# 🚀 High-level modules (ek nazar)

1. Auth & Profiles (User + Driver + Admin)
2. Rides (request → counter-offer → confirm → lifecycle)
3. Parcels / Courier
4. Service Centers
5. Wallets, Payouts, Loans (driver-loan system)
6. KYC & Documents
7. Real-time (WebSockets) & Notifications
8. SOS & Safety
9. Admin Panel & Reports

# 1) Authentication & Common

**Auth types:** OTP (mobile), Email+Password (optional), JWT tokens (access + refresh), Role-based middleware (user, driver, admin).

## Endpoints (Auth)

```
POST /api/v1/auth/send-otp              { phone } -> { otp_sent=true }
POST /api/v1/auth/verify-otp            { phone, otp } -> { access_token, refresh_token,
user_profile }
POST /api/v1/auth/login                 { email, password } -> tokens
POST /api/v1/auth/refresh               { refresh_token } -> new tokens
POST /api/v1/auth/logout                (auth) -> invalidate token
GET  /api/v1/auth/me                    (auth) -> profile
PUT  /api/v1/auth/update-profile        (auth) -> update profile fields + avatar
```

**Middleware:** `auth:api`, `role:user/driver/admin`, `kycCompleted` (for driver-specific routes if needed).

## 2) Users (Rider) — core endpoints

```
POST /api/v1/users/register                 { name, phone, email? }
GET  /api/v1/users/:id/profile              (auth) -> profile
PUT  /api/v1/users/:id/profile              (auth) -> update
GET  /api/v1/users/:id/history              (auth) -> rides & parcels
history
POST /api/v1/users/:id/wallet/topup         (auth) { amount, method } ->
initiate payment
GET  /api/v1/users/:id/wallet               (auth) -> balance,
transactions
POST /api/v1/users/:id/sos                  (auth) { location, message?
} -> creates SOS alert
POST /api/v1/users/:id/reviews              (auth) { ride_id, rating,
comment }
```

## 3) Drivers — core endpoints

```
POST /api/v1/drivers/register               { name, phone, email,
vehicle_type, vehicle_no }
GET  /api/v1/drivers/:id/profile            (auth) -> profile + status
PUT  /api/v1/drivers/:id/profile            (auth) -> update +
availability toggle
POST /api/v1/drivers/:id/location           (auth) { lat, lng, bearing }
-> real-time (also via socket)
GET  /api/v1/drivers/:id/earnings           (auth) -> wallet, payouts
POST /api/v1/drivers/:id/documents/upload (auth) { doc_type, file }
POST /api/v1/drivers/:id/kyc/submit         (auth) -> start KYC
GET  /api/v1/drivers/:id/apply-loan         (auth) { amount, term,
reason } -> create loan application
```

```
GET  /api/v1/drivers/:id/loans             (auth) -> loan history / EMI
status
POST /api/v1/drivers/:id/sos               (auth) { location, message?
} -> SOS
POST /api/v1/drivers/:id/availability      (auth) { status:
online|offline|busy }
```

## 4) Ride Flow APIs (most important: request + counter-offer)

### Create ride request (user)

```
POST /api/v1/rides/request
Headers: Authorization: Bearer <token>
Body: {
  user_id, pickup: {lat,lng,address}, drop: {lat,lng,address},
  vehicle_type, offered_price, estimated_fare, schedule_at? (null =>
now),
  meta: { is_parcel:boolean, parcel_id? }
}
Response: { request_id, status: pending, matched_drivers_count }
```

### Server matches nearby drivers and broadcasts the request via websocket or push.

### Driver receives request (via socket/push) and can:

- Accept user price
- Counter-offer (driver_price)
- Decline

### Driver actions endpoints (for API fallback / audit)

POST /api/v1/rides/:request_id/driver-action
Headers: Driver auth
Body: { action: accept|counter|decline, driver_price? }
Response: { action_id, status }

### User accepts driver counter-offer

```
POST /api/v1/rides/:request_id/user-action
Headers: User auth
Body: { action: accept_counter|cancel|confirm, driver_id? }
```

### When confirmed:

- Create `ride` record, assign driver, start lifecycle:

```
GET  /api/v1/rides/:ride_id/status          (auth) -> current status
POST /api/v1/rides/:ride_id/start           (driver) -> status:
started
POST /api/v1/rides/:ride_id/arrive          (driver) -> status:
arrived
POST /api/v1/rides/:ride_id/complete        (driver) { fare, distance,
time } -> finalize
POST /api/v1/rides/:ride_id/cancel          (user|driver) { reason }
```

**Important:** Payment capture can be pre-authorized or post-pay:

```
POST /api/v1/payments/charge               (user) { ride_id, method }
-> receipt
POST /api/v1/payments/refund               (admin) { payment_id }
```

