

About Me



ISLOMBEK DAVID

- FullStack Developer
- DevOps Engineer
- Agentic Developer
- AI Engineer

about me



Full-Stack & AI Engineer based in Seoul, South Korea, with 2+ years building scalable web platforms and production-grade backend systems. I work across TypeScript and Python, with React, Next.js, and NestJS on the stack side and GraphQL and MongoDB underneath and I run my own live products end to end, from database design to Docker deployments with SSL and reverse-proxy. I care most about turning messy, real-world problems into clean, reliable software, often powered by LLM features and automation that removes hours of manual work. Active in Seoul's tech community through meetups like Dev Korea, I speak Uzbek, Korean, English, and Russian, which lets me build and collaborate across very different audiences.

My education & training journey



Kookmin University

- Bachelor's degree, Artificial Intelligence
- Mar 2025 – Present



Kookmin University

- Korean Language Center, Korean Language
- Mar 2024 – Mar 2025



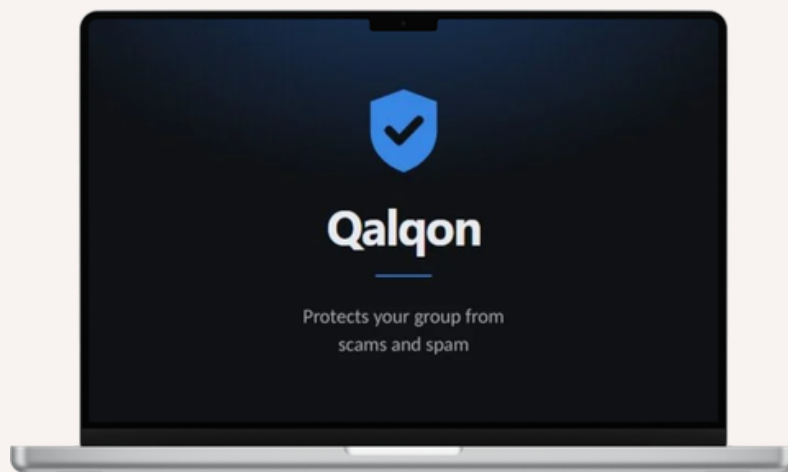
Daejin university

- Bachelor's degree, Business Administration and Management
- Sep 2023 – Feb 2024

projects

✳️ **Qalqon (AI) bot**

<https://qalqon-safe.uz>



Qalqon is an AI moderation bot that protects Telegram communities from scammers. It reads every message with a safety-tuned language model, checks links, files and the sender's profile, and removes clear attacks in under a second in any language while a lone red flag goes to a human, so real members are never banned by mistake.

✳️ **K-Apply**

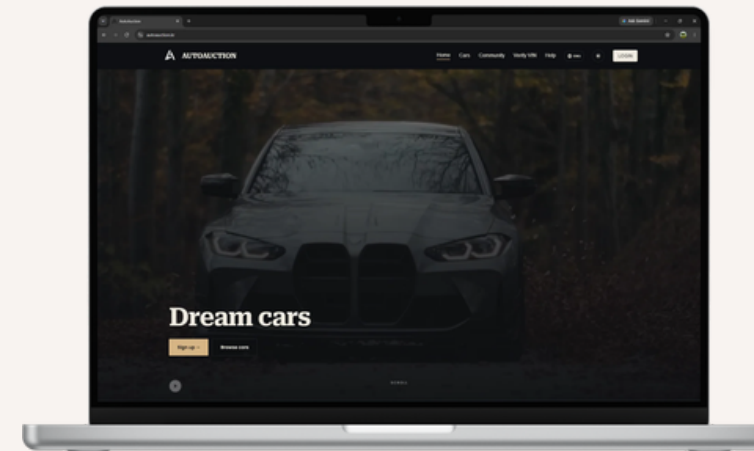
<https://k-apply.uz>



K-Apply is a production platform that helps students from Central Asia apply to study in Korea. Students browse universities, scholarships, and language courses with shareable URL-synced filters, complete a guided six-step profile, and apply while admins review every application from a dedicated panel. Ships in four languages with Google and Telegram sign-in.

✳️ **Auto Auction**

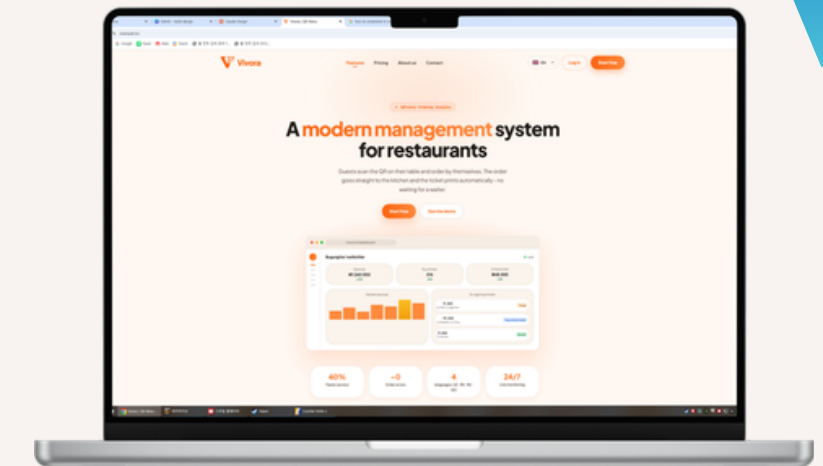
<https://autoauction.kr>



A live marketplace for buying Korean cars at fixed prices, aimed at buyers from Central Asia. Customers browse a detailed car catalog in four languages with prices in KRW and USD from live exchange rates, request a consultation directly from any listing, and earn loyalty points while admins manage inventory, sales, and leads from a dedicated panel. React + Redux Toolkit storefront, Express + MongoDB backend.

✳️ **Vivora**

<https://vivora.kr>



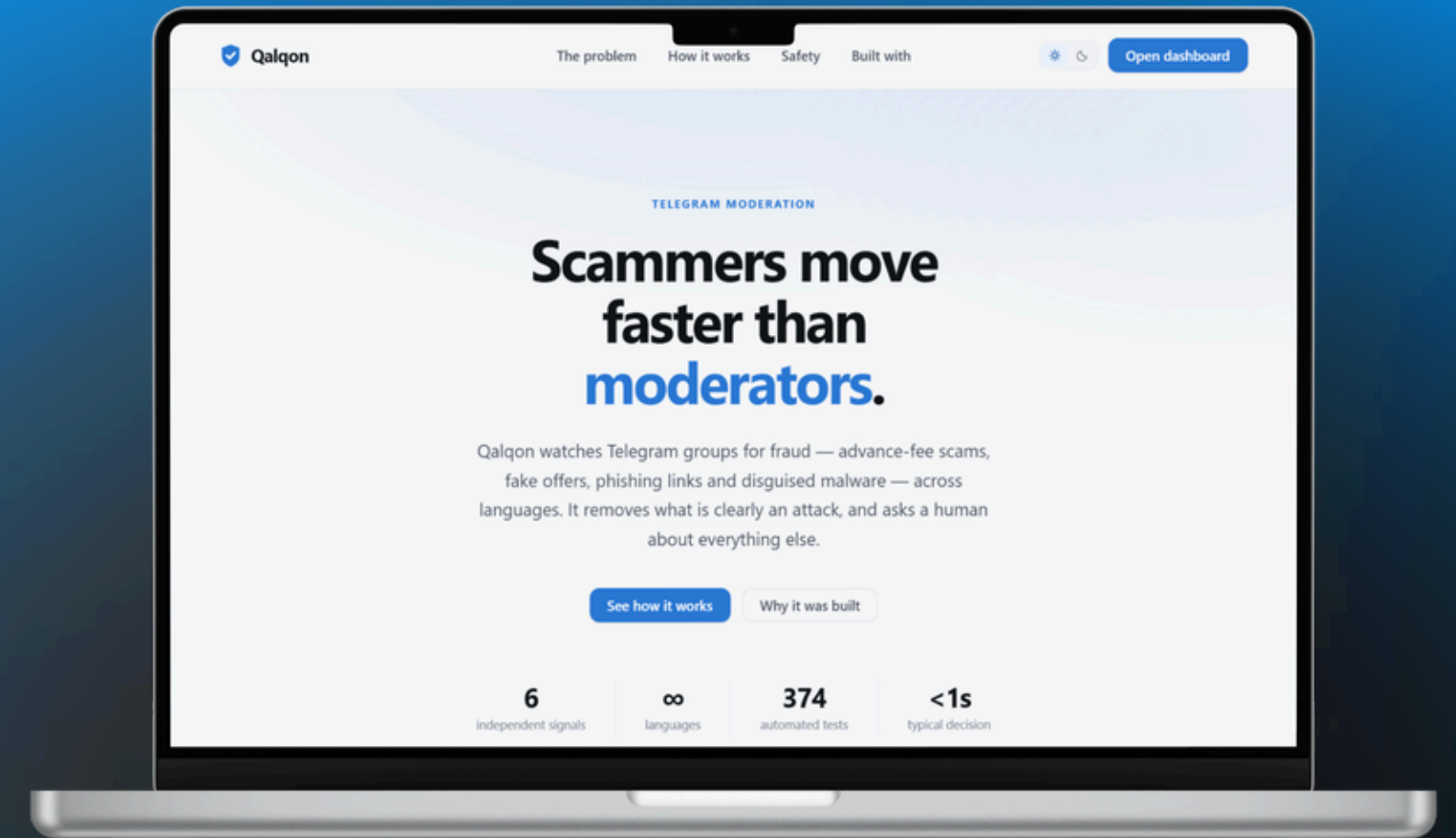
This project is a real-time application that protects other people's privacy when someone films video in public. It finds every face in the camera stream, lets the user pick one main face (by clicking it), and automatically blurs all the other faces. If the main face is lost, it blurs everyone for safety so no stranger is ever shown by accident.

Description

Qalqon Telegram Anti-Scam Moderation Bot

A production AI moderation bot that protects Telegram groups from fraud advance-fee scams, phishing links and disguised malware in any language. Every message passes through six signals: keyword filter, link and file analysis, a safety-tuned language model that judges intent rather than topic, and checks on the sender's bio, channel and photo. A pure, fully-tested policy turns them into Review, Delete or Ban. Clear attacks are removed in under a second; anything ambiguous goes to admins with one-tap Ban / Ignore / Whitelist buttons. Strikes expire, and long-standing members are never auto-banned.

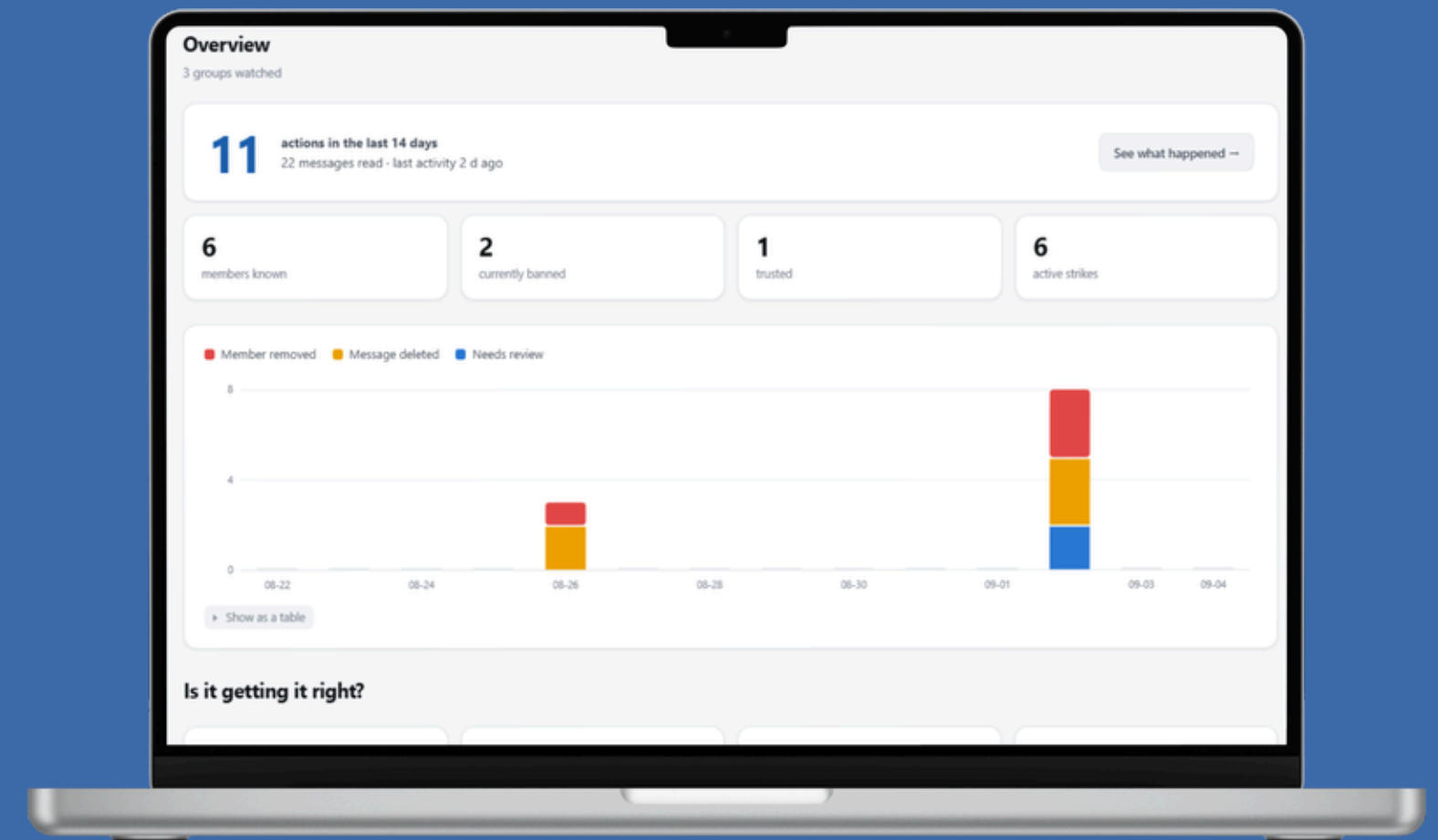
<https://qalqon-safe.uz>



Designed, built, and deployed solo: Python bot on python-telegram-bot with an event-driven MVC core; Groq gpt-oss-safeguard-20b for text and a Hugging Face vision classifier for profile photos; SQLite memory for strikes, trust and events; a read-only FastAPI dashboard with Telegram Login; Dockerized behind nginx + Let's Encrypt; 250+ offline pytest tests. Live at t.me/QalqonSafeBot

Key Features

- AI Intent Detection (Groq · gpt-oss-safeguard-20b)
- Six Independent Signals (Keyword, Link, File, LLM, Profile, Vision)
- Structural Link Analysis (Typosquats, Homographs, Credentials-in-URL)
- Dangerous File Detection (.apk / .exe on any attachment)
- Profile-Photo Screening (Hugging Face Vision Model)
- Deep Channel Scan via MTProto (Telethon)
- Pure Escalation Policy (Content × Profile + Strikes + Trust)
- Two Signals to Act — Message & Profile Must Agree
- 30-Day Strike Decay & Trusted-Member Cap
- One-Tap Admin Alerts (Ban / Ignore / Whitelist) with Detector Breakdown
- Admin Commands (/stats /status /whitelist /forgive /unban /dryrun)
- Three Autonomy Levels & 6-Hour Digests
- Flood & Raid Defence with Alert Coalescing
- Fails Loud: Degraded Verdicts, Heartbeat, Dry-Run by Default
- Read-Only Web Dashboard with Telegram Login & Live LLM Cost
- Works in Any Language · 90-Day Retention

