RideShare Platform

End-to-End Ride-Hailing Solution — Client Case Study

Prepared for: Client Presentation

Date: June 2026

Version: 1.0

Executive Summary

RideShare Platform is a production-grade ride-hailing ecosystem comprising three integrated applications: a passenger mobile app, a driver mobile app, and a web-based operations dashboard. Together, they deliver the complete lifecycle of on-demand transportation—from ride request and driver matching through trip completion, payment, and operational oversight.

Built on modern, scalable technologies (Flutter, Laravel 11, PostgreSQL with PostGIS, Redis, and real-time WebSockets), the platform is designed for reliability, security, and growth. Operations teams gain live visibility into fleet activity, while riders and drivers benefit from intuitive mobile experiences backed by automated fare calculation, surge pricing, in-ride safety tools, and push notifications.

The Challenge

Organizations entering the mobility market face a common set of requirements:

1. **Unified experience** across rider, driver, and admin touchpoints
2. **Real-time coordination** for matching, tracking, and status updates
3. **Operational control** over drivers, vehicles, pricing, and support
4. **Trust and safety** features including SOS, audit trails, and role-based access
5. **Scalable infrastructure** that can grow with demand without re-architecting

Point solutions or disconnected apps fail to meet these needs. RideShare Platform addresses them with a single backend, shared data model, and consistent API—reducing integration cost and time-to-market.

Solution Overview

RideShare Platform consists of three coordinated products sharing one Laravel backend:

Component	Purpose	Primary Users
Rider App	Book rides, track trips, pay, rate, and access safety tools	Passengers
Driver App	Onboard, go online, accept offers, complete trips, view earnings	Drivers
Operations Dashboard	Monitor live fleet, manage users, resolve support, configure surge	Admin, Ops, Support

All mobile clients authenticate via phone OTP and communicate with a REST API (`/api/v1`). Real-time updates flow through Laravel Reverb (Pusher-compatible WebSockets). Driver locations are indexed in Redis for sub-second geospatial queries.

Rider Experience

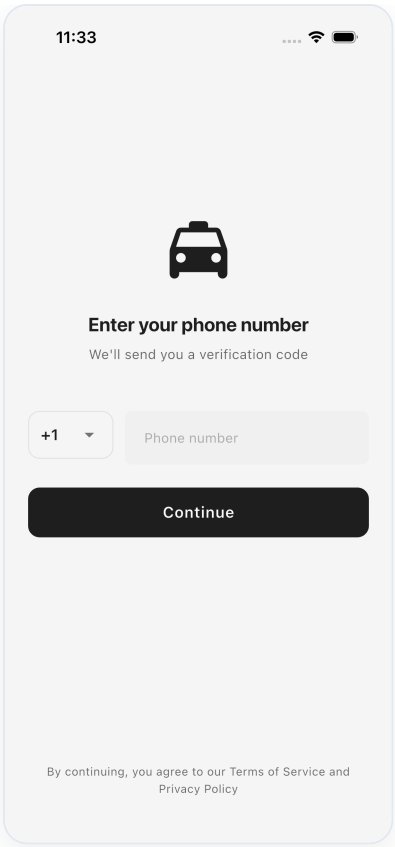
The Rider App is a Flutter application built with Riverpod state management, Material 3 design, and Google Maps integration. It supports English and Urdu localization with RTL readiness.

Key Capabilities

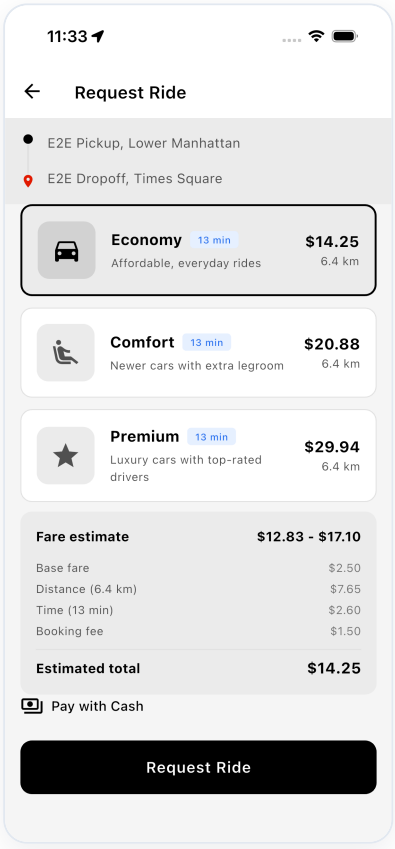
1. **Phone OTP authentication** with secure token storage (Sanctum)
2. **Map-first home screen** with pickup/dropoff selection and vehicle type choice
3. **Fare estimates** with surge multiplier visibility before booking
4. **Live ride tracking** through every status: searching → assigned → arriving → in progress → completed
5. **In-ride chat** with the assigned driver
6. **Safety center** with SOS that creates support tickets for operations
7. **Wallet & payments** including receipt generation
8. **Ride history & ratings** with post-trip feedback
9. **Push notifications** via Firebase Cloud Messaging
10. **Offline resilience** with Isar local storage

