separated between operator and customer surfaces.

Approach

Site settings, product detail, seat selection, and my-page summary are connected through the same site-aware data flow.

Effect

A product structure where CRM setting changes carry through to customer experience.

CASE 03

Standalone POS field workflow

Expo POS, cashier workflow

Problem

Field sales need fast selection, payment locks, and recovery after failure more than screen switching.

Constraint

Terminal settings, site selection, order state, and pending-payment state must not drift during use.

Approach

Login-time site context, device-local settings, and cart/payment state machine are separated.

Effect

Field operation flow becomes simpler on tablet and web, with better state recovery potential.

TECHNICAL STACK

Technical stack and selection criteria

The stack is selected around product flow, operational stability, and verifiability.

Backend / API

- Django 5, Django REST Framework, Simple JWT
- PostgreSQL/PostGIS, Redis, Celery
- API documentation with drf-spectacular and drf-yasg
- Payment helper, permission gate, site-scoped business rules

Frontend / CRM / Web

- React 19, TypeScript, Vite, pnpm
- TanStack Query, Zustand, React Router
- Radix UI, Headless UI, Tailwind CSS, Bootstrap migration surface
- Server pagination, modal contract, toast/dialog UX, export flows

Mobile / POS

- Expo 55, React Native 0.83, Expo Router
- React Query, Secure Store, SVG/QR flows
- Device-local settings, terminal identity, payment state locks
- Android/Web shared cashier interaction model

Infra / Verification

- GCP, Cloud SQL, Cloud Storage, deployment scripts
- Targeted Django tests, Vitest, TypeScript checks
- Billing export and resource state validation
- Progress docs and task snapshots for operational continuity

UI / UX

UX principles by user type

Customers, operators, and field staff use the same product, but their speed requirements and error tolerance differ.

Customer Web

- Keep purchase and reservation flows short and clear
- Show pre/post-payment state and cancellation/refund information in product language
- Keep site-scoped summary and purchase history aligned in my page

CRM

- Dense tables and bulk actions for repeated operations work
- Express permission failures, payment exceptions, and statistics rules in operator-friendly states
- Connect filters, exports, and status changes by task unit

POS

- Large touch targets and fast product selection for field use
- Prevent state damage with movement/edit locks during pending payment
- Handle failed-order recovery and terminal settings inside the work flow

WORKING STYLE

Working style

Reproduction, contract checks, small verification, and documentation sync are treated as part of development.

> **"I verify the failure path and contract first, then improve product quality in the smallest safe unit."**

- Read the codebase and existing contracts first to narrow the change scope.
- Align shared state meaning across backend, customer web, CRM, and POS.
- Trace real callers, endpoints, and permission gates for high-risk payment, permission, and reservation flows.
- Run targeted tests and manual verification based on change risk.
- Record progress and follow-up work so the next operator can continue safely.

Kihwan Lee
Product-minded Full-stack Developer
kifhan@gmail.com
https://github.com/kifhan