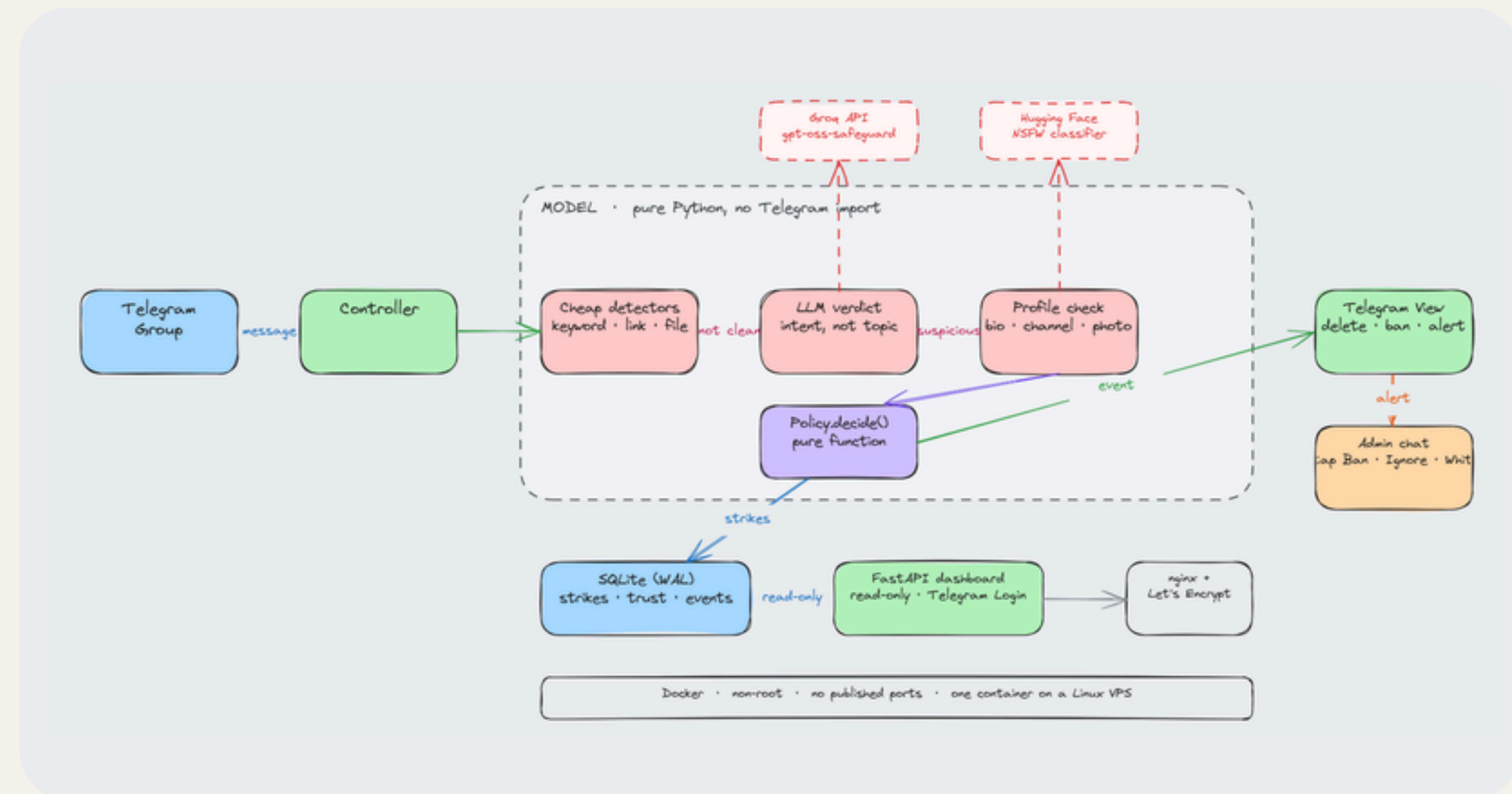


<https://qalqon-safe.uz>

System Architecture

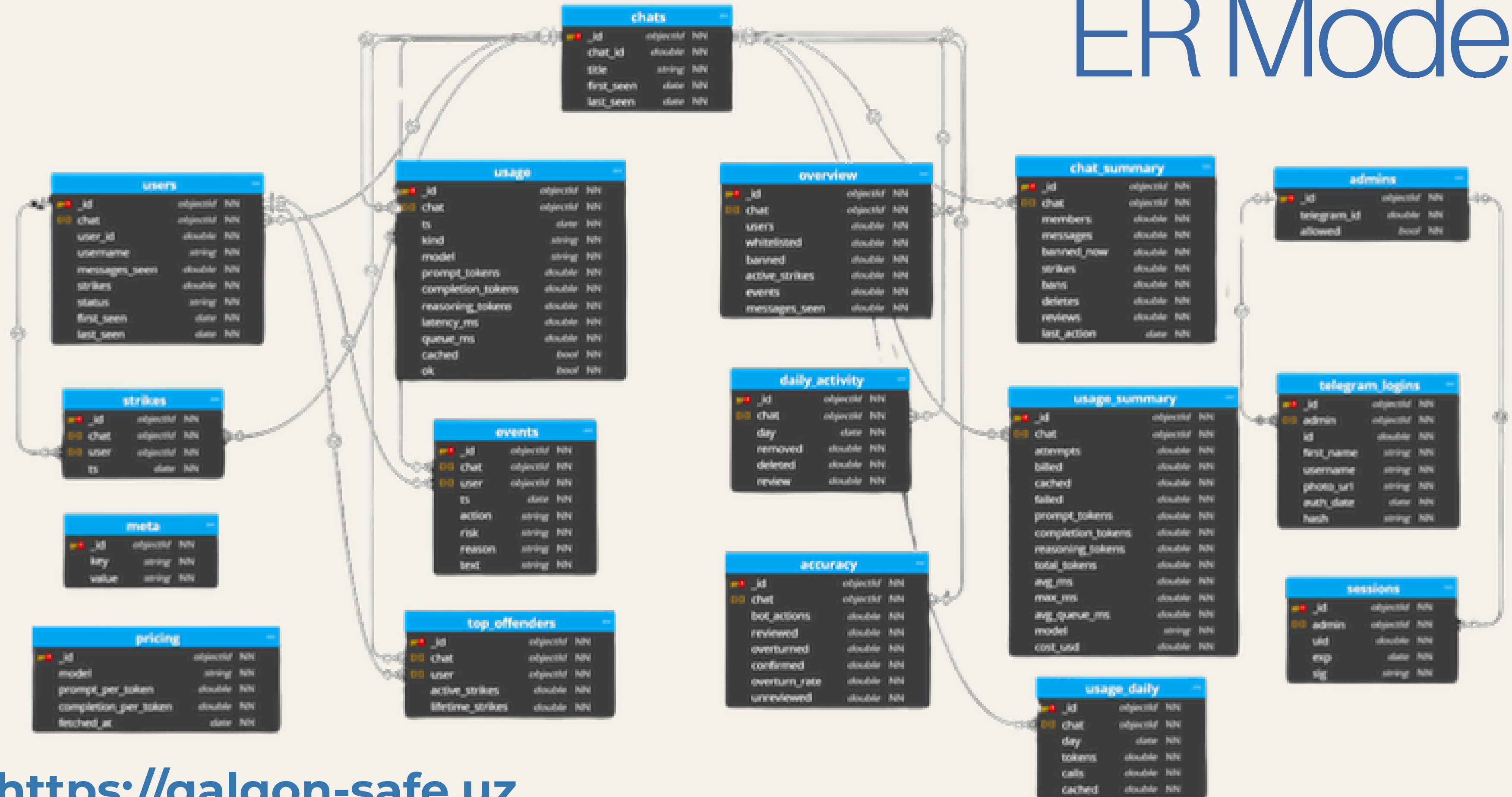
Event-driven Model-View-Controller. The Controller translates a Telegram update into a plain message object and decides nothing. The Model holds every rule: cheap detectors run first, the Groq model only when it can still change the outcome, and the sender's profile only when the message is already suspicious. `Policy.decide()` is a pure function with no I/O, and the Model announces a domain event.

Views subscribe: the TelegramView deletes, bans and posts alerts with inline buttons; the Digest and AlertBatcher coalesce routine actions; a Heartbeat proves the process is alive. State lives in SQLite (WAL). The FastAPI dashboard is a separate process that opens the database read-only, so a bug there cannot touch moderation. The bot publishes no ports and runs as one non-root Docker container, only the dashboard sits behind nginx + Let's Encrypt.



<https://qalqon-safe.uz>

ER Model



<https://qalqon-safe.uz>

Detection Pipeline & Escalation

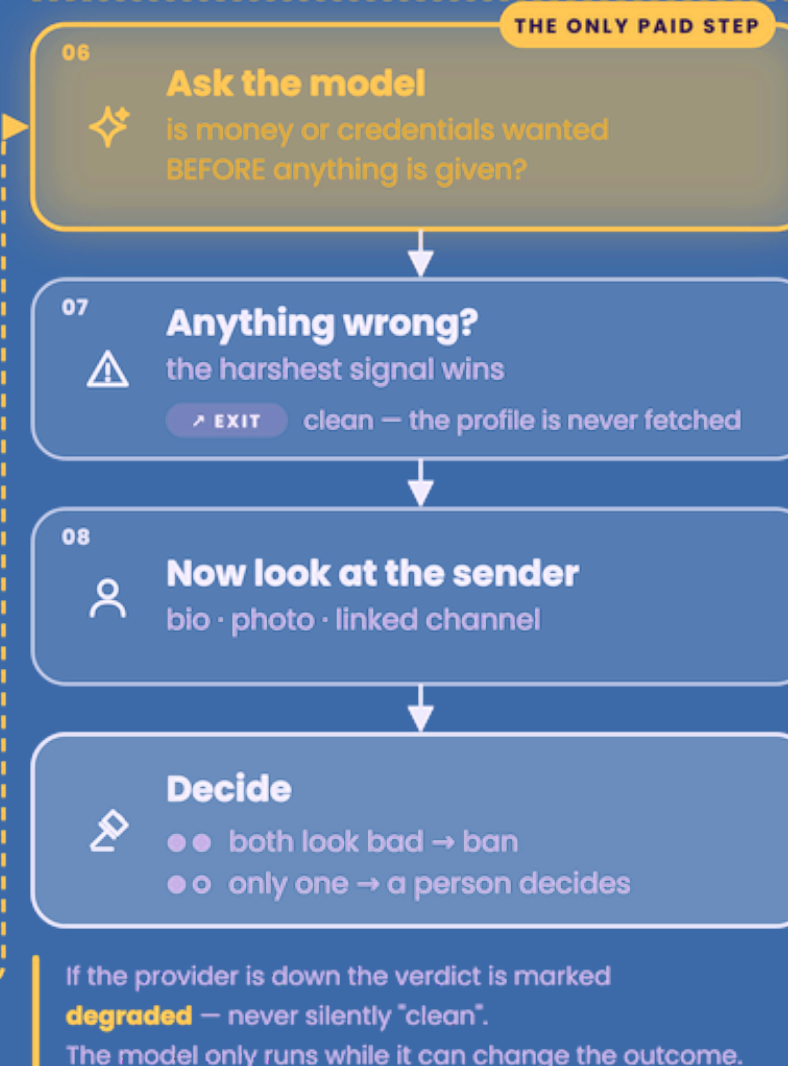
The topic is never the signal. Real communities talk about money constantly a generic ruleset flagged four of every fifteen legitimate posts. Qalqon asks the model one question instead: is someone being asked to part with money or credentials before receiving anything? That is why it works in a language it has never been tuned for. Cheap detectors run first, the model only when it can still change the outcome, and if a provider is down the verdict is marked degraded never silently "clean".

<https://qalqon-safe.uz>

FREE · RUNS ON EVERY MESSAGE



PAID · ONLY WHEN STILL UNDECIDED



Built to Be Wrong Safely

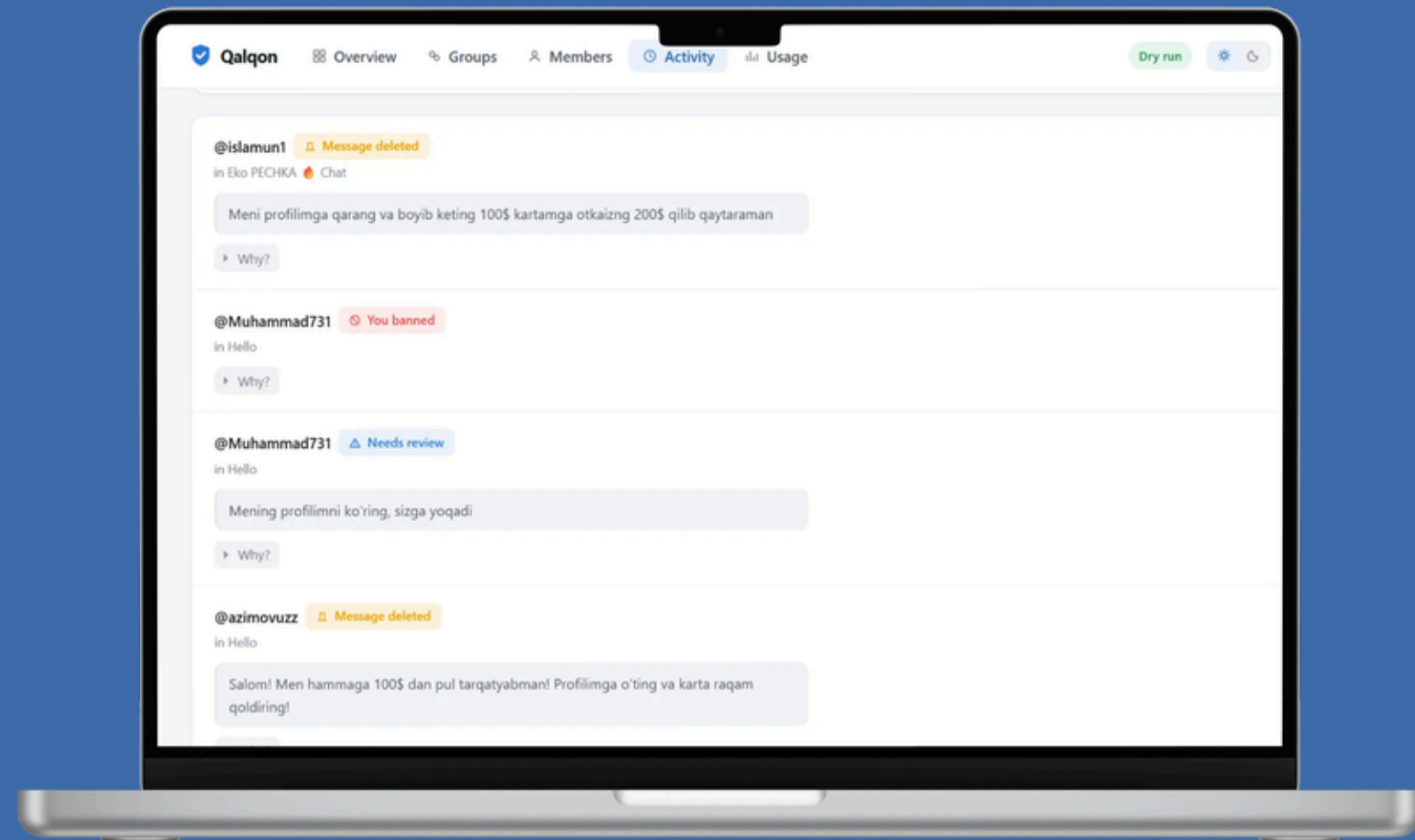
A moderation bot's worst failure is not missing a scam it is removing a real member invisibly. So a suspicious message alone is never enough: the sender's profile has to agree, and a lone red flag goes to a human. New deployments run in Dry Run and report what they would do before they are trusted.

Every alert carries Ban, Ignore and Whitelist buttons, and marking a decision wrong clears the strike behind it. Strikes expire after 30 days, established members are never auto-banned, and flood detection can delete but never ban. Ordinary conversation is never stored only acted-on messages, for 90 days.



<https://qalqon-safe.uz>

Admin Panel



<https://qalqon-safe.uz>

A read-only FastAPI dashboard behind Telegram Login (HMAC-verified) and an admin allow-list. Server-rendered, zero JavaScript, no CDN.

- Overview – actions today, active groups, review queue, activity chart
- Groups – every moderated chat with Review / Delete / Ban counts
- Members – strikes (active / lifetime), trust, whitelist, bans
- Activity – who was flagged, by which detector, what happened
- Usage – LLM tokens, latency, cache hits, live cost per model
- Privacy – what is stored and for how long
- In Telegram – /stats /status /whitelist /forgive /unban /dryrun + alert buttons

Used Technologies (Qalqon)

Language and runtime



Python 3.12 · asyncio



python-telegram-bot 22

Backend / Bot



Event-driven MVC (models · views · controllers)



Telethon (MTPROTO deep channel scan)



SQLite (WAL) via stdlib · versioned migrations



httpx · python-dotenv

AI / ML



Groq API · openai/gpt-oss-safeguard-20b (text intent)



Hugging Face Inference · Falconsai/nsfw_image_detection (profile photos)

Custom detector pipeline → pure policy function

Web dashboard



FastAPI + uvicorn



Server-rendered HTML, zero JS, strict CSP



Telegram Login Widget (HMAC-SHA256) · signed stateless session cookies

Infrastructure



Docker Compose



nginx + certbot (HTTPS)



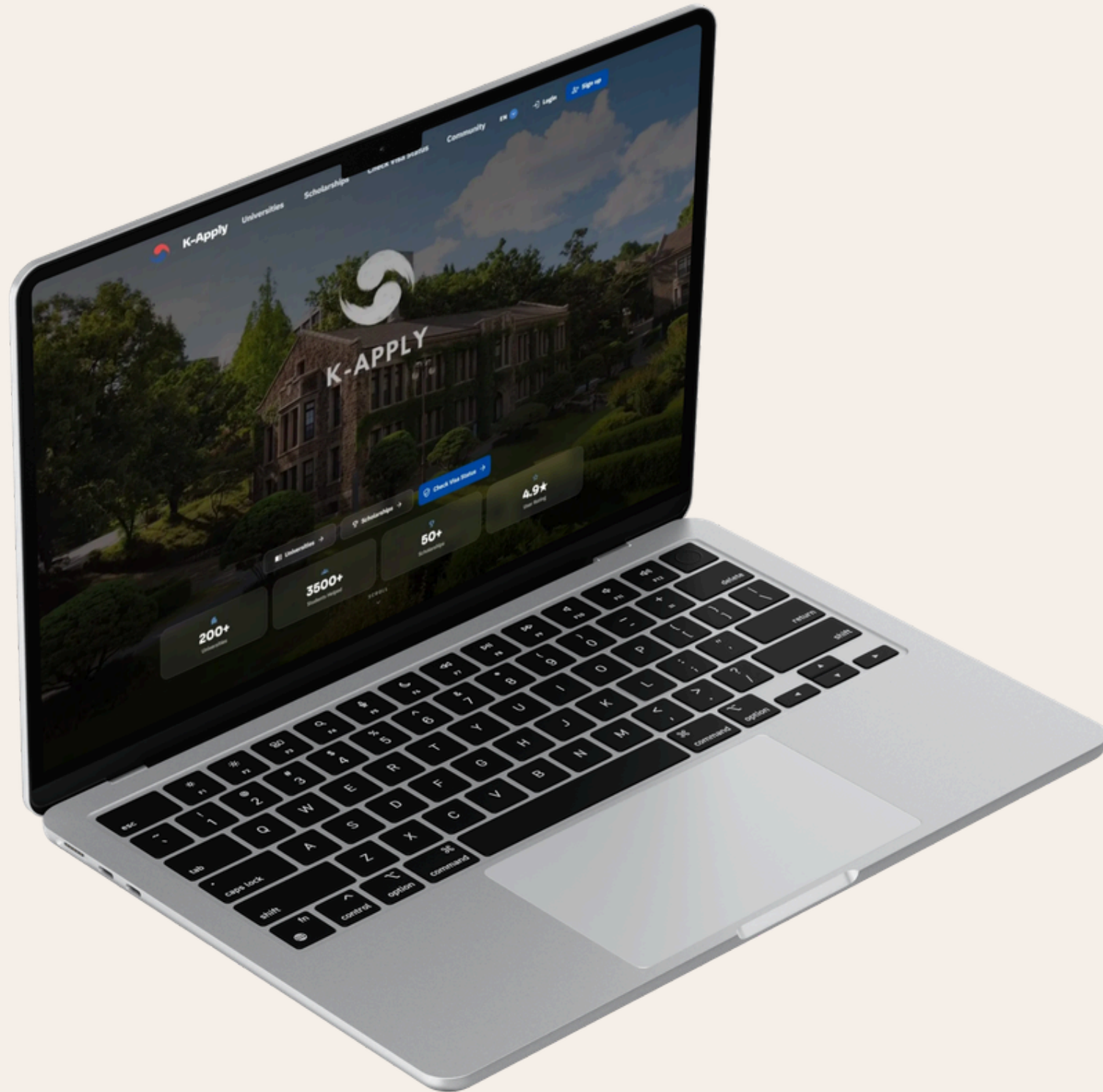
Linux VPS



GitHub deploy key workflow

<https://qalqon-safe.uz>

DESCRIPTION **K-APPLY** STUDY-IN-KOREA APPLICATION PLATFORM



A production full-stack platform that helps international students discover Korean universities, scholarships, and language courses and apply through a guided workflow. Students complete a 7-step profile wizard (education history, TOPIK/IELTS certificates, essays) and track their applications, while admins review submissions from a dedicated dashboard. The entire product ships in 4 languages with Google & Telegram sign-in, an integrated visa-status checker, and product analytics.

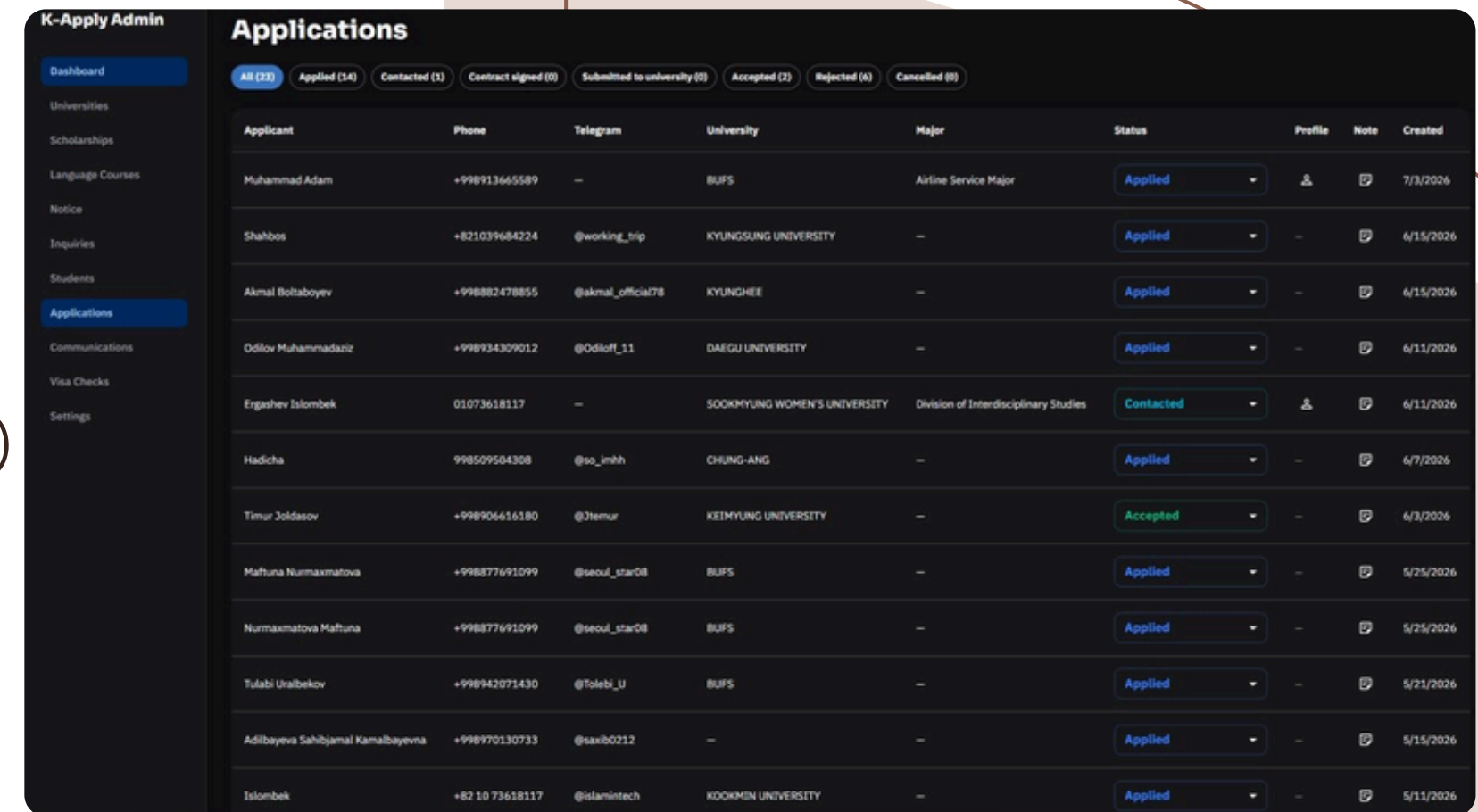
Designed, built, and deployed solo: NestJS GraphQL API with MongoDB, role-based auth and rate limiting; Next.js + MUI frontend with Apollo Client; Dockerized deployment behind nginx live at k-apply.uz (<https://www.k-apply.uz>).