User Journey

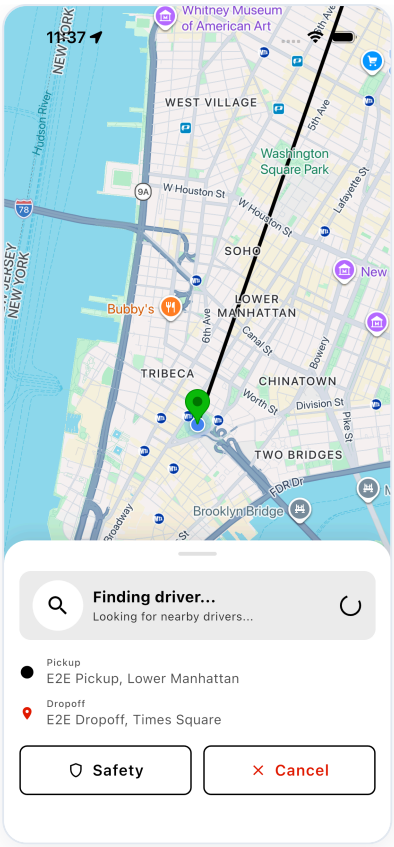
1. Sign in with phone number and OTP
2. Set pickup and destination on the map
3. Review fare estimate and confirm ride
4. Track driver arrival and trip progress in real time
5. Message driver or trigger SOS if needed
6. View receipt, rate the trip, and access history



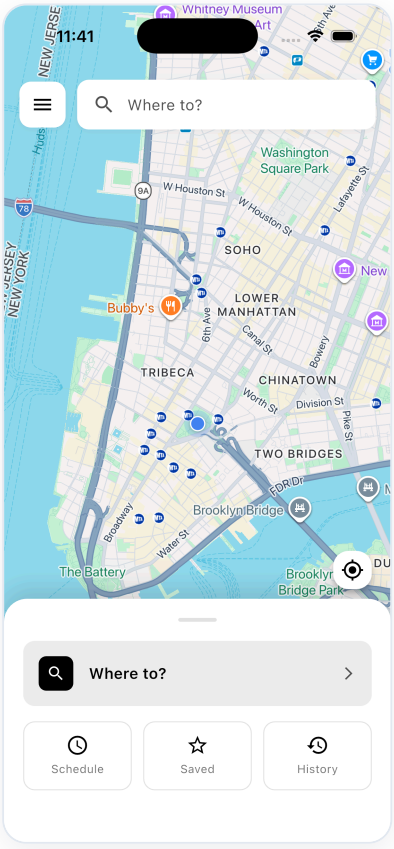
Rider phone login



Fare estimate and vehicle selection



Live ride tracking while matching drivers



Post-trip rating sheet

Driver Experience

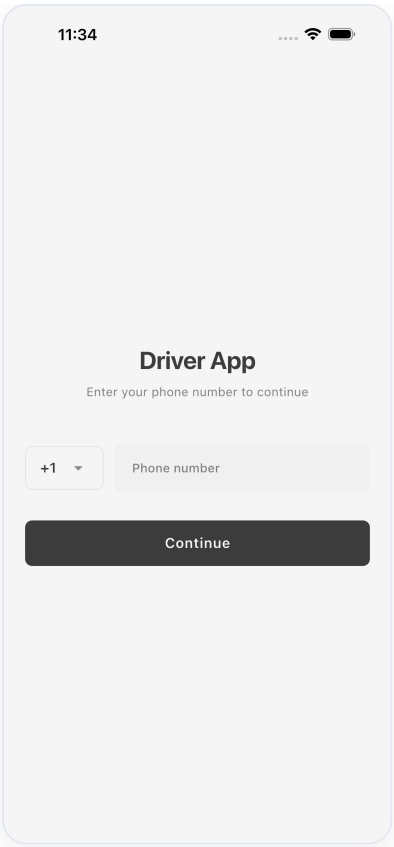
The Driver App follows Clean Architecture with feature-based modules. Drivers must complete document onboarding and receive admin approval before going online.