[HTTPS://K-APPLY.UZ](https://k-apply.uz)

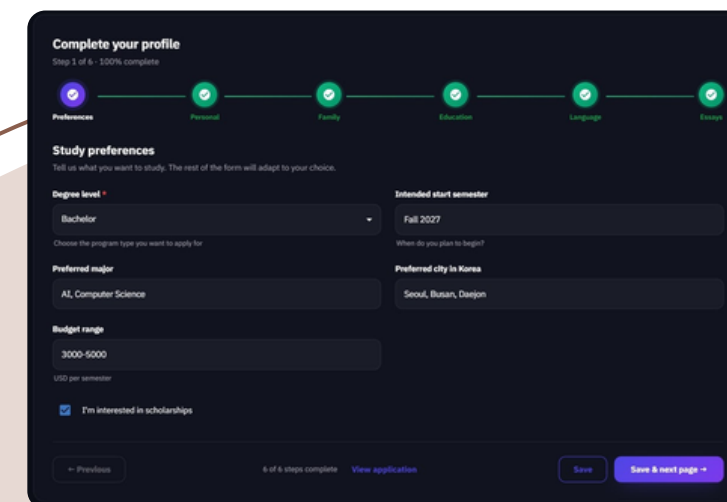
KEY FEATURES

- Multi-provider Authentication (Google, Telegram, JWT)
- 6-Step Student Profile Wizard (Education, Certificates, Essays)
- University, Scholarship & Language Course Catalog
- Dynamic Search, Filters & Rankings (Top 1%, Korean Top 10)
- Application Submission & Status Tracking
- Admin Dashboard (Applicant Review, Content & User Management)
- Visa Status Checker (Standalone Microservice)
- Public Q&A Board & Inquiry System
- Email Verification & Secure Password Change Flow
- Media Uploads via Cloudinary (Images & Documents)
- Multilingual Support (English, Korean, Russian, Uzbek)
- Dark & Light Themes
- Visa Status Checker (Standalone Microservice)
- Public Q&A Board & Inquiry System
- Email Verification & Secure Password Change Flow
- Media Uploads via Cloudinary (Images & Documents)
- Multilingual Support (English, Korean, Russian, Uzbek)
- Dark & Light Themes
- Fully Responsive Design
- Product Analytics (PostHog) & API Rate Limiting

[HTTPS://K-APPLY.UZ](https://k-apply.uz)



K-Apply Admin									
Applications									
All (23) Applied (14) Contacted (1) Contract signed (0) Submitted to university (0) Accepted (2) Rejected (6) Cancelled (0)									
Applicant	Phone	Telegram	University	Major	Status	Profile	Note	Created	
Muhammad Adam	+998913665589	—	BUFS	Airline Service Major	Applied	👤	📝	7/3/2026	
Shahbos	+821039684224	@working_trip	KYUNGSUNG UNIVERSITY	—	Applied	—	📝	6/15/2026	
Akmal Boltaboyev	+998882478855	@akmal_official78	KYUNGHEE	—	Applied	—	📝	6/15/2026	
Odlov Muhammadaziz	+998934309012	@Odloff_11	DAEGU UNIVERSITY	—	Applied	—	📝	6/11/2026	
Ergashev Islombek	01073618117	—	SOOKMYUNG WOMEN'S UNIVERSITY	Division of Interdisciplinary Studies	Contacted	👤	📝	6/11/2026	
Hadicha	998509504308	@so_lmth	CHUNG-ANG	—	Applied	—	📝	6/7/2026	
Timur Joldasov	+998906616180	@Jtemur	KEIMYUNG UNIVERSITY	—	Accepted	—	📝	6/3/2026	
Maftuna Nurmaxmatova	+998877691099	@seoul_star08	BUFS	—	Applied	—	📝	5/25/2026	
Nurmaxmatova Maftuna	+998877691099	@seoul_star08	BUFS	—	Applied	—	📝	5/25/2026	
Tulabi Uraltbekov	+998942071430	@Tolabi_U	BUFS	—	Applied	—	📝	5/21/2026	
Adilbayeva Sahibjamal Kamalbayeva	+998970130733	@saxib0212	—	—	Applied	—	📝	5/15/2026	
Islombek	+82 10 73618117	@Islamintech	KOOKMIN UNIVERSITY	—	Applied	—	📝	5/11/2026	



Complete your profile
Step 1 of 6 - 100% complete

Preferences

Study preferences
Tell us what you want to study. The rest of the form will adapt to your choice.

Degree level
Bachelor

Intended start semester
Fall 2027

Preferred major
AI Computer Science

Budget range
3000-5000 USD per semester

I'm interested in scholarships

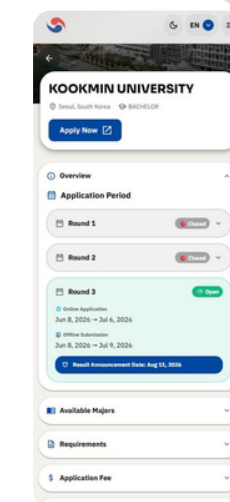
Previous

4 of 6 steps complete

View application

Save

Save & next page



K-Apply

KOOKMIN UNIVERSITY

Apply Now

Overview

Application Period

Round 1

Round 2

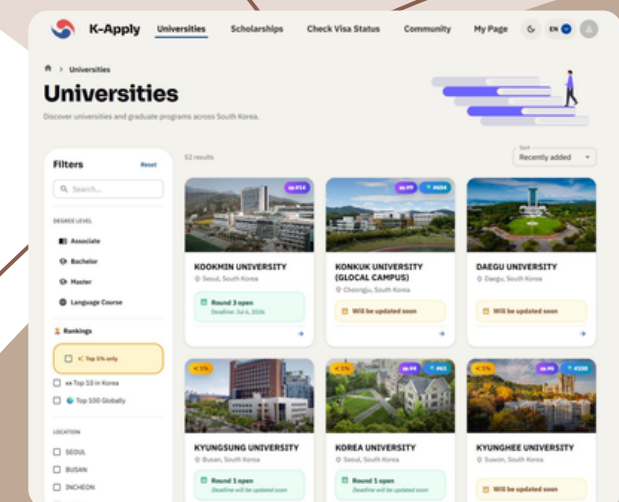
Round 3

Round Assessment Date: Aug 11, 2026

Available Majors

Requirements

Application Fee



K-Apply

Universities

Discover universities and graduate programs across South Korea.

Filters

Search

Round 1 open

Round 2 open

Round 3 open

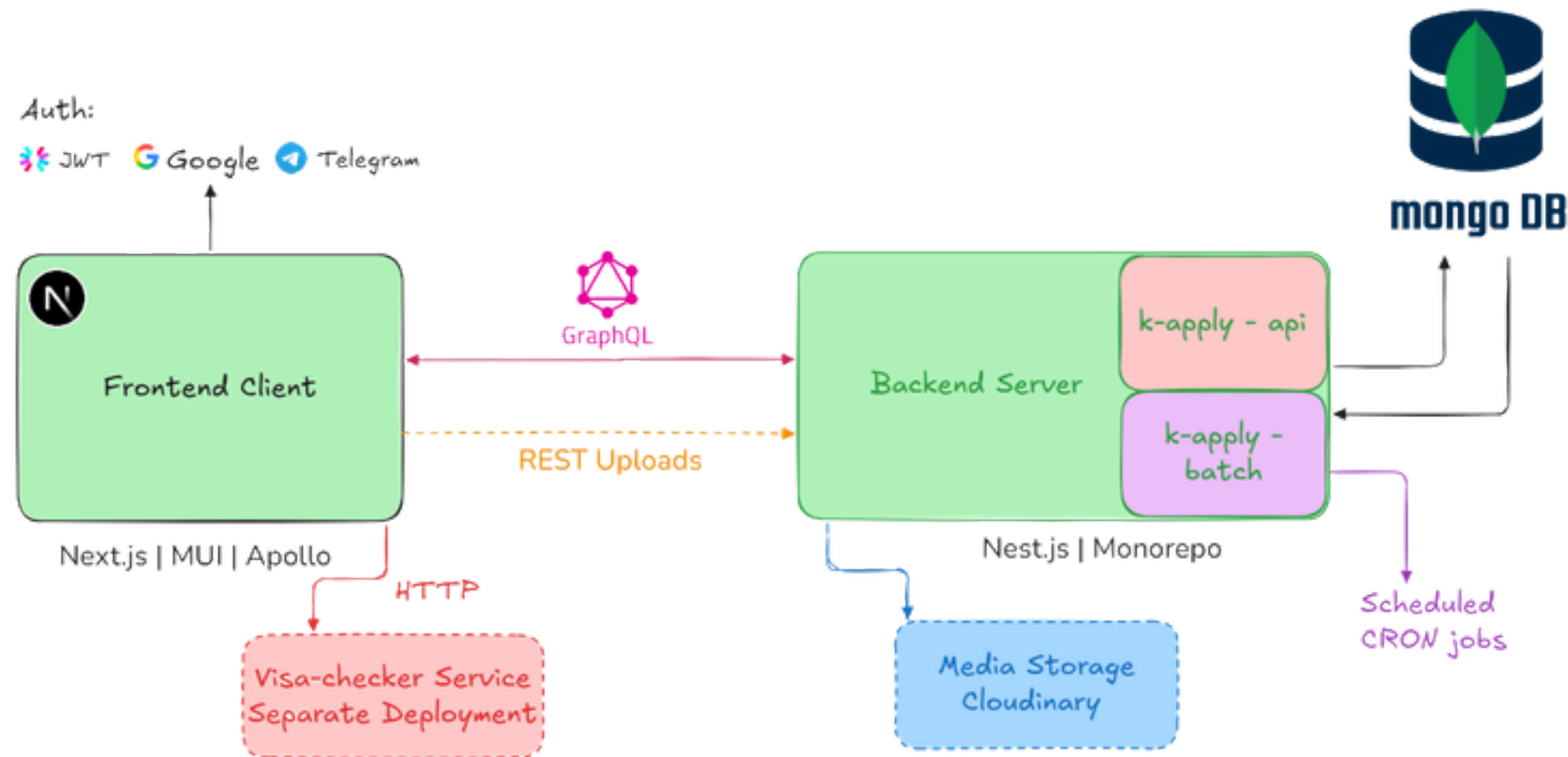
Round Assessment Date: Aug 11, 2026

Available Majors

Requirements

Application Fee

SYSTEM ARCHITECTURE

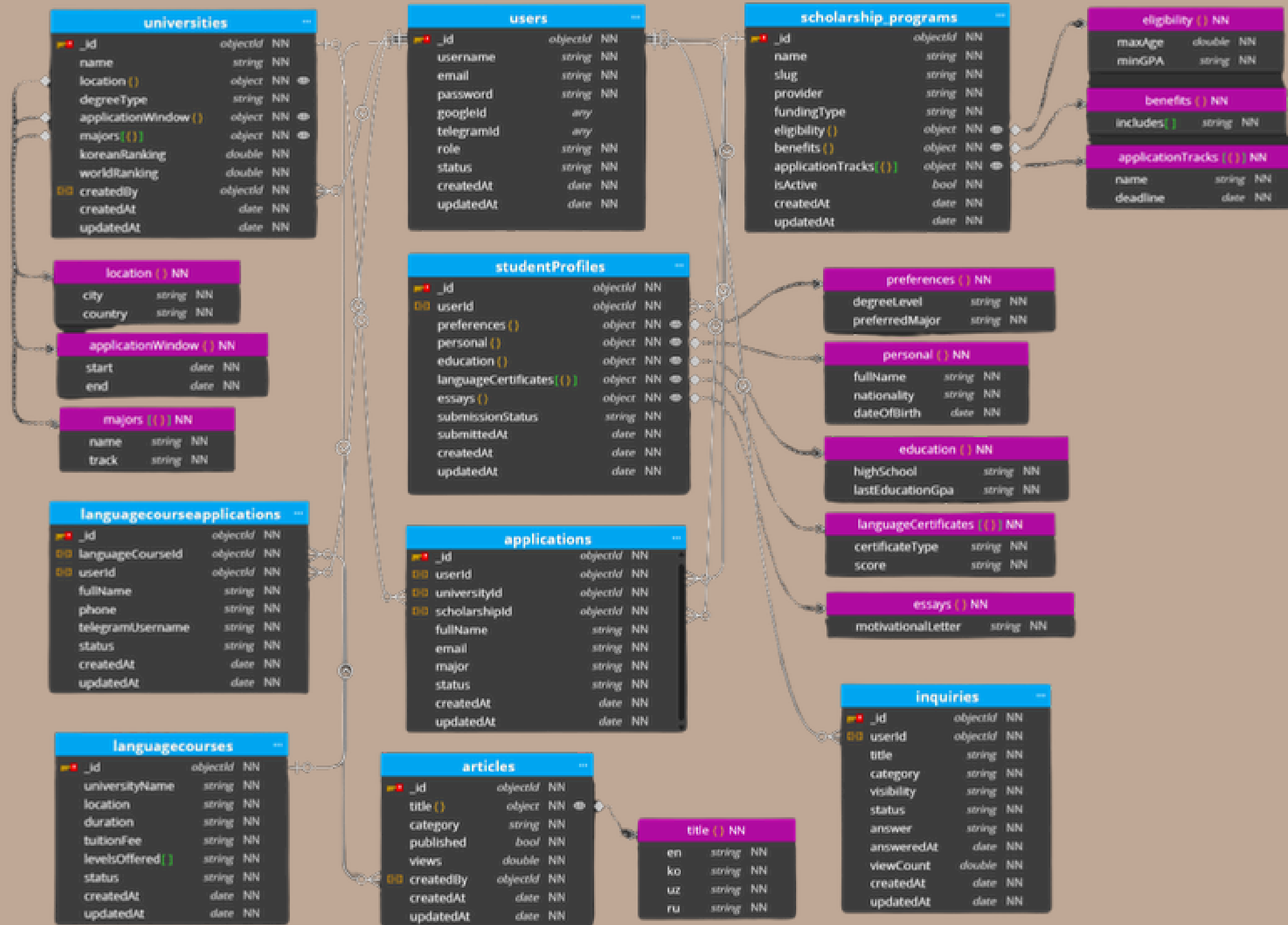


[HTTPS://K-APPLY.UZ](https://k-apply.uz)

Next.js handles the frontend, and Apollo Client talks to the backend over GraphQL while caching responses locally to avoid redundant requests. The backend is a NestJS monorepo a code-first GraphQL API (kapply-api) plus a batch app (kapply-batch) for scheduled cron jobs split into independent modules (auth, student profiles, universities, scholarships, applications) that each own their resolvers and schemas. Data lives in MongoDB through Mongoose.

ign-in supports credentials plus Google and Telegram OAuth on a JWT scheme, with role-based access control on every resolver. File uploads bypass GraphQL and go over REST into Cloudinary, while visa eligibility checks call a separately deployed visa-checker service over HTTP. The whole UI ships in four languages (EN / KR / RU / UZ), and in production everything runs in Docker Compose behind Nginx on a single Linux VPS.

K-APPLY ER MODEL



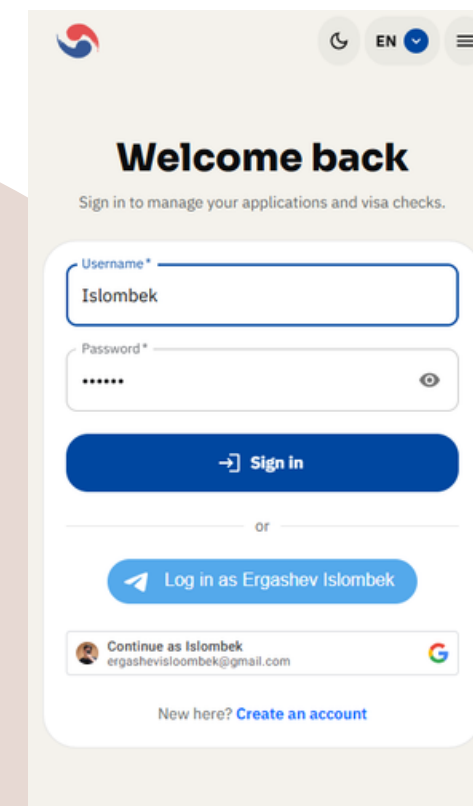
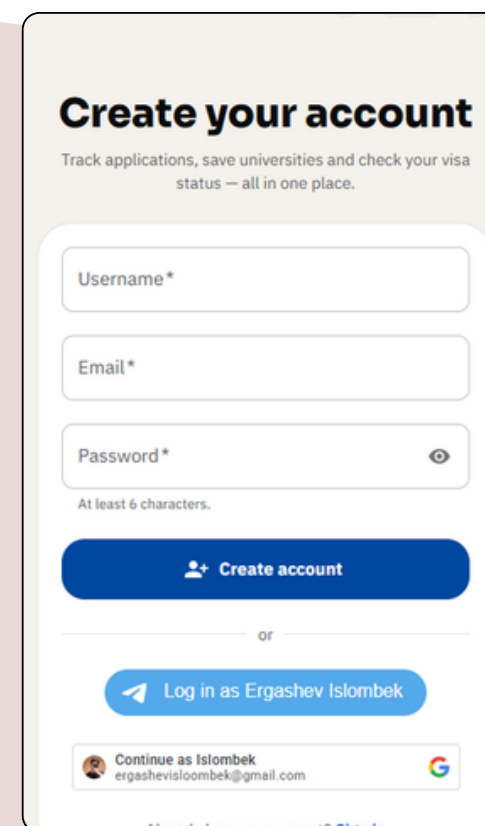
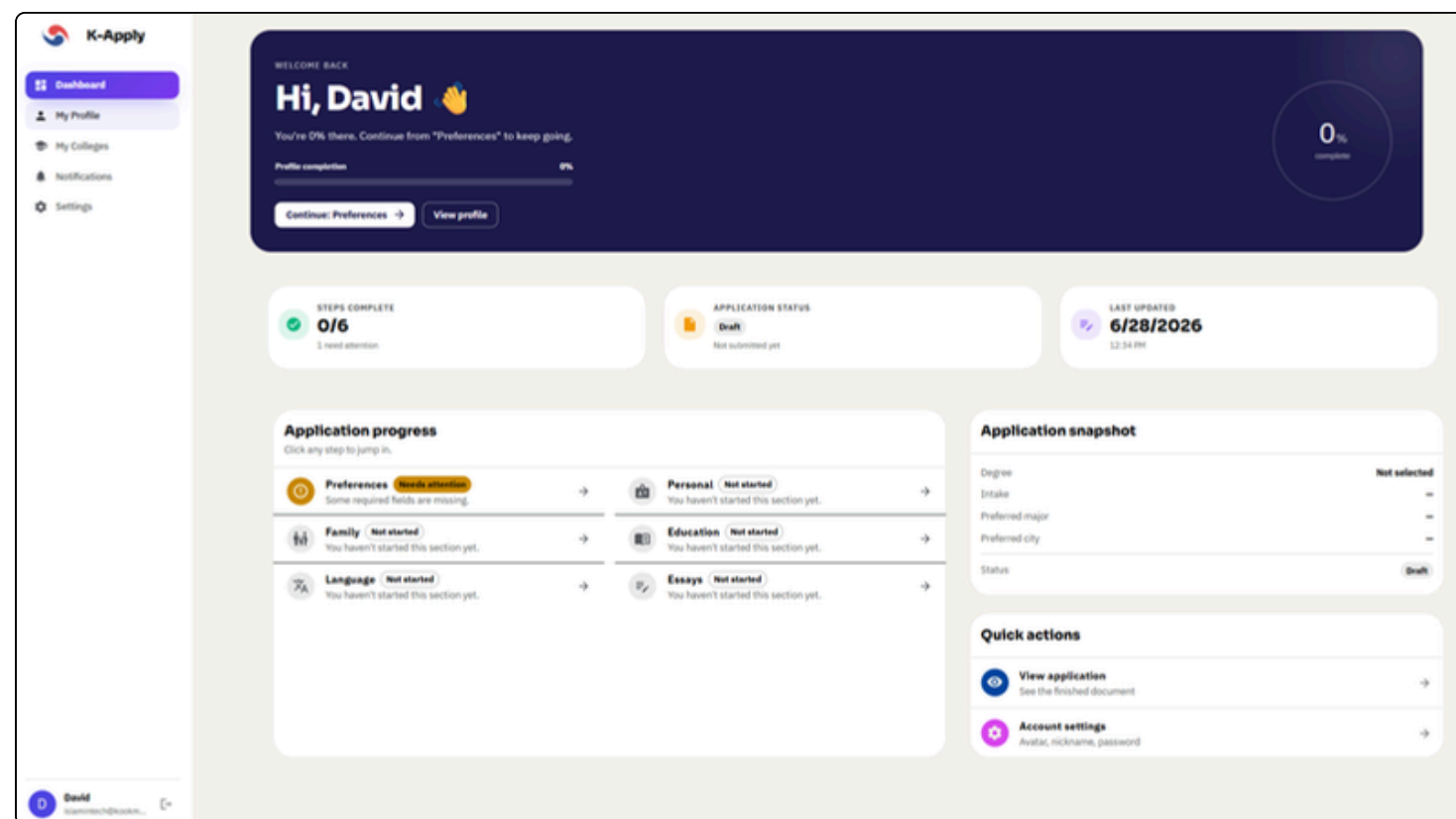
[HTTPS://K-APPLY.UZ](https://k-apply.uz)

AUTHENTICATION & PROFILE MANAGEMENT

Every user manages their own account profile details, avatar upload, password changes with email verification, and language preferences across four locales.

Administrators get a dedicated dashboard to review students, track applications, and control account status platform-wide.

Implemented multi-provider authentication: Google OAuth, Telegram Login with server-side HMAC signature verification, and JWT-based email/password sign-in. Role guards enforce strict separation between student and administrator access, backed by API rate limiting on all auth endpoints.

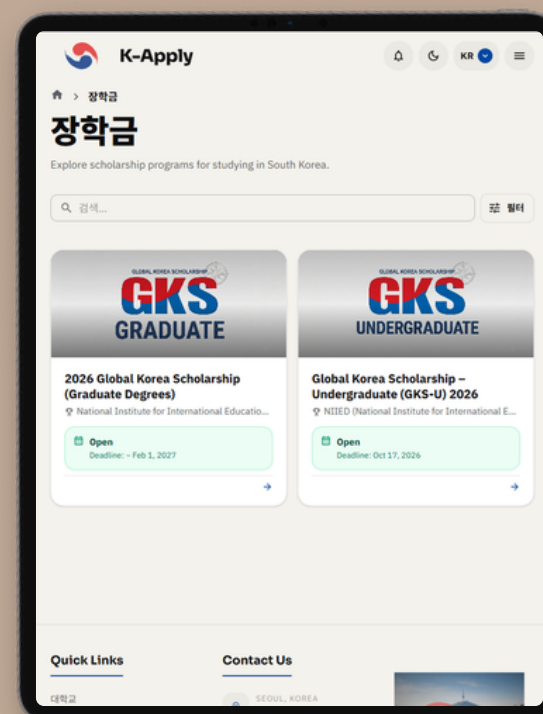
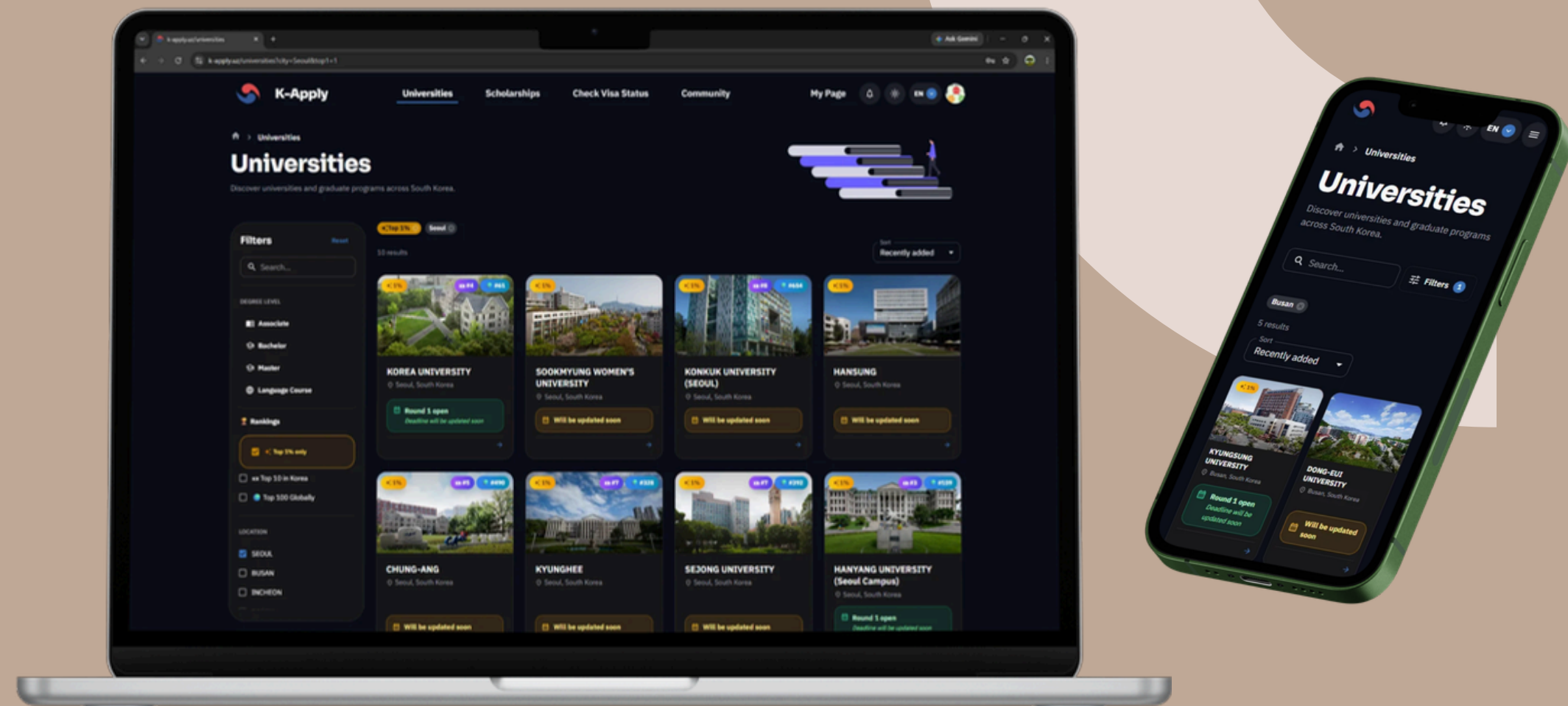


HTTPS://K-APPLY.UZ

DYNAMIC FILTERING & RESPONSIVE LAYOUT

Advanced catalog discovery lets students filter universities by degree level, city, and ranking badges (Top 1%, Korea Top 10, World Top 100), scholarships by funding type and open application periods, and language courses by level and location. Every filter is synchronized with the URL bookmark, share, or revisit any filtered view and the exact search state is reproduced instantly. The platform is built mobile-first on MUI's component system: on desktop, a sticky filter sidebar sits beside the results grid; on phones it becomes a bottom drawer with an always-visible search bar and a pinned "Show results" button. Adaptive card grids, dark and light themes, and four interface languages stay consistent across every screen size.

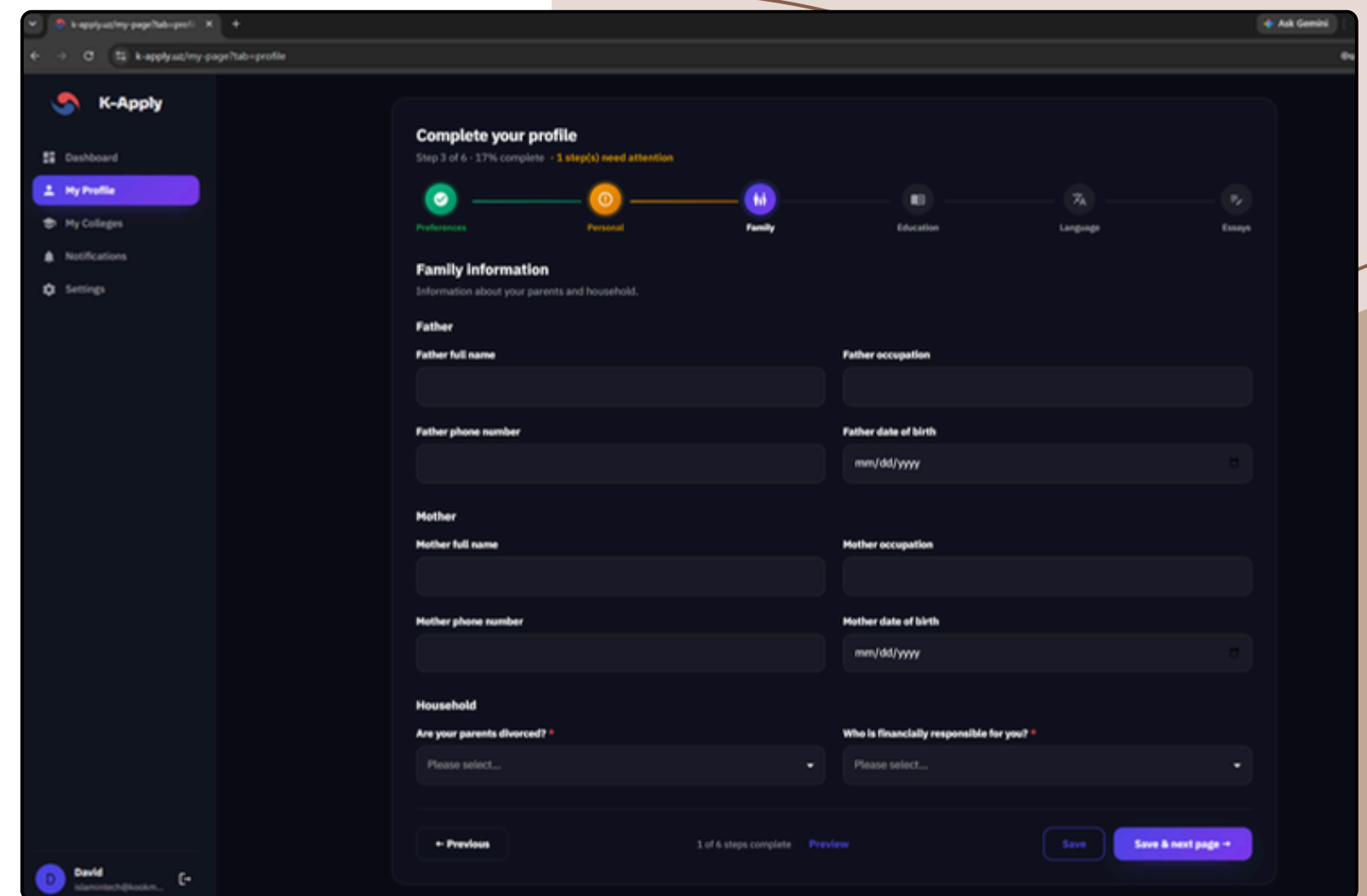
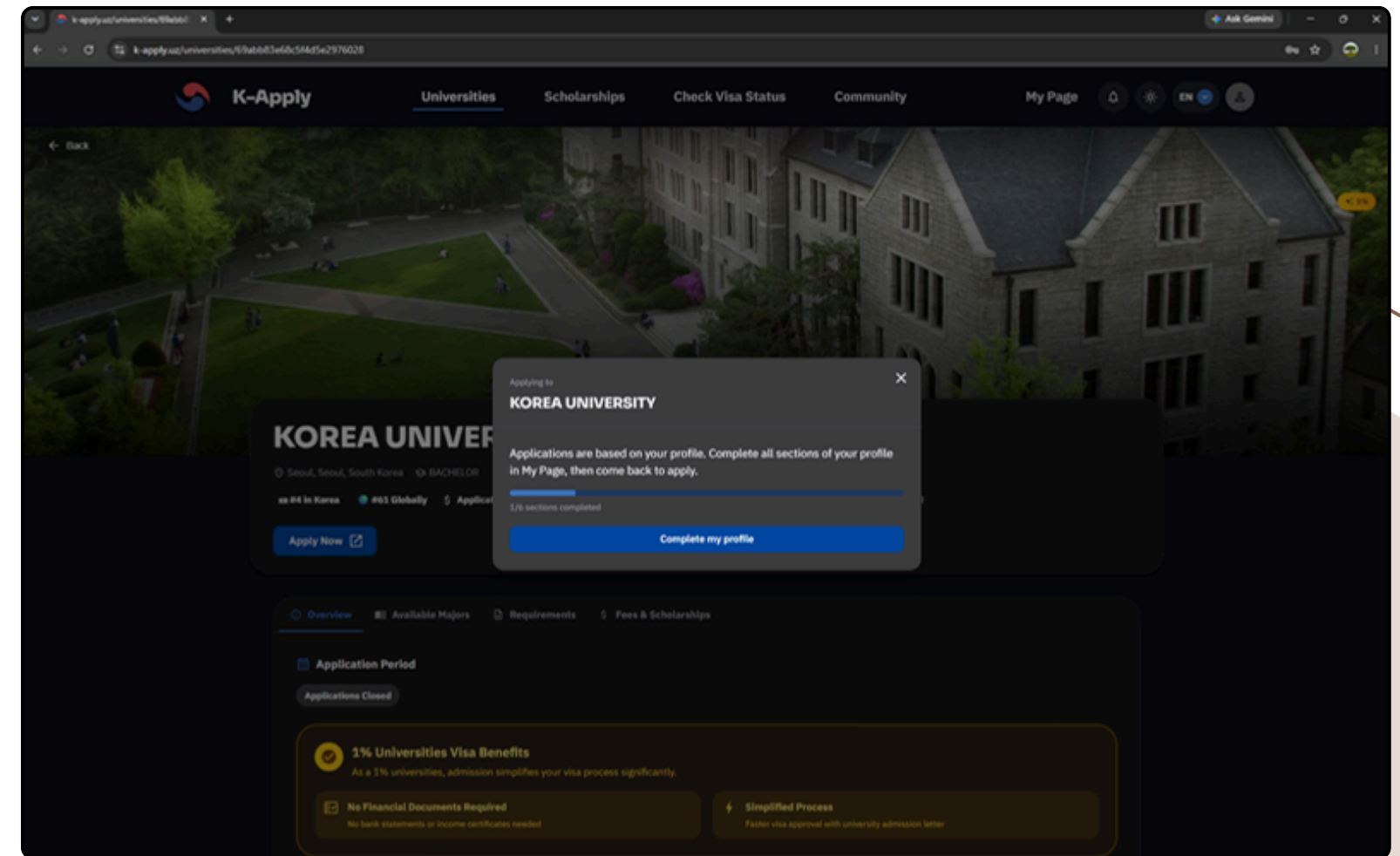
[HTTPS://K-APPLY.UZ](https://k-apply.uz)

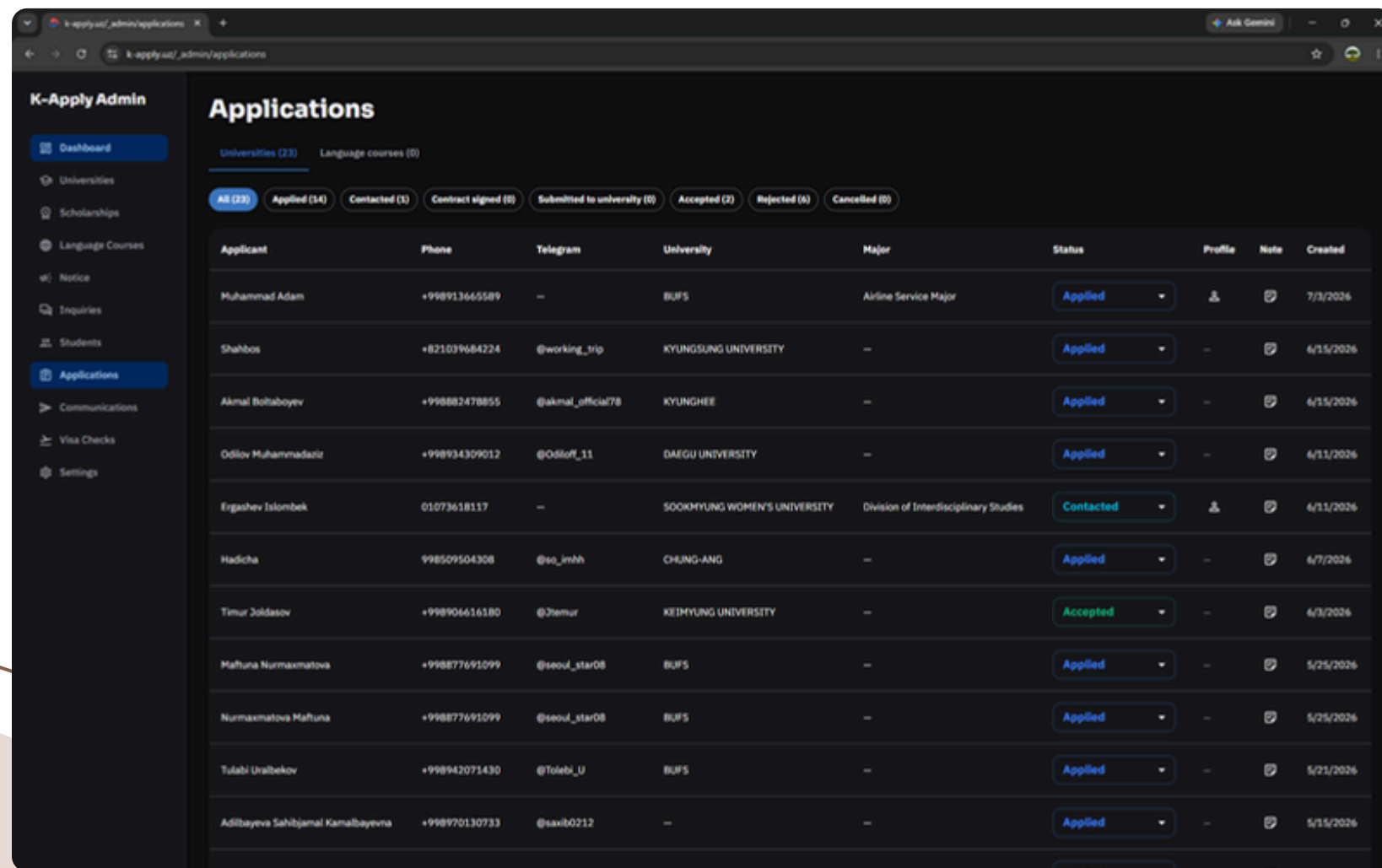


GUIDED STUDENT APPLICATION JOURNEY

Students complete a 6-step profile wizard preferences, personal details, family, education history, language certificates (TOPIK/IELTS), and essays with per-step progress tracked and saved as they go. Applying to a university is gated until the profile is complete, so admins always receive submission-ready applicants. After applying, students track every application's status from their dashboard, and admins review the full profile from the same pipeline.