Key Capabilities

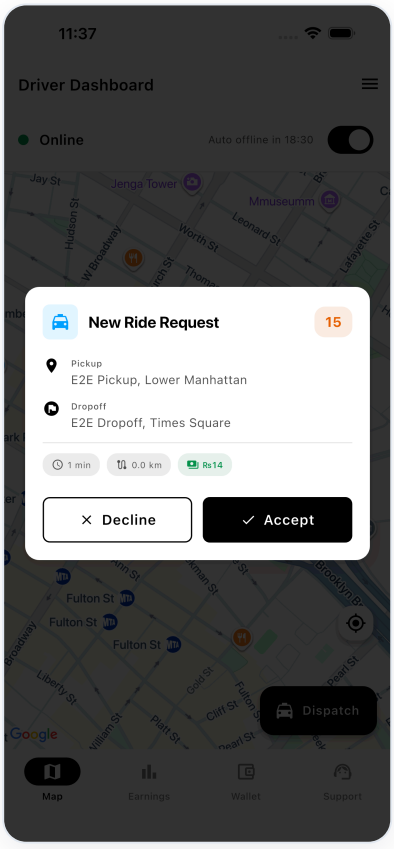
1. **OTP login** with driver-specific session handling
2. **Document onboarding** with upload and compliance tracking
3. **Online/offline toggle** gated on approval status
4. **Ride offer dispatch** with countdown, fare preview, and surge indicator
5. **Trip lifecycle management**: arrive → start → complete
6. **Background location updates** to Redis for matching and live map
7. **Earnings dashboard** with day/week summaries
8. **Safety center & support** mirroring rider-side workflows
9. **In-ride chat** during active trips

User Journey

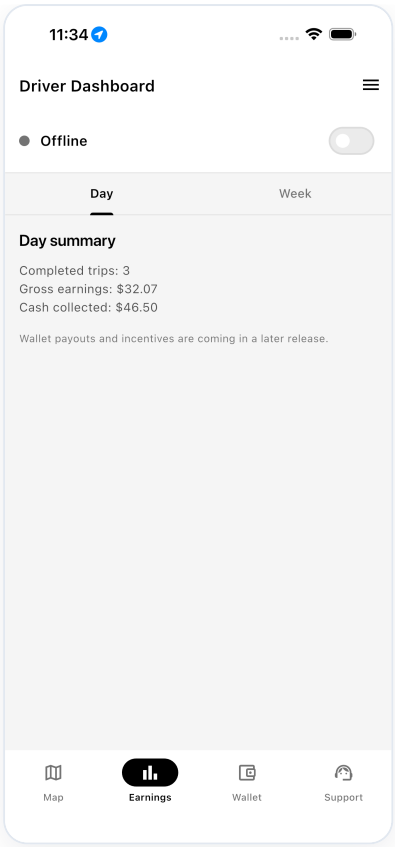
1. Register and submit onboarding documents
2. Await admin approval in the dashboard
3. Go online and receive ride offers
4. Accept offer, navigate to pickup, start and complete trip
5. Review earnings and compliance status