[HTTPS://K-APPLY.UZ](https://k-apply.uz)





ADMIN PANEL

A dedicated control panel for platform administrators, fully separated from the student experience by role-based guards. Admins manage the entire catalog and every application from one place.

[HTTPS://K-APPLY.UZ](https://k-apply.uz)

- Manage the full catalog universities, scholarship programs, and language courses with rich structured data (rounds, deadlines, tuition, majors)
- Review applications in a pipeline: Applied → Contacted → Contract signed → Submitted → Accepted, with internal notes on every applicant
- Open any applicant's complete student profile (education, certificates, essays) without leaving the review screen
- Publish multilingual notices and answer community inquiries in four languages
- Manage student accounts and control account status platform-wide
- Monitor visa-check requests and platform activity

USED TECHNOLOGIES (K-APPLY)

Frontend

-  Next.js 14 (Pages Router)
-  TypeScript
-  MUI 5
-  Apollo Client
-  SCSS
-  next-i18next (EN·KO·RU·UZ)

Backend

-  NestJS (monorepo: API + batch)
-  GraphQL — code-first (Apollo Server)
-  MongoDB Atlas
-  Mongoose
-  REST + Multer for file uploads

Auth & Security

-  JWT
-  Google OAuth (ID token verification)
-  Telegram Login (HMAC-verified)
-  bcrypt
-  Role guards (Admin/User)
-  API rate limiting (@nestjs/throttler)

Infrastructure

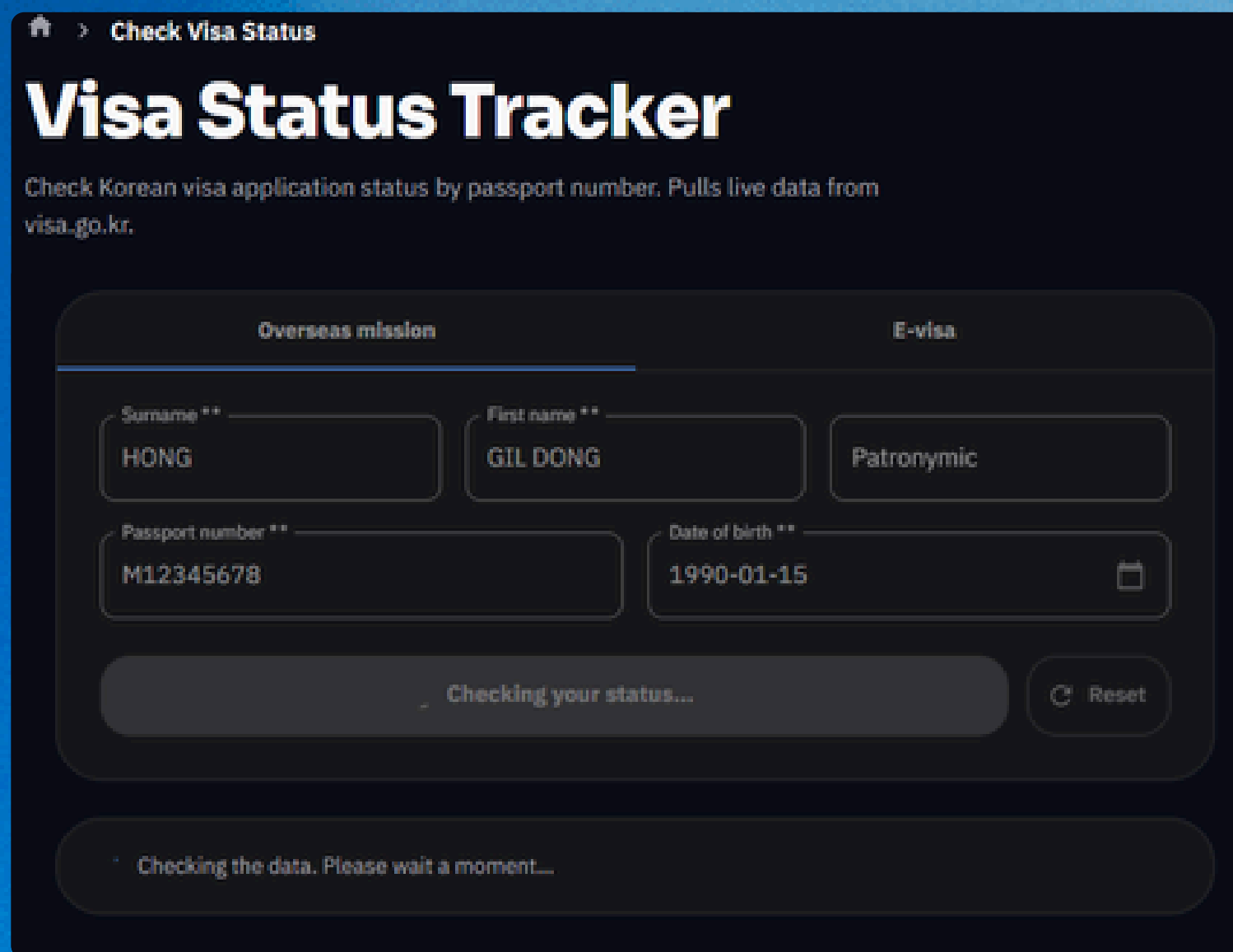
-  Docker Compose
-  nginx + certbot (HTTPS)
-  Linux VPS
-  GitHub deploy key workflow

Media & Analytics

-  Cloudinary (images + documents)
-  PostHog

[HTTPS://K-APPLY.UZ](https://k-apply.uz)

VISA STATUS CHECKER



The screenshot shows a web application titled "Check Visa Status" and "Visa Status Tracker". Below the title, it says "Check Korean visa application status by passport number. Pulls live data from visa.go.kr." The interface has two tabs: "Overseas mission" (selected) and "E-visa". Under the "Overseas mission" tab, there are input fields for "Surname" (containing "HONG"), "First name" (containing "GIL DONG"), and "Patronymic" (empty). Below these are fields for "Passport number" (containing "M12345678") and "Date of birth" (containing "1990-01-15" with a calendar icon). A large button labeled "Checking your status..." is present, along with a "Reset" button. At the bottom, a status bar indicates "Checking the data. Please wait a moment..."

Features: Real-time status lookup, multi-visa type support, notification alerts

Architecture: REST API backend, async job queue, third-party gov API integration

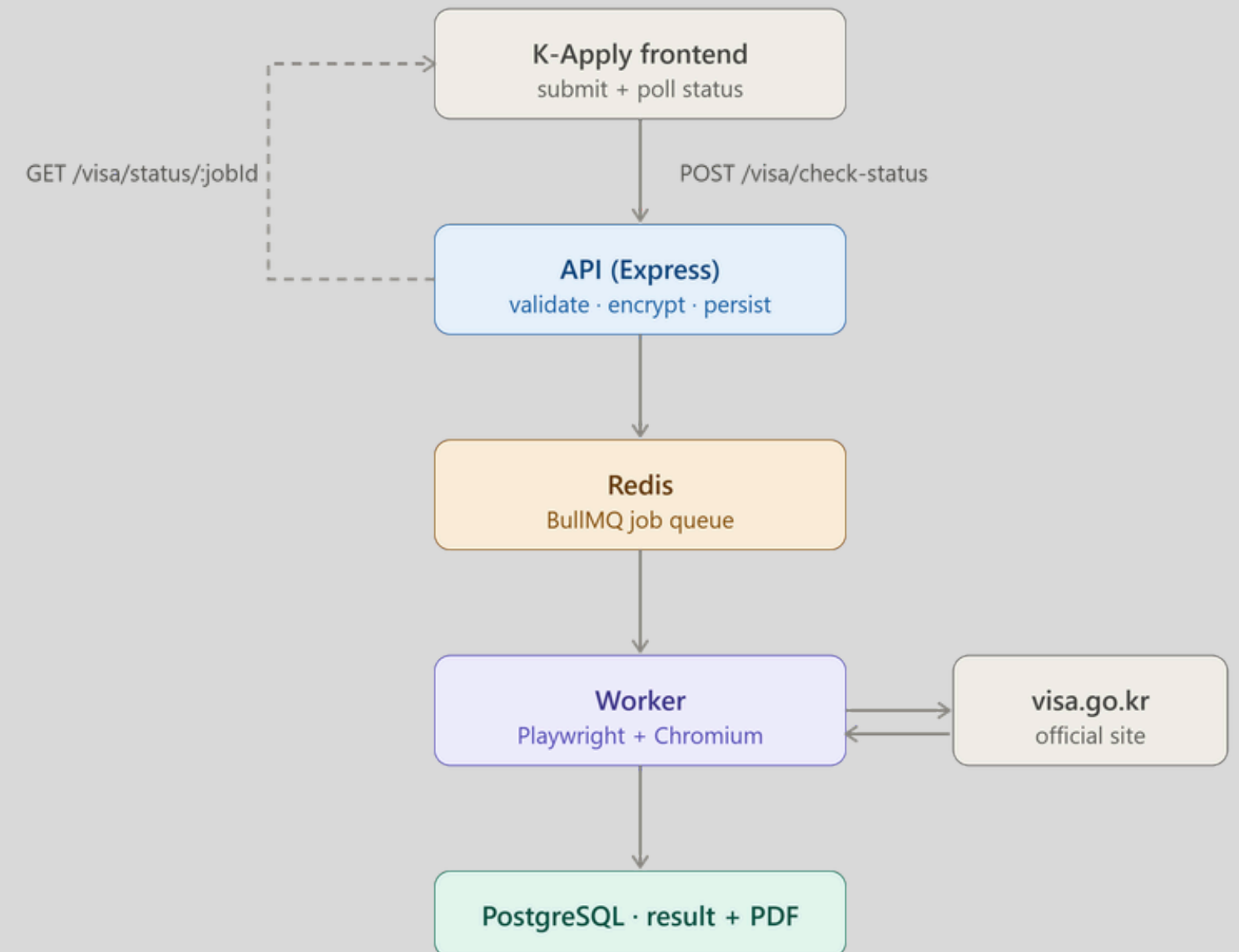
Tech Stack: Next.js, Node.js, PostgreSQL, Redis, Docker, CI/CD pipeline

Security & Deploy: Encrypted data handling, HTTPS, AWS deployment — visastatus.kr

<https://github.com/Islamintech/visa-checker>

System Architecture

The API validates and encrypts the request, persists it, and enqueues a BullMQ job responding in milliseconds. A separate worker process consumes the queue, drives a headless browser through the portal's search flow, and writes the parsed result (and issued-visa PDF) back to PostgreSQL. The client polls the job status: pending → processing → completed / failed. API and worker scale independently, and worker concurrency and job timeouts are configuration-driven.



```
function encrypt(plaintext) {
  const iv = crypto.randomBytes(12);
  const cipher = crypto.createCipheriv('aes-256-gcm', KEY, iv);
  const ct = Buffer.concat([cipher.update(plaintext, 'utf8'), cipher.final()]);
  const tag = cipher.getAuthTag();
  return Buffer.concat([iv, tag, ct]).toString('base64');
}
```

```
const visaQueue = new Queue(queue.name, {
  connection,
  defaultJobOptions: {
    attempts: queue.attempts,
    backoff: { type: 'exponential', delay: 5_000 },
    removeOnComplete: { age: 3600, count: 1000 },
    removeOnFail: { age: 24 * 3600 },
  },
});
```

Security & Reliability

- PII encrypted at rest: passport numbers stored as AES-256-GCM ciphertext (random IV + auth tag); plaintext exists only in memory during a job. Logs only ever see a masked form
- Resilient scraping: exponential-backoff retries per job, attempt tracking in the DB, and per-job timeouts so a hung browser can't stall the queue
- Selector isolation: all portal DOM selectors centralized in one file when the government site's markup changes, it's a one-file fix
- Queue hygiene: completed jobs auto-expire after an hour, failures kept 24h for debugging
- Failure honesty: a lost lookup returns failed with an error message never a fabricated status

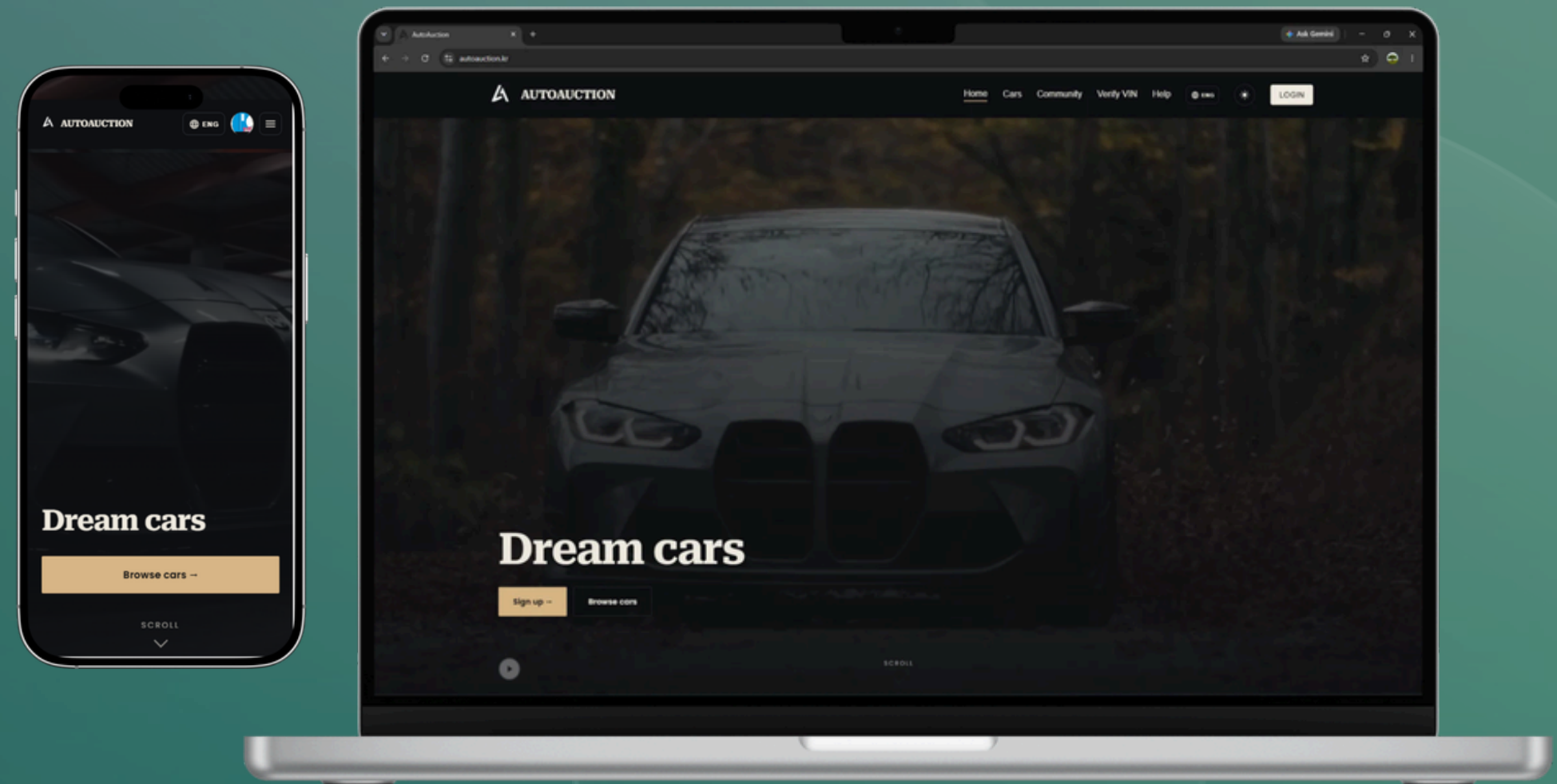
Description

AutoAuction

Car Export Marketplace

A production full-stack marketplace that helps buyers in Central Asia purchase inspected Korean vehicles ready-to-drive or crashed-with-parts from licensed Korean dealers. Buyers browse a filterable catalog, verify any car's history through a public VIN checker backed by real sale records, and book consultations that admins track through a request pipeline. The entire product ships in 4 languages (English, Korean, Russian, Uzbek) with a gamified community members earn points and climb a public leaderboard for activity.