Driver phone login



Incoming ride offer dispatch sheet



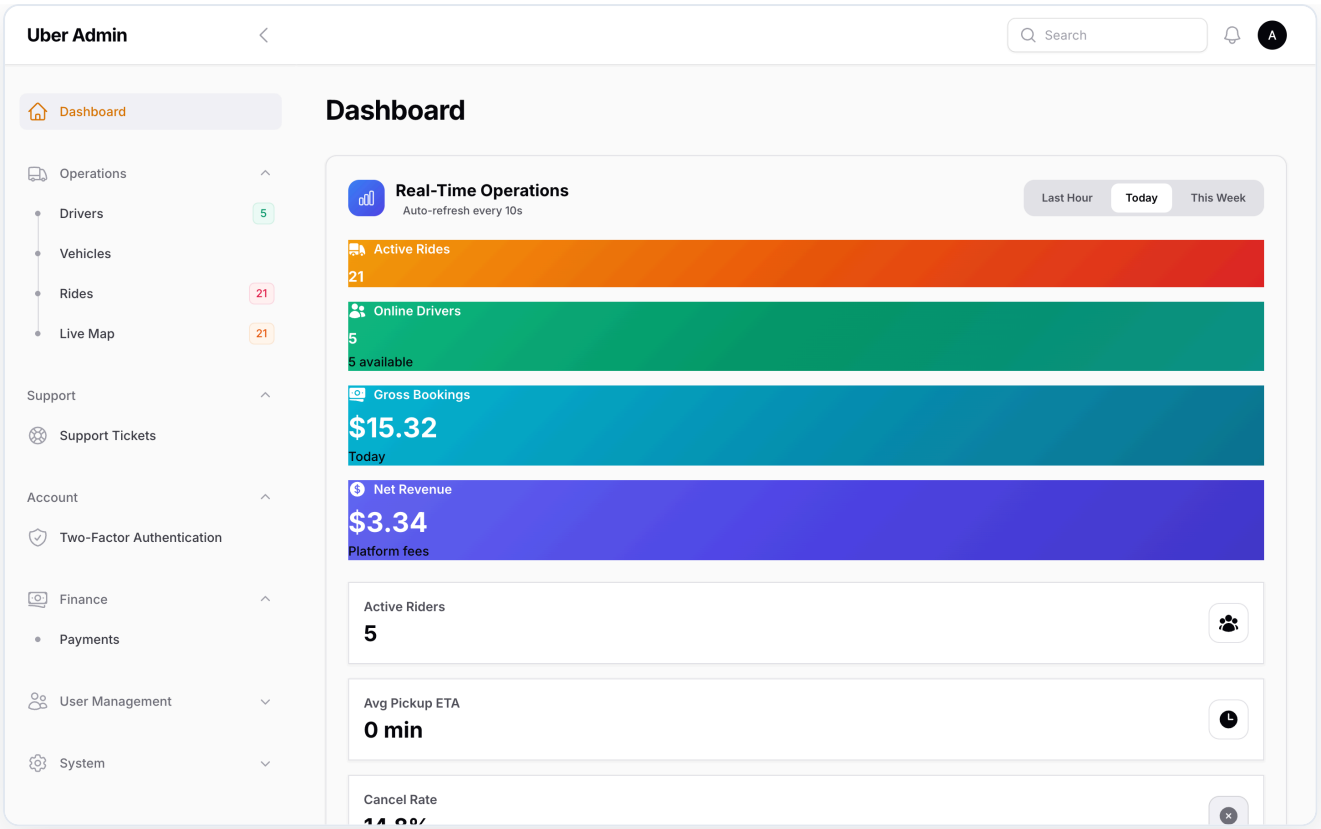
Driver earnings dashboard

Operations Dashboard

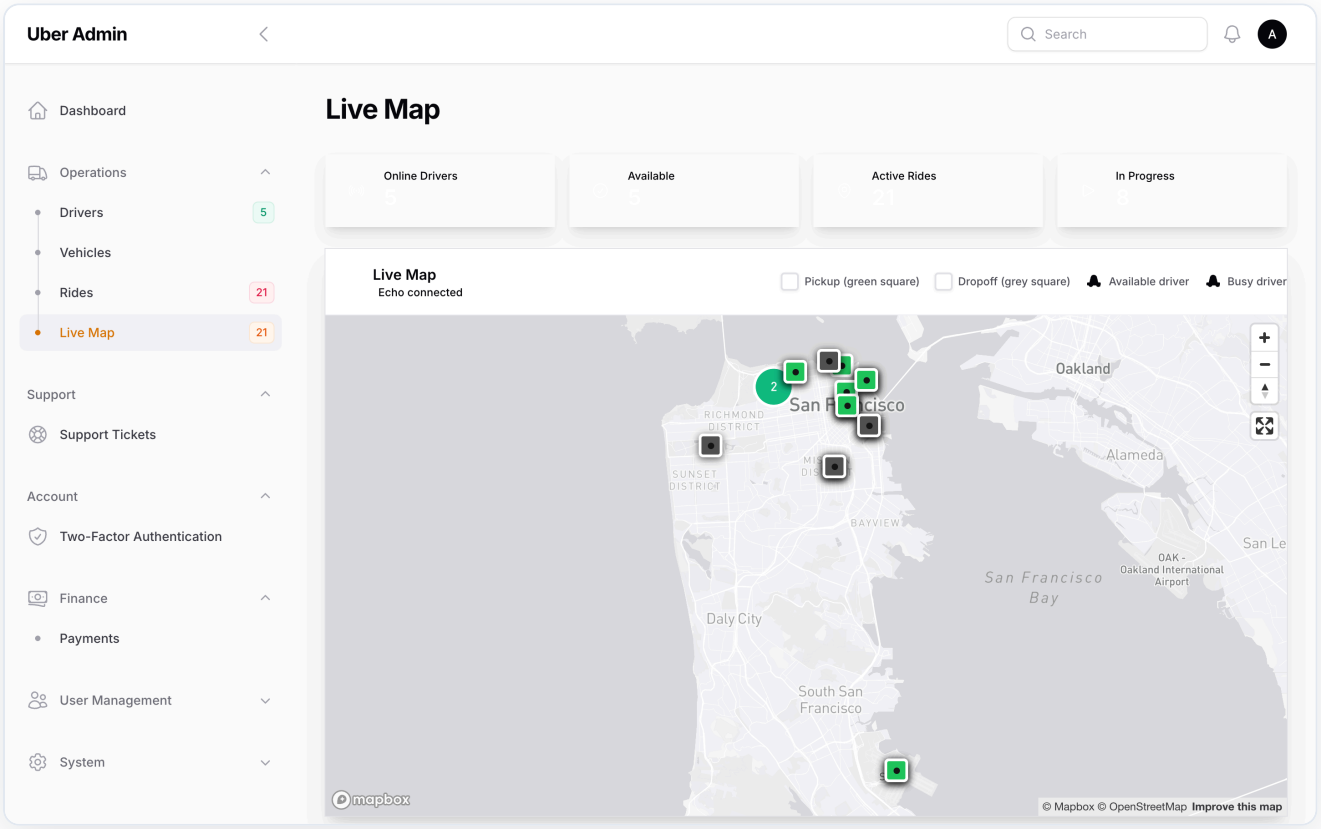
The admin panel is built with Filament v3 on Laravel 11, providing a modern web interface for platform management.