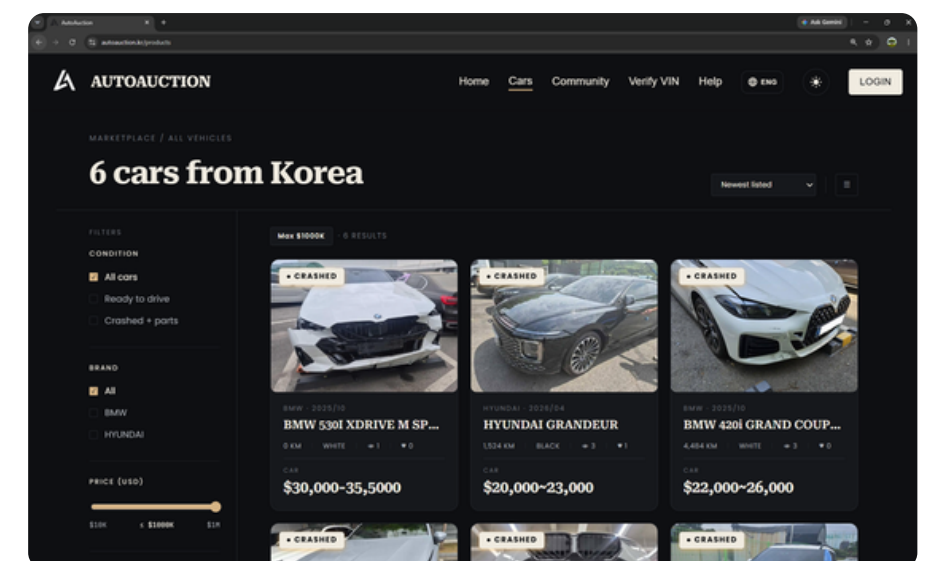
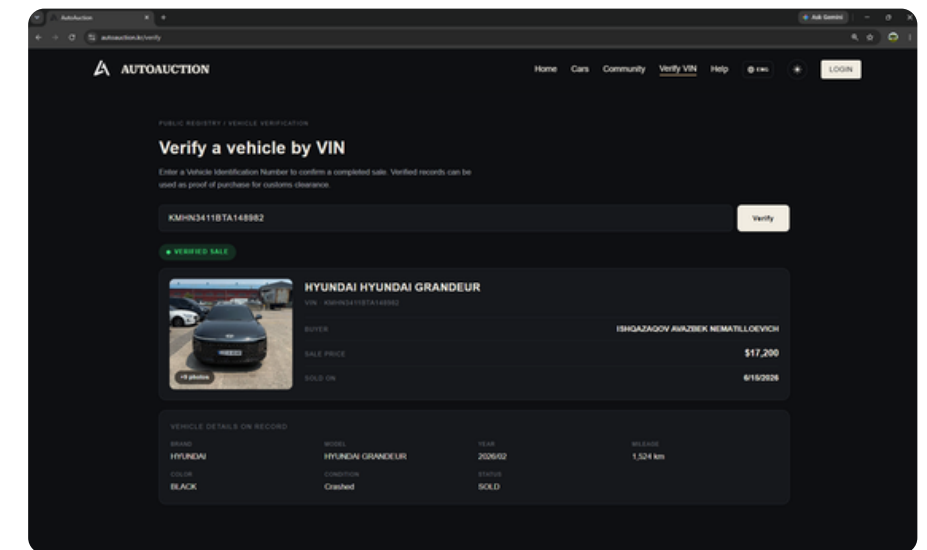
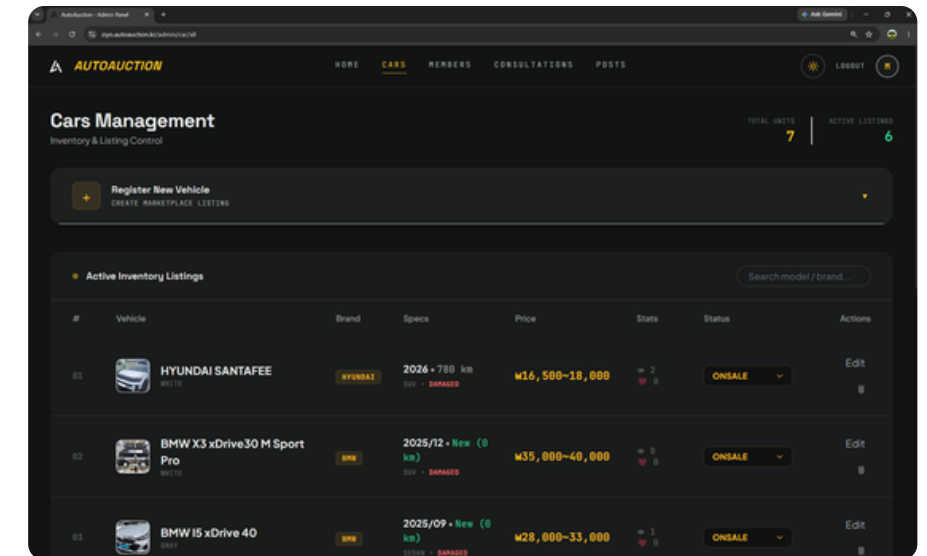
Designed, built, and deployed solo: Node.js + Express + TypeScript REST API with MongoDB/Mongoose, session-based auth and an EJS admin panel on a dedicated subdomain (ziyo.autoauction.kr); React 18 + TypeScript frontend with Redux Toolkit, MUI, and react-i18next; Cloudinary media storage; deployed on Railway behind Cloudflare, live at autoauction.kr (<https://www.autoauction.kr>).



<https://www.autoauction.kr>

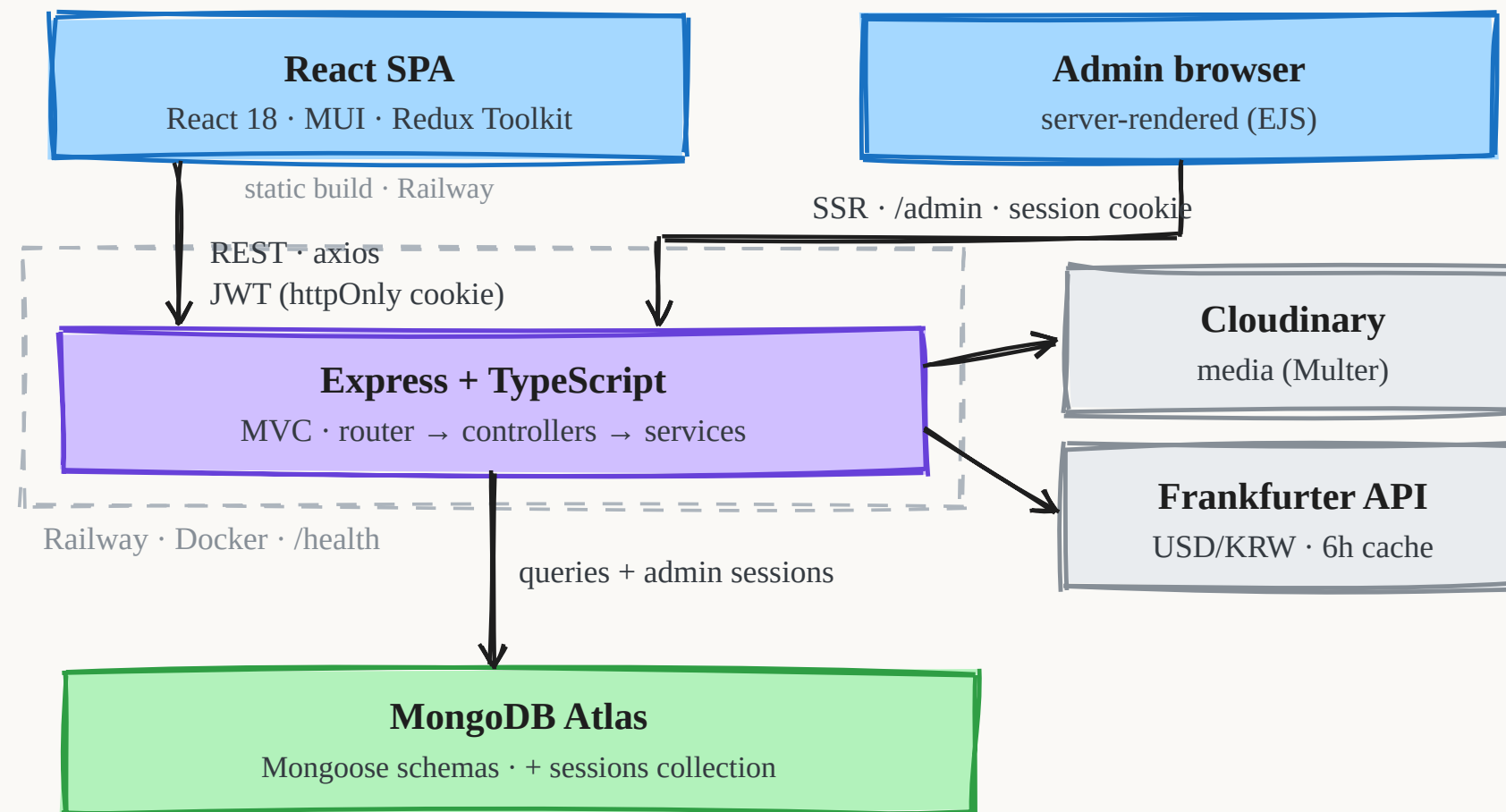
Key Features

- Dual Frontend Architecture React SPA for customers, server-rendered EJS panel for admins
- Dual Authentication JWT (httpOnly cookie) for the SPA, MongoDB-backed sessions for the admin panel
- Car Catalog with Search, Brand Filters, Sorting & Pagination ("Ready" vs "Crashed" categories)
- Public VIN Verification anyone can confirm a car was really sold through the platform (buyer, price, date)
- Guest-friendly Consultations inquire on any car without an account (WhatsApp / Telegram / Email / Call)
- Gamified Points System signup, daily login, views, likes, comments & posts earn points; public Top-Users leaderboard
- Community News Feed with Likes, Comments & unique View Tracking
- Admin Dashboard full car / member / consultation / news management, sale recording (buyer, price, date)
- Prices in KRW with Live USD Estimate (cached exchange-rate service)
- Media Uploads via Cloudinary (up to 10 images per car)
- Multilingual Support (English, Korean, Russian, Uzbek)
- Dark & Light Themes, Fully Responsive Design
- Dockerized Multi-stage Deployment with Healthchecks (Railway)



<https://www.autoauction.kr>

System Architecture

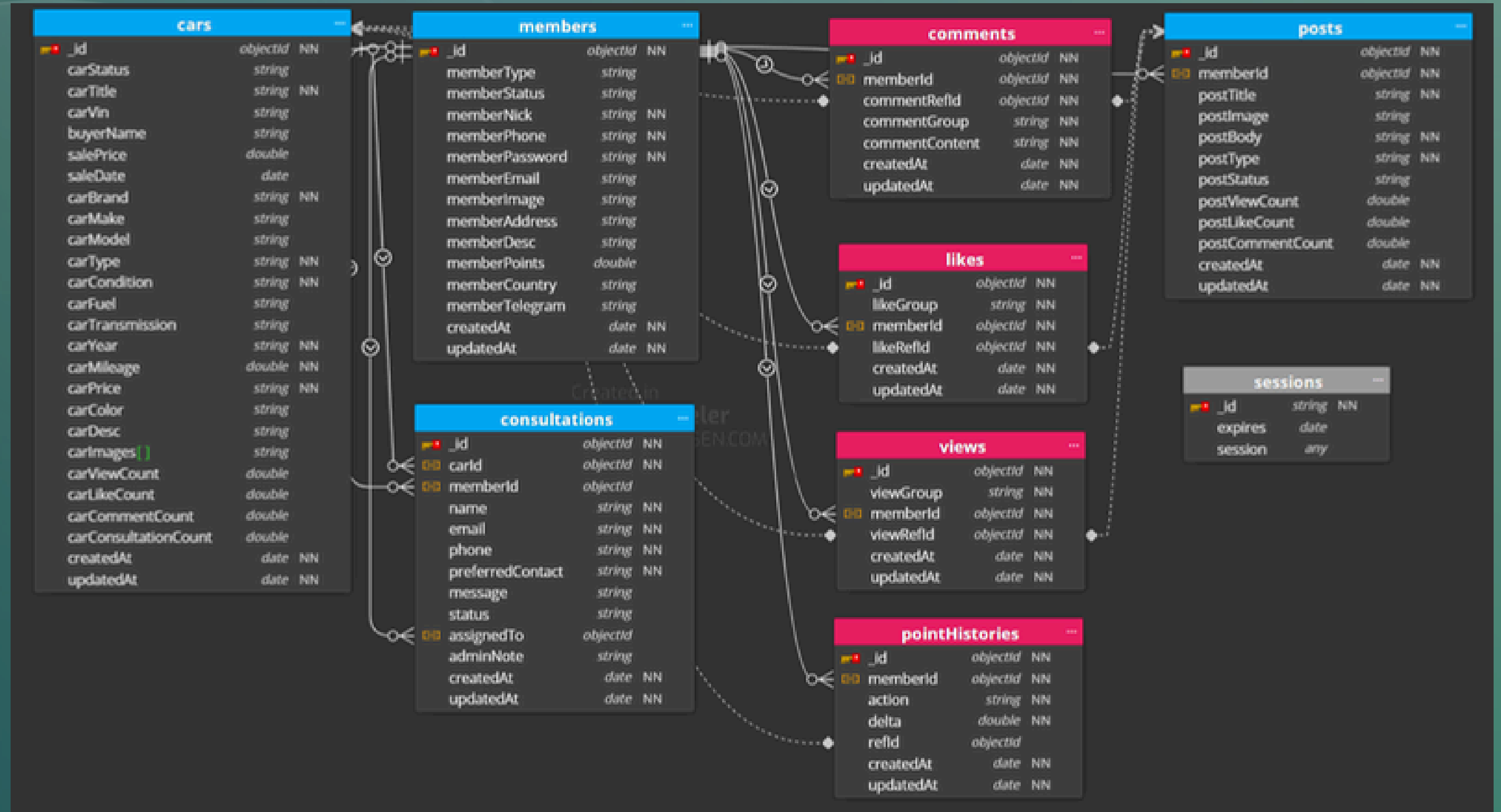


The backend is a single Express + TypeScript app organized as MVC controllers delegate to service classes, and Mongoose schemas own the data layer. It serves two clients at once: the React SPA (Redux Toolkit + axios) consumes a REST API on /, while the admin panel is server-rendered with EJS under /admin. Auth is split by client: customers get a JWT in an httpOnly cookie; admins get an express-session persisted in MongoDB, so an admin stays logged in across deploys behind Railway's proxy.

Car images upload through Multer straight to Cloudinary. A currency service fetches the live USD/KRW rate from an external API and caches it in memory for 6 hours, so every KRW price ships with a fresh USD estimate at zero cost per request. The public VIN endpoint exposes sold-car proof (buyer, sale price, date) to anyone the platform's trust feature. Everything runs as a multi-stage Docker image on Railway with /health probes, and the SPA is deployed separately as a static build.

<https://www.autoauction.kr>

Auto Auction ER Model

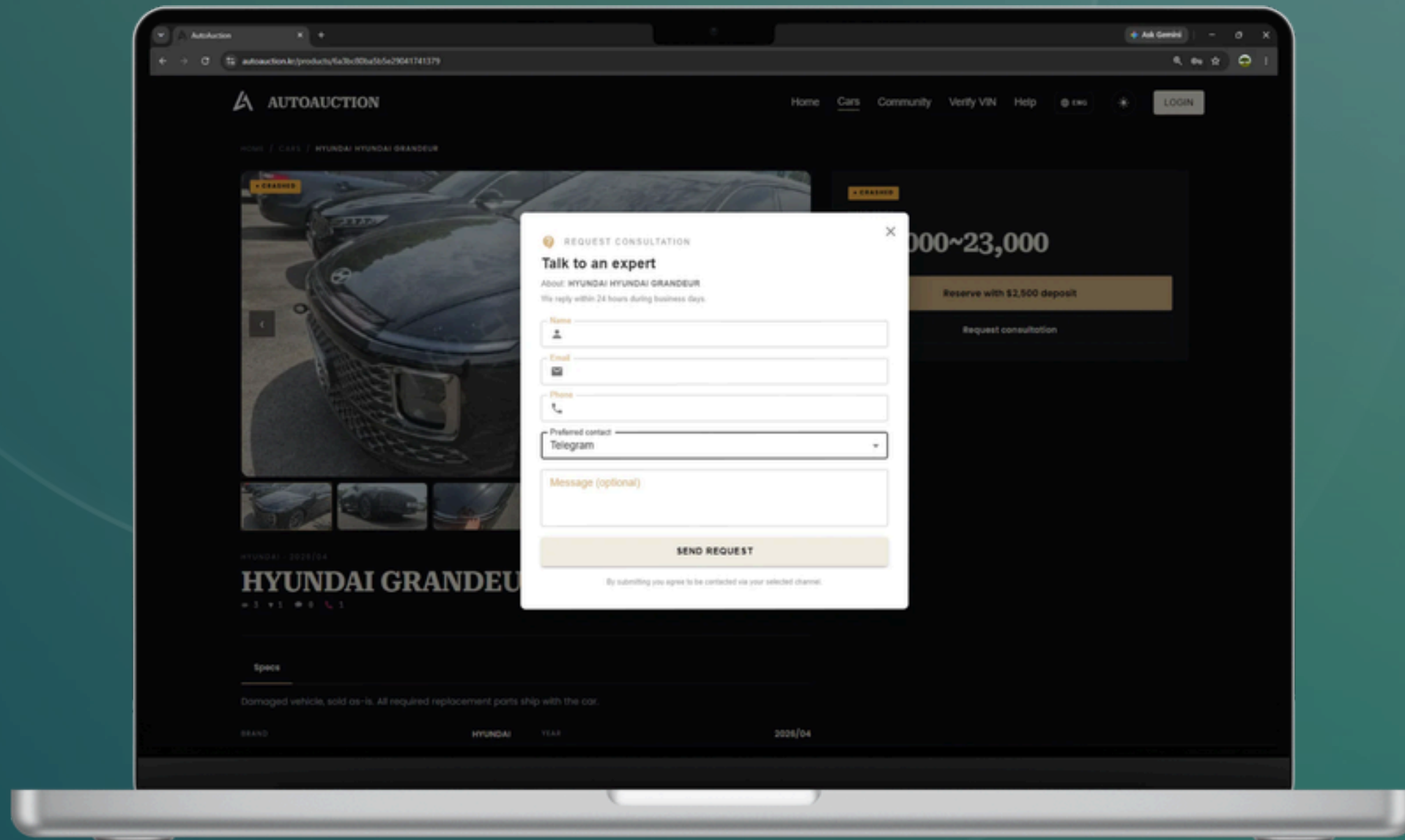


<https://www.autoauction.kr>

Authentication & Profile Management

Two clients, two auth strategies. The React SPA signs in with phone + password over JWT stored in an httpOnly cookie safe from XSS, sent automatically with every request. The admin panel uses classic express-sessions persisted in MongoDB, with secure, sameSite-aware cookies configured for production behind Railway's proxy chain.

Every member has a profile with avatar upload (Cloudinary), contact details, country and Telegram handle, and a live points balance. New signups are awarded 50 points instantly, and a once-per-day login bonus keeps users returning. Role guards keep the /admin surface completely separate from customer routes.

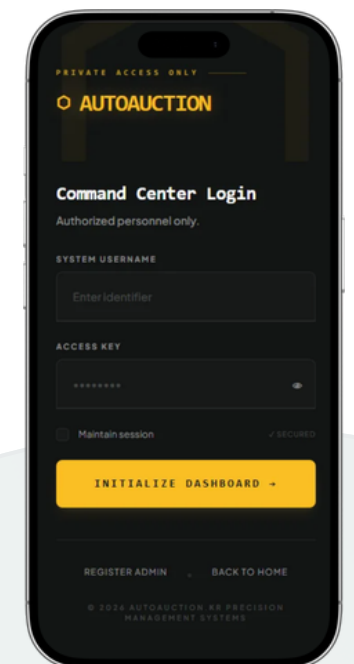
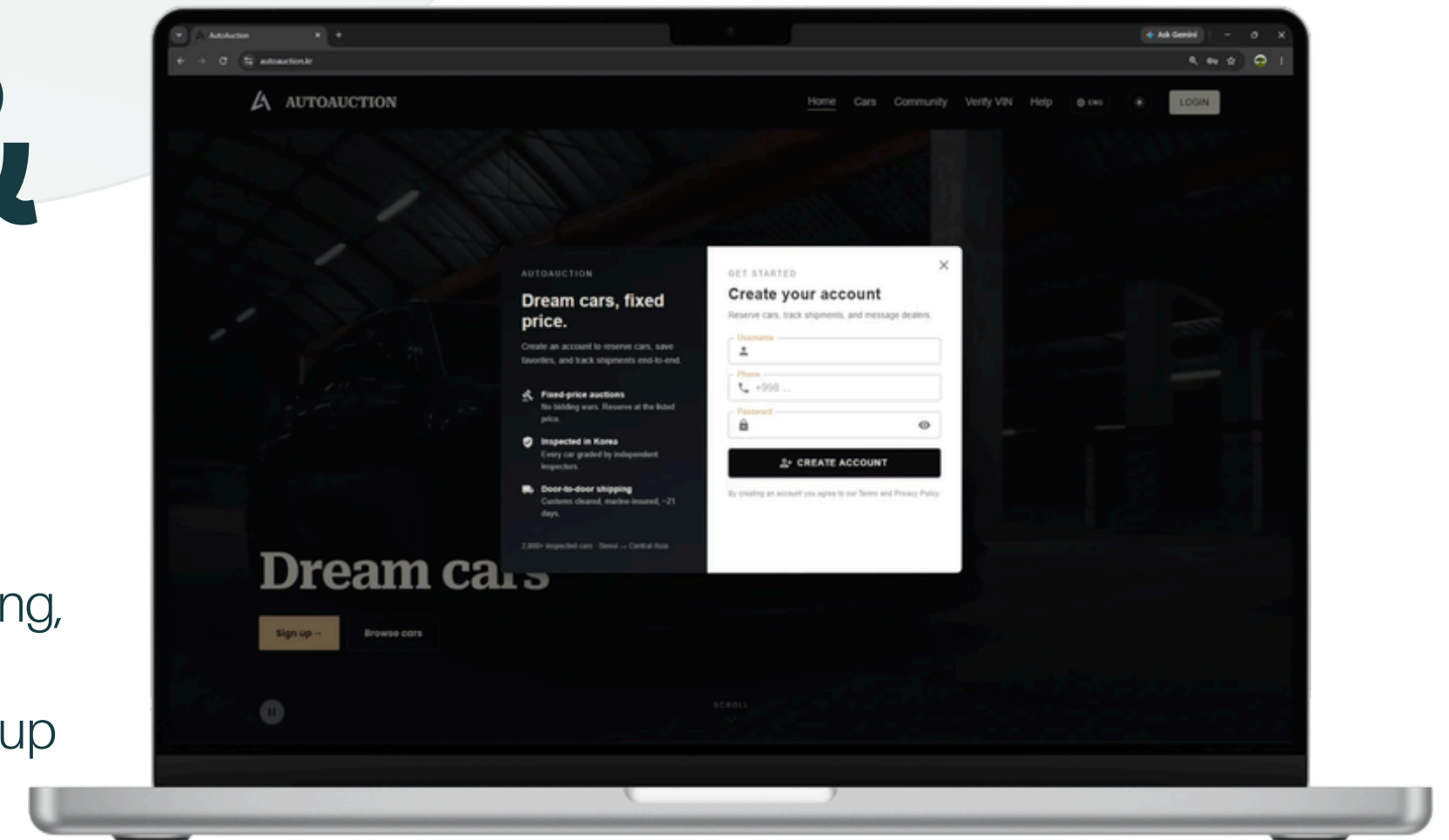


<https://www.autoauction.kr>

Catalog, Filtering & VIN Verification

Buyers browse the catalog by brand, free-text search, price or recency ordering, with pagination split into "Ready to drive" and "Crashed" (repairable-damage) categories that match how export buyers actually shop. Every car ships with up to 10 photos, full specs (fuel, transmission, mileage, year, color), and a KRW price with a live USD estimate.

The trust feature: a public VIN verification page. Anyone a customs officer, a second-hand buyer, a bank can enter a VIN and instantly confirm whether the car was genuinely sold through AutoAuction, including buyer name, sale price, and sale date. Available both as a page in the SPA and as a standalone server-rendered page.

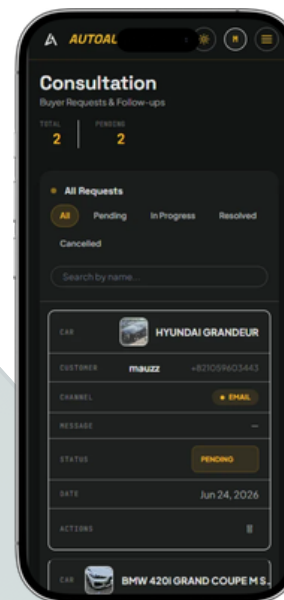
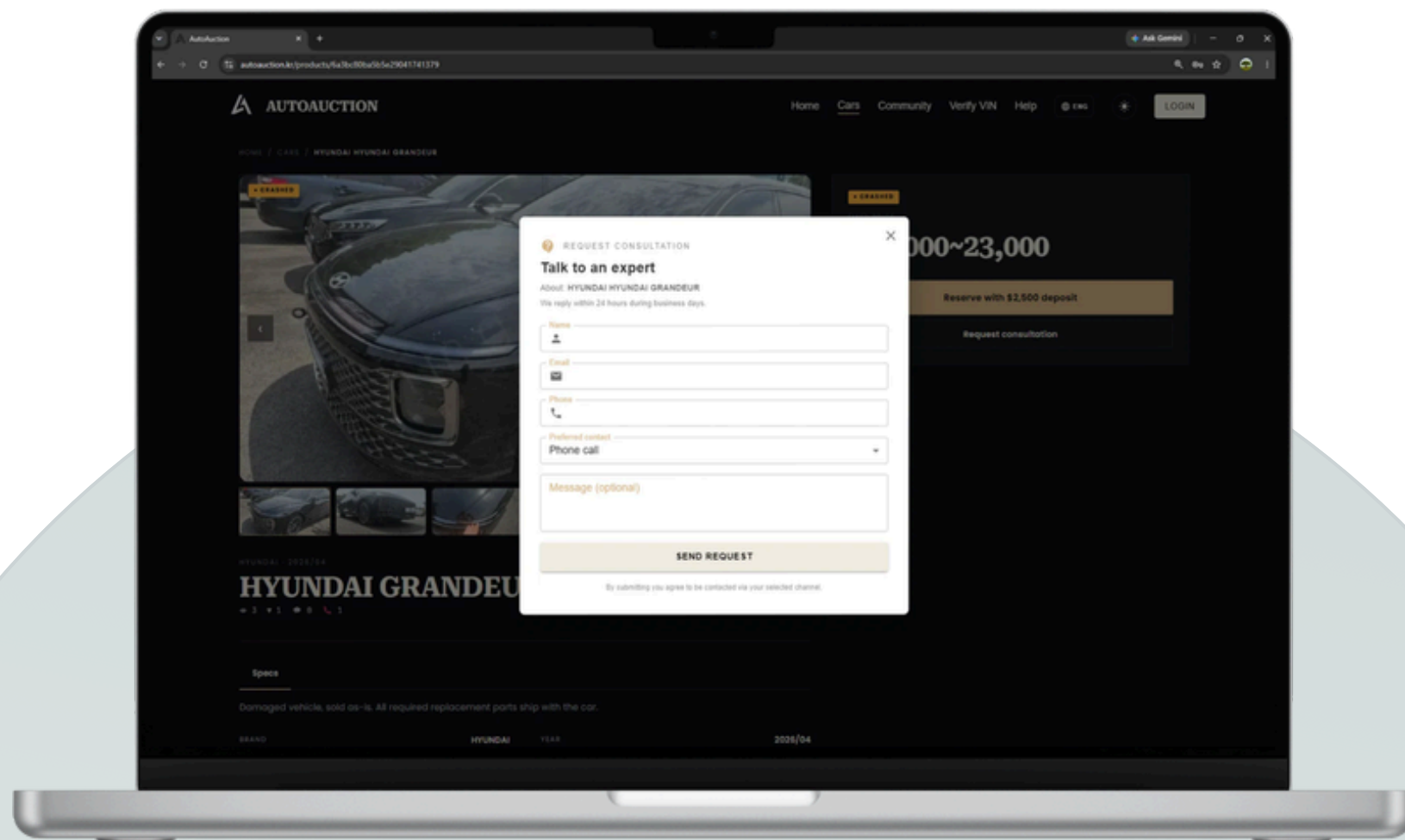


<https://www.autoauction.kr>

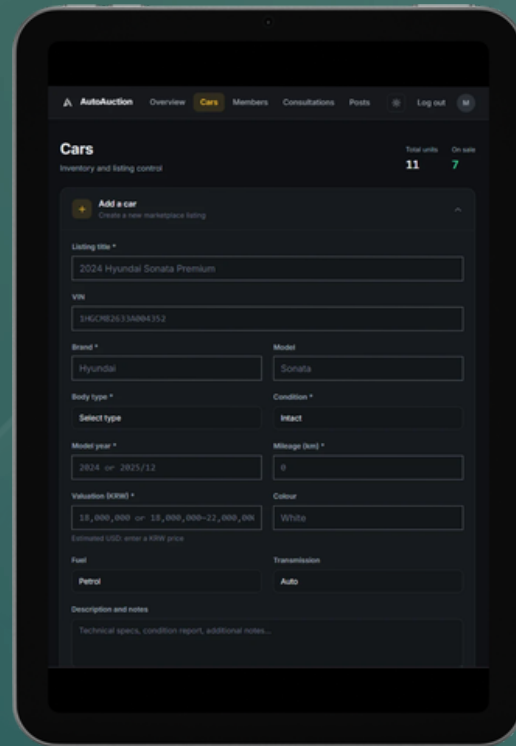
Gamified Community & Consultations

Every interaction earns points: signup (+50), daily login (+5), viewing a car (+1, counted once per member), liking (+3), commenting (+5), and posting (+10). An append-only point-history ledger powers a personal history view and a public Top-Users leaderboard on the landing page.

Consultations are deliberately friction-free: guests can request one on any car without creating an account just name, phone, and a preferred channel (WhatsApp, Telegram, email, or call). Logged-in members track all their requests from their dashboard, while admins triage them through a pipeline: Pending → In Progress → Resolved, with assignment and internal notes.

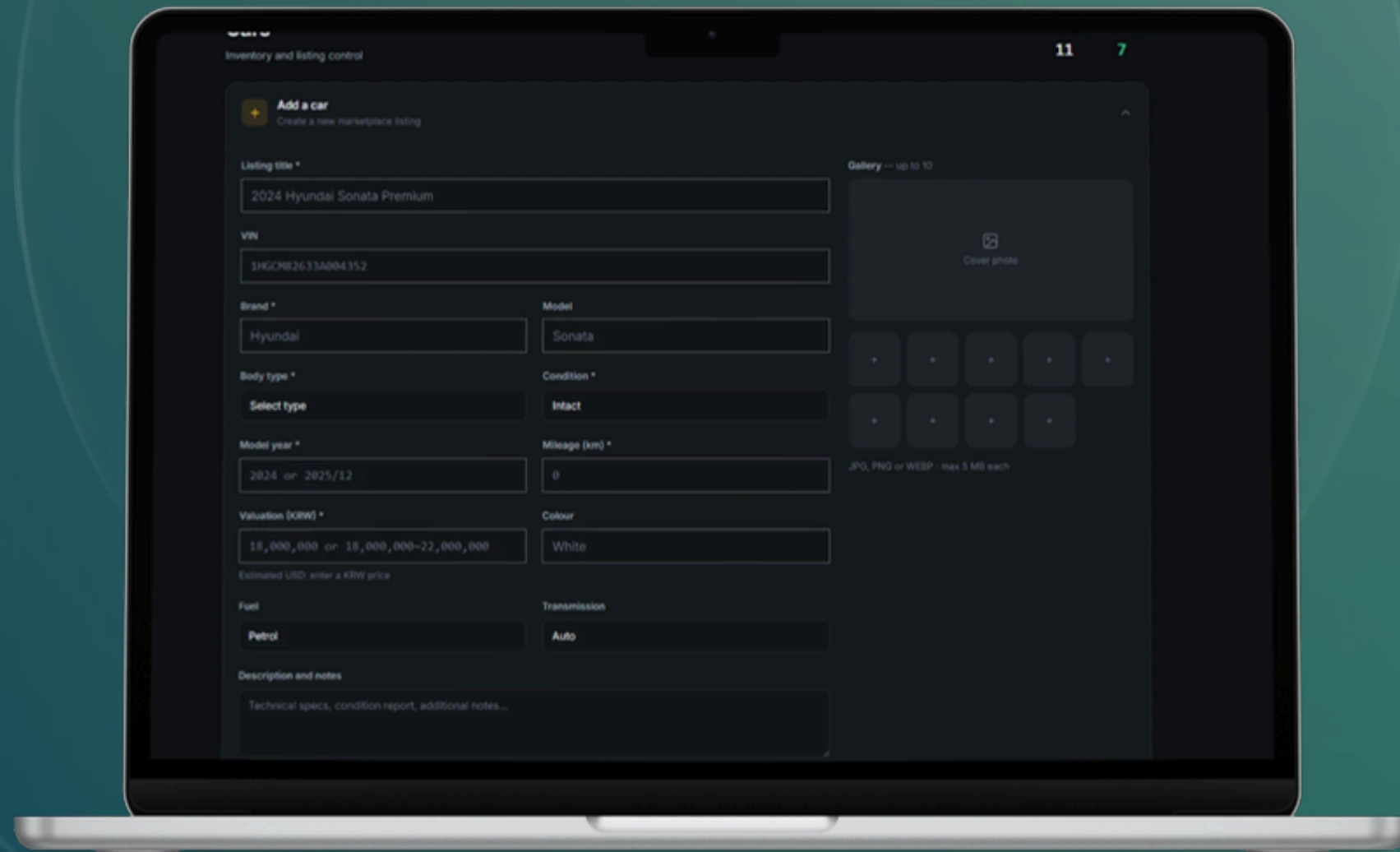


<https://www.autoauction.kr>



Admin Panel

A server-rendered EJS control panel, fully separated from the customer SPA by session-based auth. Admins run the whole business from one place.



- Manage the full car catalog with up to 10 Clouinary images per car, rich specs, and free-form pricing (single price or range, KRW with live USD estimate)
- Record sales: mark a car SOLD with buyer name, sale price, and date this is what powers public VIN verification
- Triage consultations in a pipeline (Pending → In Progress → Resolved / Cancelled) with assignment and admin notes
- Publish news posts and moderate the community
- Manage member accounts and status platform-wide
- Admin subdomain (ziyo.autoauction.kr) routes straight to the panel

<https://www.autoauction.kr>

Used Technologies (AutoAuction)

Frontend

-  React 18 (CRA)
-  TypeScript
-  MUI 7 · styled-components
-  Redux Toolkit · axios
-  react-i18next (EN·KO·RU·UZ)

Media & Integrations

-  Cloudinary (car images, avatars, post images)
-  Frankfurter API (live USD/KRW, 6h in-memory cache)

Backend

-   Node.js + Express (MVC)
-  TypeScript
-  MongoDB Atlas
-  Mongoose
-  REST + Multer for file uploads
-  EJS server-side rendering (admin panel)

Infrastructure

-  Docker (multi-stage, non-root runtime)
-  Railway (healthcheck /health, restart policy)

Auth & Security

-  JWT (httpOnly cookie) — SPA users
-  express-session + MongoDB store
-  Telegram Login (HMAC-verified)
-  bcrypt password hashing
-  Role guards (Admin/User) · CORS allowlist
-  Secure proxy-aware cookies in production

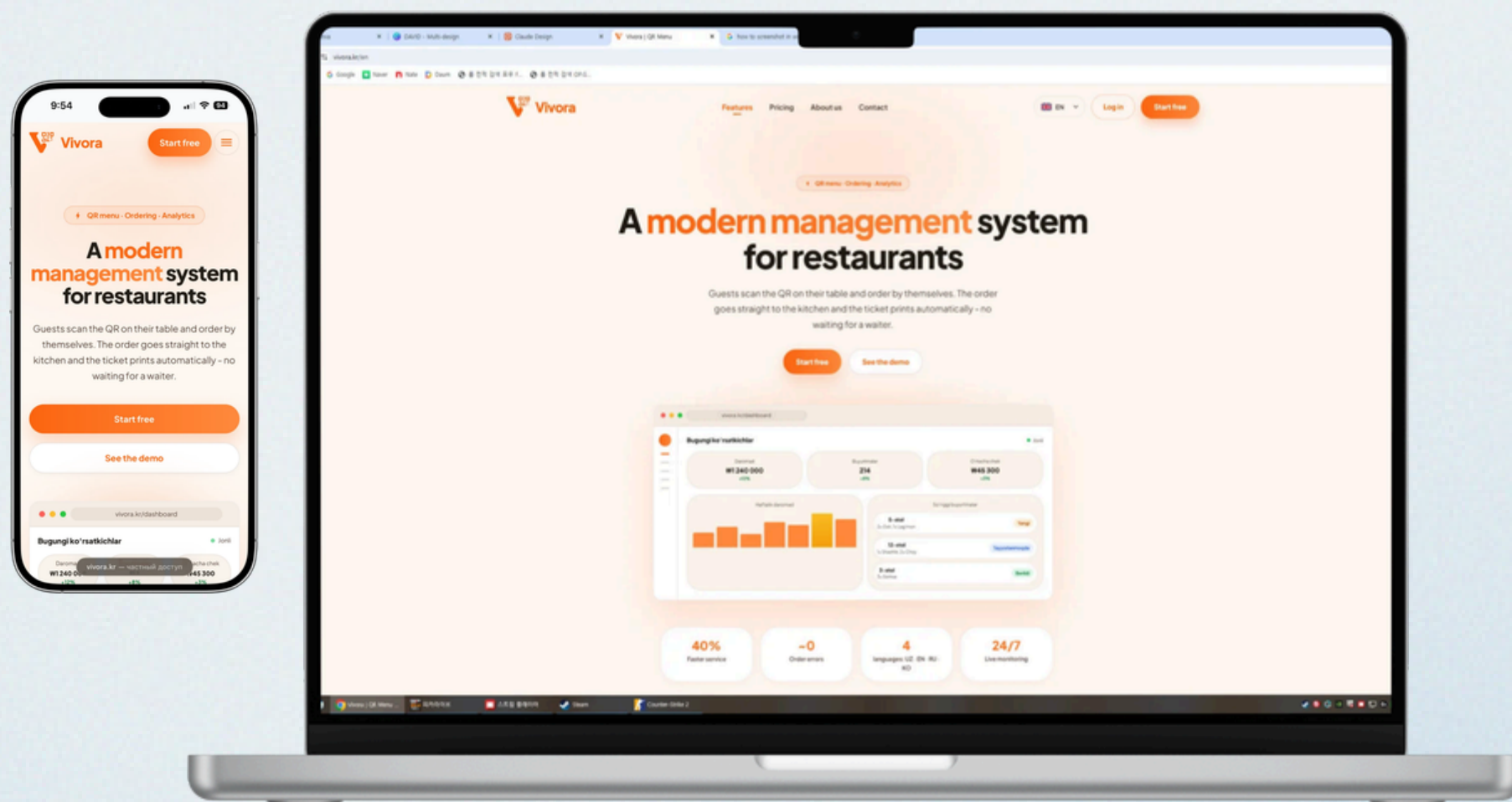
<https://www.autoauction.kr>

DESCRIPTION VIVORA

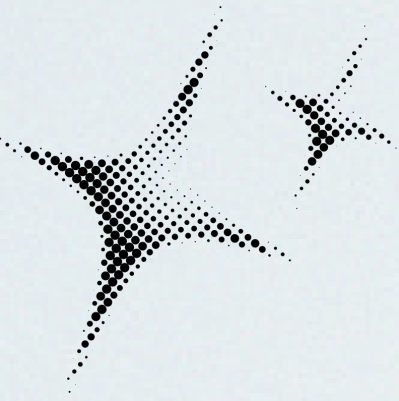
RESTAURANT MANAGEMENT PLATFORM

A production multi-tenant SaaS that lets a restaurant take orders without waiters, apps or a POS terminal. Guests scan the QR code on their table, browse a live menu in their own language, and order from their phone; the ticket reaches the kitchen display in under a second and prints on the thermal printer by the pass. Owners run menus, tables, running tabs and revenue from one dashboard. The platform holds new signups for approval and invoices each restaurant a flat monthly fee. Built for Uzbek-run restaurants in South Korea, in 4 languages.

Designed, built, and deployed solo: NestJS + TypeScript GraphQL API (code-first) with MongoDB/Mongoose, JWT auth and role guards, and live order feeds over graphql-ws; Next.js 14 + MUI v5 frontend with Apollo Client and Zustand; a standalone Node print agent driving ESC/POS printers on the restaurant's own LAN; Cloudinary media, Telegram staff alerts, Atlas database; deployed on Railway and Vercel, live at vivora.kr.



<https://vivora.kr>



KEY FEATURES

QR ordering, zero install - Scan a table code, order in your language, watch the status change live. No app download, no account.