Key Capabilities

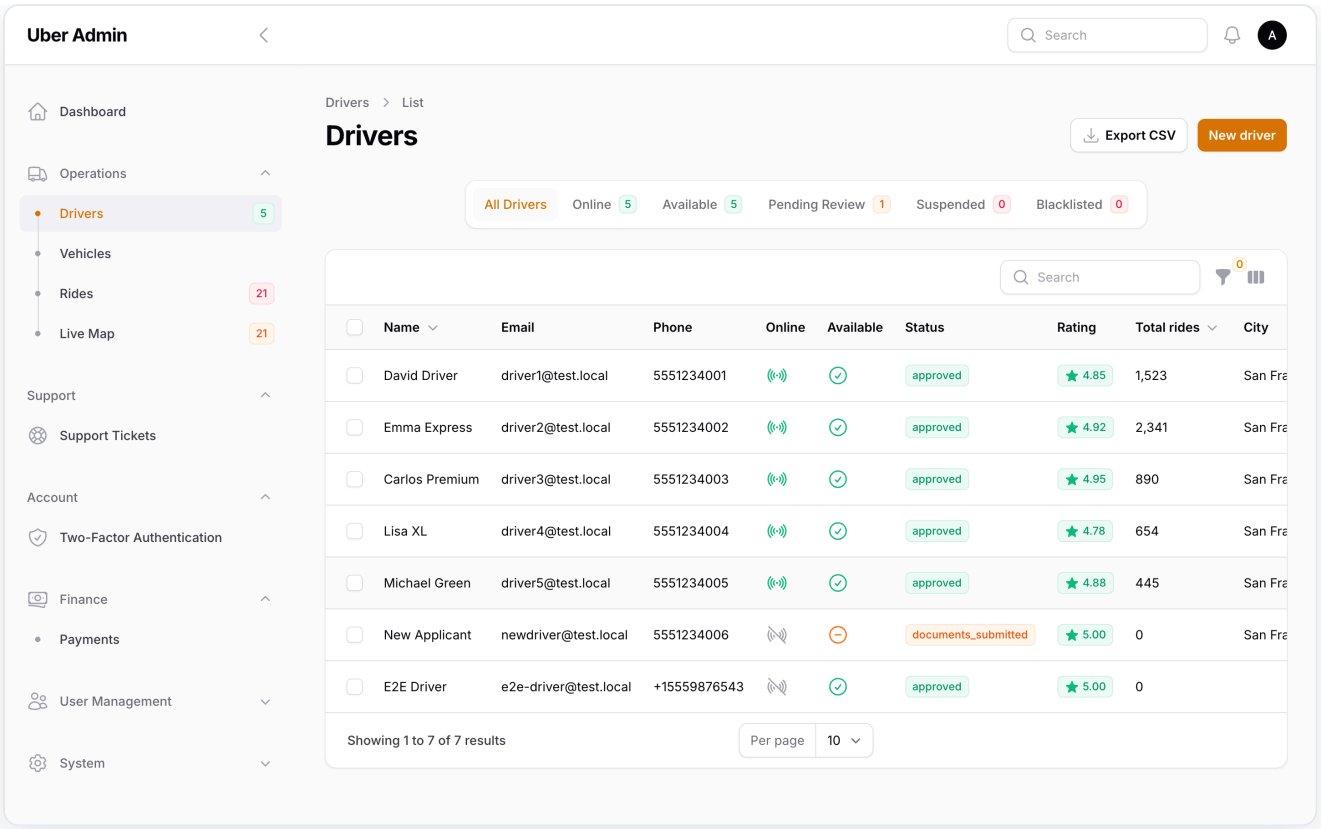
1. **Role-based access control** — Admin, Ops, and Support roles via Spatie Permissions
2. **Live Map** — Real-time driver locations and active rides (Laravel Echo + Reverb)
3. **Driver management** — Approve documents, view vehicles, blacklist accounts
4. **Ride management** — Cancel, reassign, force-complete with full event audit trail
5. **Payment oversight** — Fare breakdown, exports, revenue analytics
6. **Support tickets** — SOS alerts and customer support workflow
7. **Surge pricing controls** — Toggle and set multiplier from Quick Actions widget
8. **Analytics widgets** — KPIs, revenue charts, demand/supply, utilization metrics
9. **Two-factor authentication** for admin accounts
10. **Audit logs** — Compliance-grade activity tracking