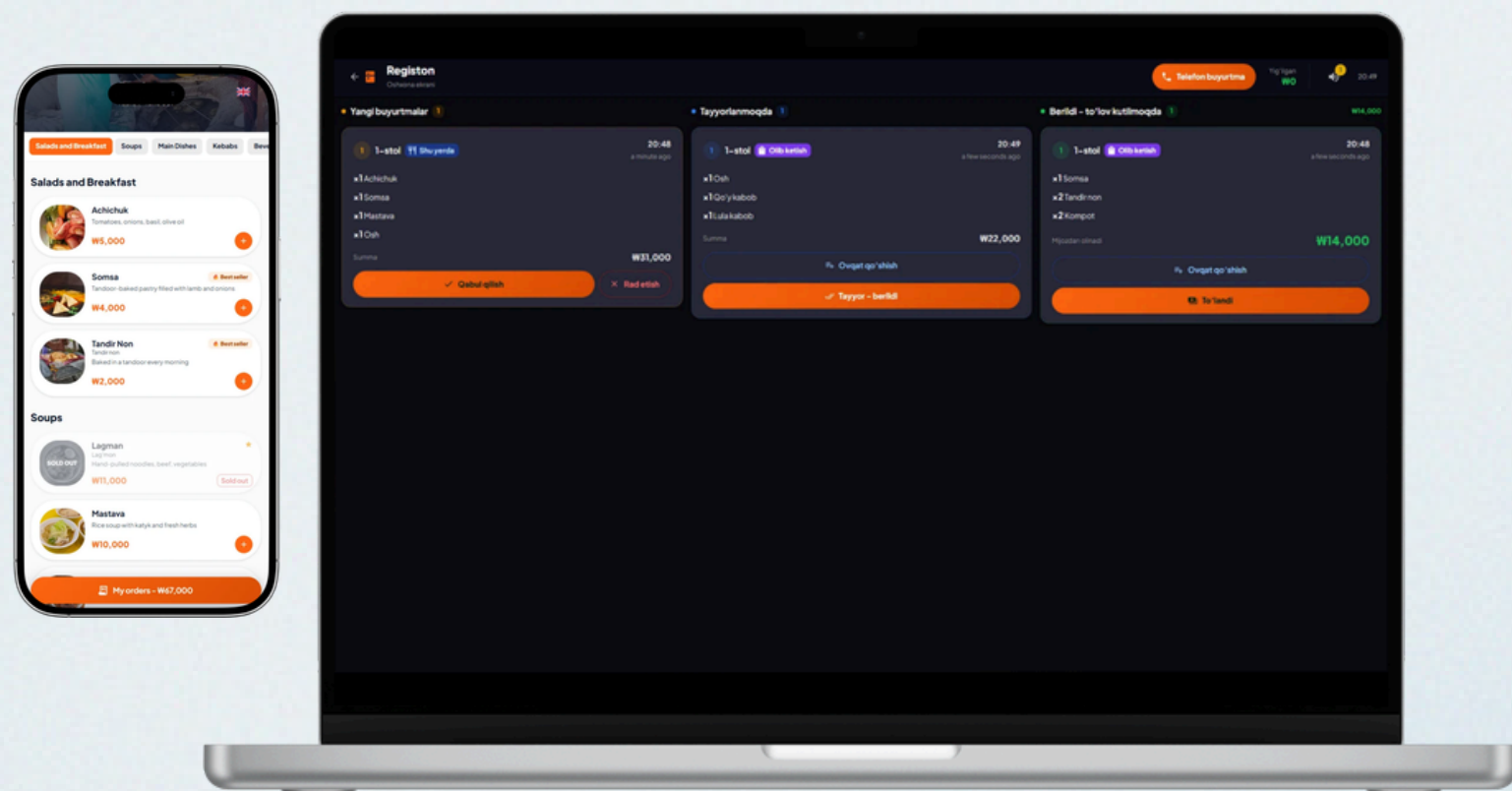
Real-time kitchen display - Orders arrive on a dark always-on board over WebSocket subscriptions, and print automatically on the local thermal printer.

Running table tabs - The first scan opens a tab; every later order joins it. A database-level index guarantees one visit never splits into two bills.

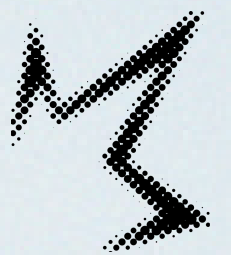
Four languages, one kitchen language - Guests read English, Korean or Russian. Tickets and reports stay in Uzbek, because that is what the cooks read.

Self-driving order lifecycle - Orders accept, serve and settle themselves on a timer, so service completes even when nobody touches a screen.

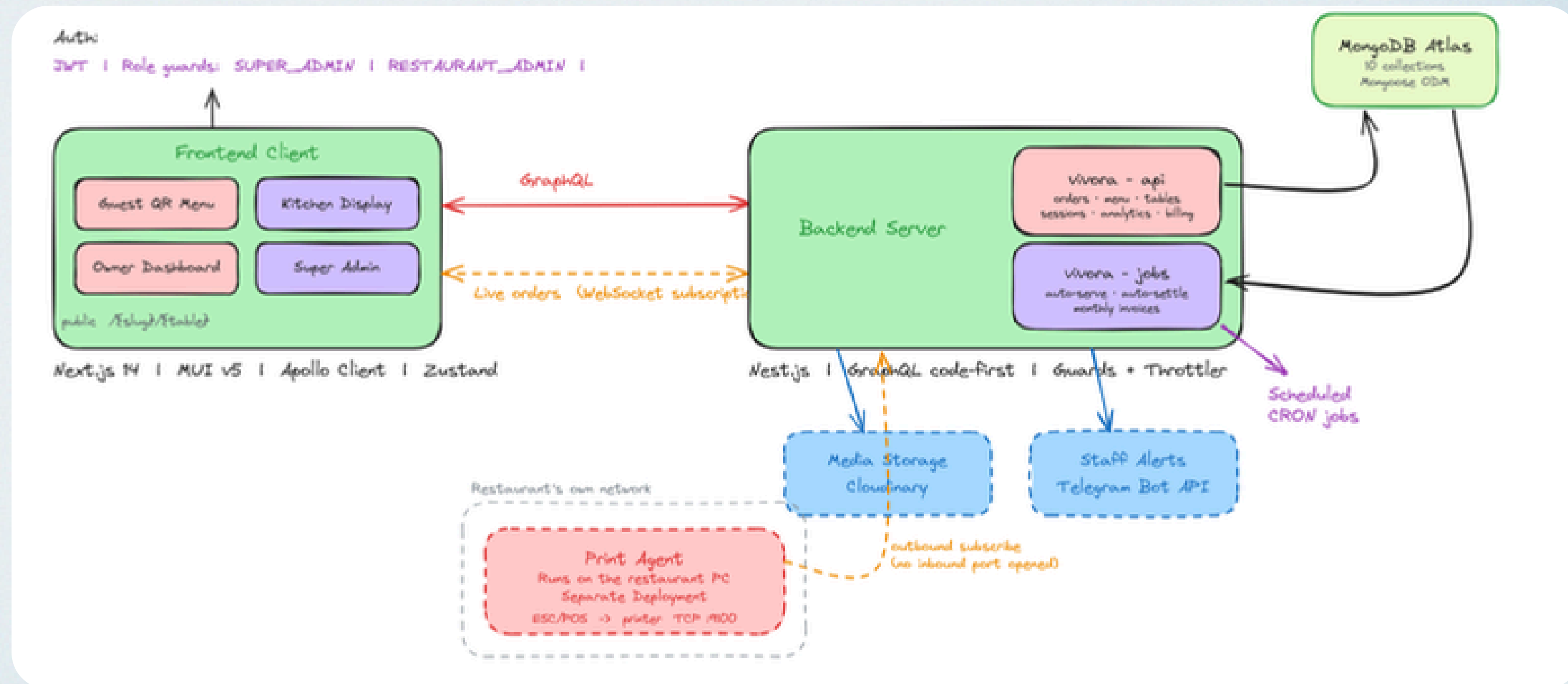
Owner analytics - Revenue, peak hours, best-selling dishes and average ticket, computed in MongoDB aggregation pipelines and charted in the dashboard.



<https://vivora.kr>

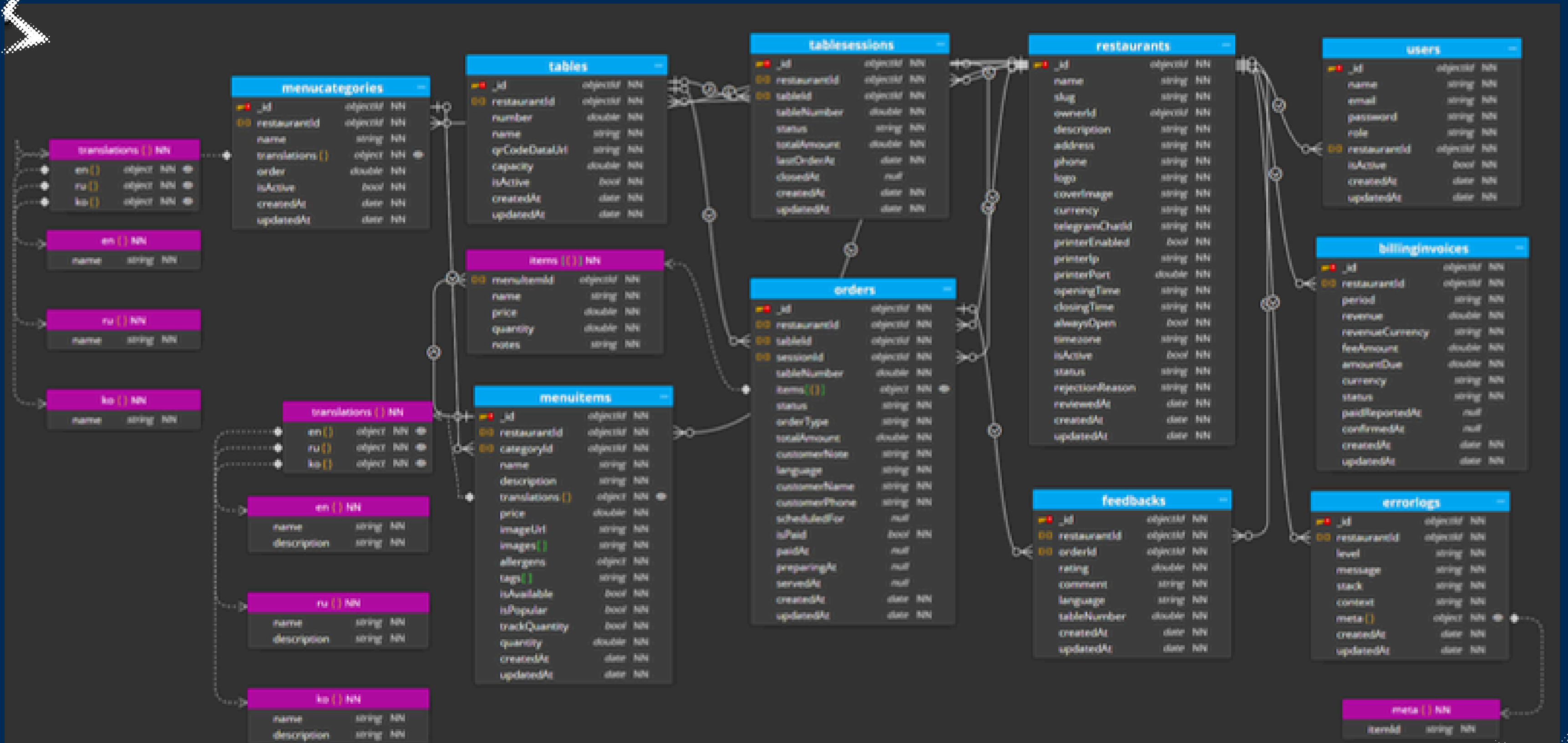


SYSTEM ARCHITECTURE ⚡



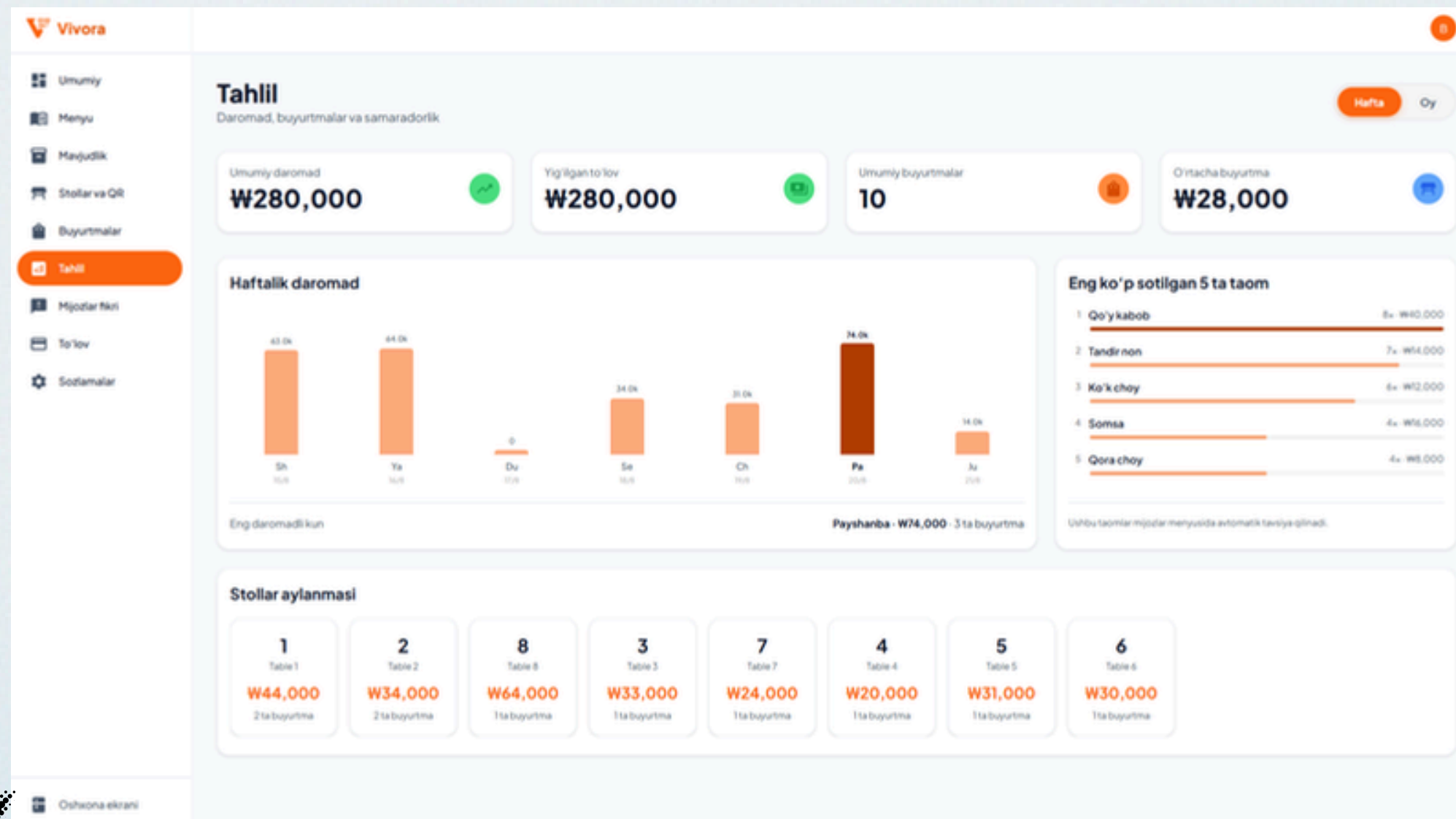
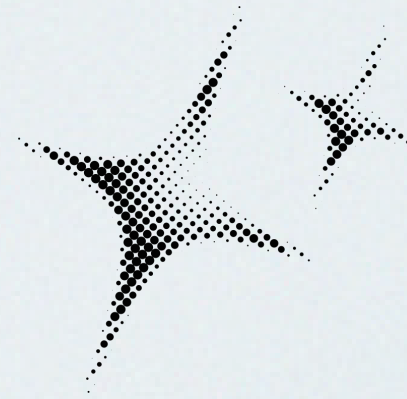
<https://vivora.kr>

VIVORA ER MODEL



<https://vivora.kr>

RESTAURANT ADMIN PANEL



Overview -today's revenue, live orders, open tabs

Orders -full list, status changes, phone orders, payment

Menu -categories and dishes, four languages, photo upload

Availability -prepped portions, auto sold-out at zero

Tables -table setup, QR codes, open tabs, close bill

Analytics -revenue trend, peak hours, top dishes

Feedback - guest ratings and comments

Billing - monthly invoices, report a bank transfer

Settings - hours, timezone, currency, printer, Telegram

<https://vivora.kr>

SUPER ADMIN PANEL

The platform operator's view, gated by a role guard rather than a hidden URL. Restaurants sign up but stay locked until a super admin approves them, so nobody goes live unreviewed. Seven tabs cover approvals, tenants, users, orders, analytics, billing and the error log.

Suspending a restaurant is separate from rejecting one: an approved restaurant can be switched off and back on without losing its approval.

Platforma admini

Platforma ko'rinishi
Barcha restoranlar va platforma faoliyatini kuzating

Jami restoranlar
5
4 faol

Jami foydalanuvchilar
7

Jami buyurtmalar
62

Umumiy daromad
W1,476,000

1 ta restoran tasdiqlashni kutmoqda.

Restoranlar

So'nggi buyurtmalar

Xatolar jumali

Tasdiqlash

Foydalanuvchilar

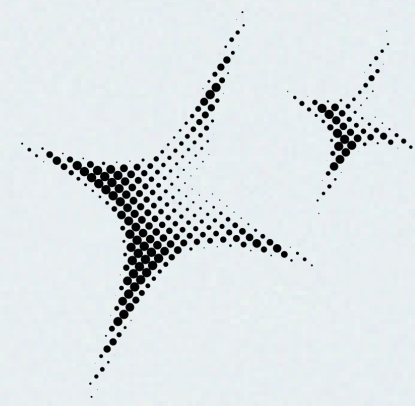
Tahil

To'lovlar

RESTORAN	SLUG	TASDIQ	FAOL	BUYURTHALAR	DAROMAD	YARATILGAN	AMAL
1 Registon	registon	Tasdiqlangan	Faol	10	W280,000	Aug 21, 2026	O'chirish
2 Toshkent Lagmon Uyi	toshkent-lagmon	Ko'rib chiqilmoqda	Faol	0	W0	Aug 19, 2026	Tasdiqlash Rad etish
3 Farg'ona Choyxona	fargona-choyxona	Rad etilgan	Faol	0	W0	Aug 10, 2026	Tasdiqlash
4 Samarqand Osh Markazi	samarqand-osh	Tasdiqlangan	Faol	34	W778,000	May 17, 2026	O'chirish
5 Buxoro Tandir	buxoro-tandir	Tasdiqlangan	Notfaol	18	W418,000	Mar 31, 2026	Faollashtirish

<https://vivora.kr>

USED TECHNOLOGIES (VIVORA)



Language and runtime



TypeScript 5.3

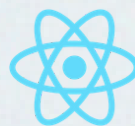


Node 20

Frontend



Next.js 14 (Pages Router)



React 18



MUI v5 with Emotion



Apollo Client 3 with a graphql-ws link



Zustand for auth state

Infrastructure



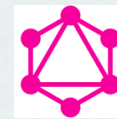
Compose

Docker and Docker Compose

Backend



NestJS 10



GraphQL code-first via @nestjs/graphql + @nestjs/apollo (Apollo Server 4)



MongoDB 7 with Mongoose 8



Passport-JWT + bcryptjs

External services



Google Gemini API (menu translation)

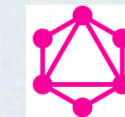


Telegram Bot API (staff alerts and inbound button taps)



Cloudinary (unsigned image upload)

Real-time



GraphQL Subscriptions over graphql-ws



Graphql-subscriptions PubSub

Print agent



Node



node-windows to install as a Windows service,

<https://vivora.kr>

Contact ME



ergashevisloombek@gmail.com



010-7361-8117



github.com/Islamintech



Ergashev Islombek