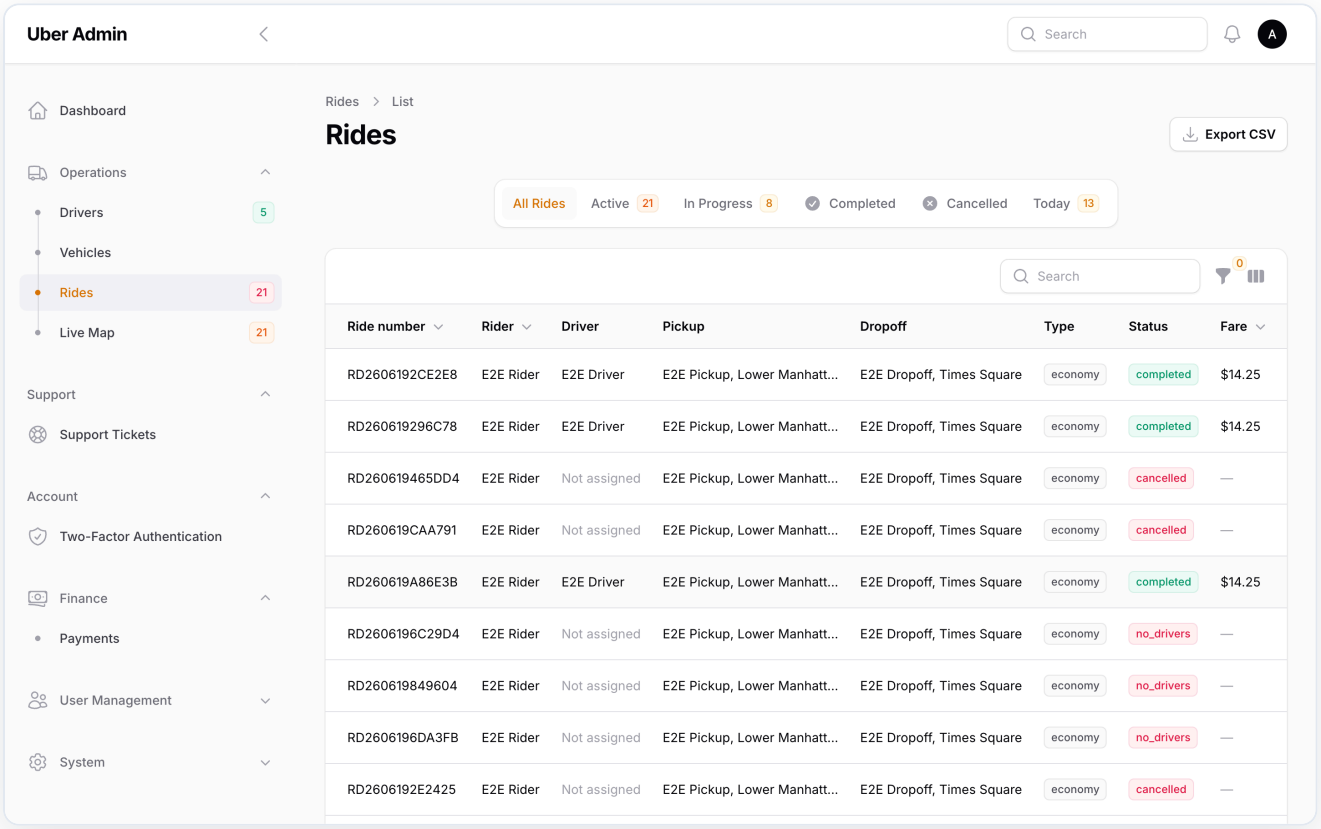
Admin dashboard overview



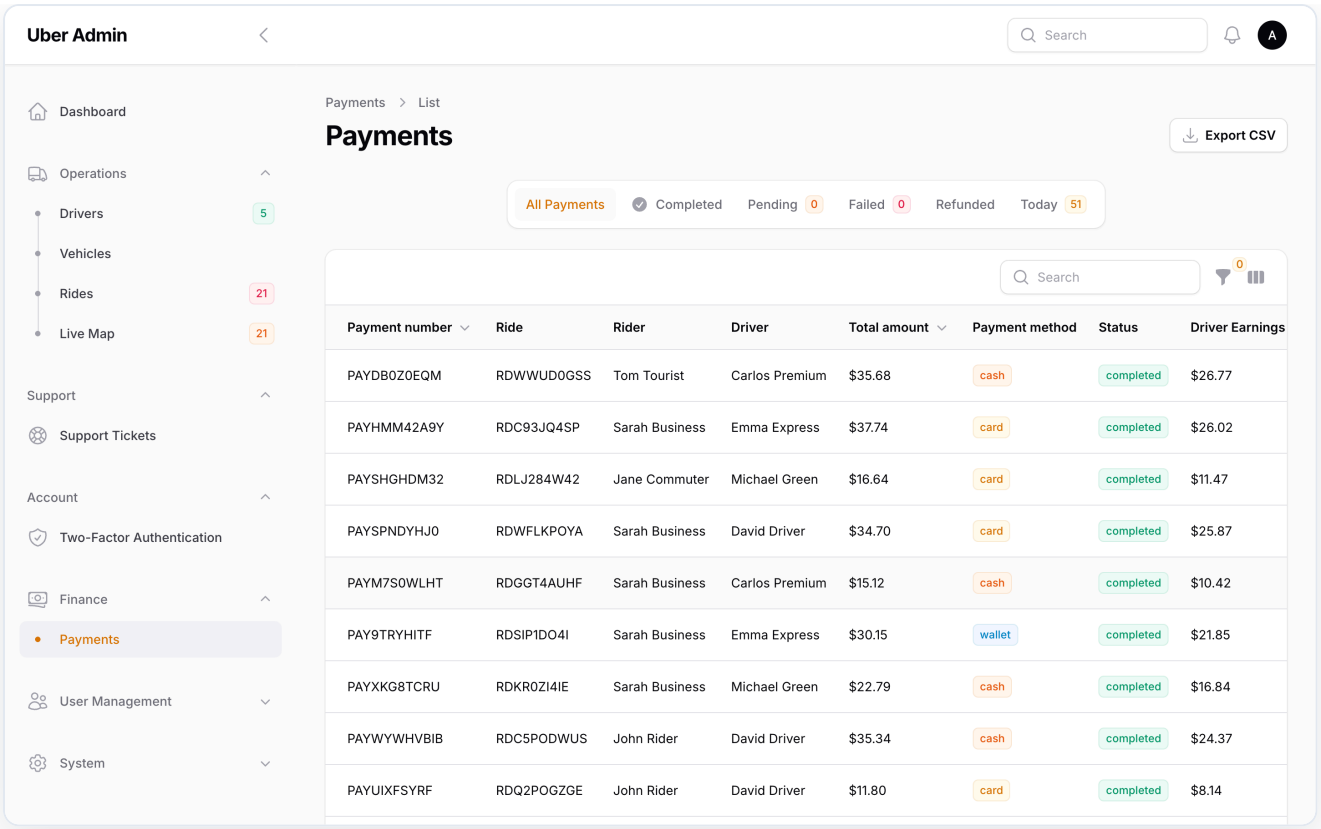
Live operations map



Driver fleet management



Active rides table



Payment records

Technical Architecture

Technology Stack

Layer	Technologies
Rider & Driver Apps	Flutter 3.9+, Riverpod, go_router, Dio, Google Maps, Firebase FCM
Backend API	Laravel 11, PHP 8.2+, Sanctum, REST API v1
Admin Panel	Filament v3, Livewire, Tailwind CSS, Vite
Database	PostgreSQL 16 + PostGIS
Cache & Geo	Redis 7 (GEOADD, online/available sets)
Queues	Laravel Horizon
Real-time	Laravel Reverb (WebSockets), private channels per ride/driver
Infrastructure	Docker Compose (Nginx, PHP-FPM, Postgres, Redis, Horizon, Reverb, Scheduler)
CI/CD	GitHub Actions, GitLab CI

Communication Flow

Mobile apps send authenticated HTTPS requests to the Laravel API. High-frequency driver location updates are throttled and written to Redis with broadcast events. Ride status changes emit WebSocket events to subscribed riders, drivers, and the admin live map. Push notifications are queued through Horizon when Firebase is configured.

Data Model Highlights

Core entities include Users, Drivers, Vehicles, Rides, RideEvents, RideOffers, Payments, DriverLocations, SupportTickets, and SosAlerts. Rides carry full fare breakdown (base, distance, time, surge, tips, discounts) and support six vehicle types: Economy, Comfort, Premium, XL, SUV, and Green.

Ride status flow:

`requested → searching → driver_assigned → driver_arriving → arrived → in_progress → completed`

Driver approval flow:

`pending → documents_submitted → under_review → approved`

Security & Compliance

1. **Mobile auth:** Phone OTP with Laravel Sanctum bearer tokens; secure storage on device
2. **Admin auth:** Email/password with optional Google 2FA
3. **Authorization:** Spatie roles (Admin, Ops, Support) with policy-based Filament access
4. **Channel security:** Private WebSocket channels authorized per ride/driver membership
5. **Safety:** SOS triggers create auditable support tickets visible to operations
6. **Audit trail:** RideEvents log every status change; AuditLog resource for admin actions
7. **Account controls:** Blacklist actions for users and drivers; document verification gating

---

Scalability & Infrastructure

1. **Dockerized deployment** with separate containers for app, web server, database, cache, queue worker, WebSocket server, and scheduler
2. **Redis geospatial indexing** for nearby-driver queries without heavy database load
3. **Horizon-managed job queues** for notifications, matching, and async processing
4. **PostGIS** for geographic data and route-related queries
5. **API versioning** (`/api/v1`) for backward-compatible evolution
6. **Dev/prod flavors** on mobile with environment-based configuration

---

Quality Assurance

The platform includes automated backend feature tests covering:

1. Full ride lifecycle with cash payment and earnings
2. Driver onboarding and go-online approval gate
3. Safety/SOS, chat, and support ticket creation
4. Surge pricing in fare estimates
5. Push notification service (mocked HTTP)
6. API health endpoint

An end-to-end verification checklist validates rider, driver, and admin flows together. Maestro device tests cover OTP login and full ride lifecycle on dual iOS simulators. The rider app includes unit, widget, and golden UI tests.

Capabilities Matrix

Capability	Rider App	Driver App	Dashboard
Phone OTP login	✓	✓	—
Profile management	✓	✓	✓
Map & location	✓	✓	✓ (Live Map)
Ride booking	✓	—	—
Driver matching	✓	✓	✓
Trip lifecycle	✓	✓	✓
In-ride chat	✓	✓	—
SOS / safety	✓	✓	✓
Payments & receipts	✓	—	✓
Earnings	—	✓	✓
Document onboarding	—	✓	✓
Surge pricing	✓	✓	✓
Support tickets	SOS only	✓	✓
Push notifications	✓	✓	—
Role-based admin	—	—	✓
Analytics & exports	—	—	✓
Real-time WebSockets	✓	✓	✓

Extensibility & Roadmap

The architecture supports future enhancements without structural changes:

1. **Payment gateways** — Payment methods API and wallet UI are in place; card processors can be integrated
2. **Scheduled rides** — Ride schema includes `is_scheduled` and `scheduled_at` fields
3. **Additional vehicle types** — Configurable via backend vehicle type definitions
4. **Map provider swap** — Rider app abstracts map provider behind an interface
5. **Multi-language expansion** — Flutter l10n infrastructure with English/Urdu baseline

6. **White-label branding** — Separate mobile flavors and admin theming

---

Conclusion

RideShare Platform delivers a complete, integrated ride-hailing solution ready for client deployment and customization. Three purpose-built applications share a robust backend, enabling seamless coordination between passengers, drivers, and operations teams. With real-time tracking, comprehensive admin tooling, and a security-first design, the platform provides a strong foundation for launching and scaling a modern mobility service.

For technical documentation, API reference, and deployment guides, see the project README files and `uber-admin-dashboard/docs/API.md`.