# 5) Parcels / Courier

```
POST /api/v1/parcels/create
Body: { sender_id, pickup:{...}, drop:{...}, weight, dimensions, type,
price_offer? }
GET  /api/v1/parcels/:id/status
POST /api/v1/parcels/:id/assign-driver    (driver accepts)
POST /api/v1/parcels/:id/pod              { signature_image, photo,
receiver_name }
```

---

## 6) Service Centers

```
POST /api/v1/service-centers/register     { name, address, coords,
services, docs }
GET  /api/v1/service-centers/nearby       { lat, lng, radius }
PUT  /api/v1/service-centers/:id/update   (admin or center auth)
POST /api/v1/service-centers/:id/offers   { driver_discount,
coupon_code }
GET  /api/v1/service-centers/:id/history
```

---

## 7) Wallets, Payouts & Loan

**Wallets**

```
GET  /api/v1/wallets/:owner_id
POST /api/v1/wallets/:owner_id/topup      { amount, txn_ref }
POST /api/v1/wallets/:owner_id/transfer   { to_user_id, amount } //
admin-controlled
```

**Payouts**

```
POST /api/v1/payouts/request               (driver) { amount,
bank_details_id }
```

```
GET  /api/v1/payouts/history                (driver)
POST /api/v1/payouts/process                (admin) { payout_id } //
triggers external bank/P
```

**Loan app & EMI**

```
POST /api/v1/loans/apply                    { driver_id, amount,
term_months, reason }
GET  /api/v1/loans/:loan_id                 -> loan details +
amortization schedule
POST /api/v1/loans/:loan_id/approve         (admin) -> create EMI
schedule, credit wallet
POST /api/v1/loans/:loan_id/reject          (admin)
```

## 8) KYC Flow (User & Driver)

**Driver KYC (strict)**:

- Required docs: Govt ID (Aadhaar / Passport / DL), Driver License (DL), Vehicle RC, Vehicle Insurance, Police Verification (optional), Selfie with ID.
- Steps:
    1. Upload docs via `/drivers/:id/documents/upload`
    2. System runs basic checks (file types, size) and stores as VERIFIED/PENDING/REJECTED.
    3. Admin/automated OCR + manual review updates `driver.kyc_status` = pending/verified/rejected.
    4. Only `kyc_status=verified` + `active_status=online` allow taking passengers (or allow limited operations until verified).

**User KYC (light)**:

- Optional: Govt ID for higher limits (payments > X), for parcel insurance.
- Upload via `/users/:id/documents/upload`

**KYC endpoints**

```
GET /api/v1/kyc/:entity/:id/status
POST /api/v1/kyc/:entity/:id/upload-doc  { doc_type, file }
POST /api/v1/kyc/:entity/:id/submit       -> mark submitted
PUT  /api/v1/kyc/:entity/:id/review       (admin) { status, remarks }
```

## 9) SOS & Safety

```
POST /api/v1/sos/trigger     (user/driver) { location, ride_id?,
message }
GET  /api/v1/sos/:id         (admin) -> details
POST /api/v1/sos/:id/respond (admin) { action:
alert_police|notify_drivers|dispatch }
```

On SOS:

- Notify nearest drivers (within X km)
- Notify admin with live tracking
- Optionally integrate with third-party emergency APIs or local police endpoints (if available).

---

## 10) Real-time (WebSockets) Events (recommended: Laravel WebSockets or Pusher)

**Socket channels:**

- `private-user.{user_id}` — for user-specific push (request updates, driver location)
- `private-driver.{driver_id}` — for driver-specific requests
- `public-nearby.{latlng-cell}` — optional geo-clustering

**Events:**

- `RideRequestCreated` -> sent to nearby drivers
- `DriverCounterOffer` -> sent to user
- `RideConfirmed` -> both
- `DriverLocationUpdate` -> user & admin
- `SOSAlert` -> admin & nearby drivers
- `ParcelUpdate` -> user & assigned driver
- `ServiceCenterOffer` -> driver

**Payload sample (DriverCounterOffer):**

```
{
  "request_id": "req_123",
  "driver_id": 45,
  "driver_price": 350,
  "eta_seconds": 120,
  "driver_name": "Rahul",
  "vehicle_no": "DL1AB1234"
}
```

# 11) Database Structure (main tables + key columns)

Use InnoDB, utf8mb4. Add proper indexes on frequently searched fields (lat/lng for geosearch via spatial index or use geohash).

## users

- id, name, phone (unique), email, password, role ENUM(user,driver,admin), avatar, created_at, updated_at, last_seen, kyc_status ENUM

- indexes: phone, email

## drivers

- id (FK users.id), vehicle_type, vehicle_no, vehicle_rc_url, insurance_url, license_url, is_available (bool), rating_avg, total_rides, joined_at, status ENUM(active,banned,suspended)

- indexes: is_available, vehicle_type

## driver_documents

- id, driver_id, doc_type, file_url, status, reviewed_by, reviewed_at, created_at

## rides

- id, request_id (temp), user_id, driver_id, pickup_lat, pickup_lng, pickup_addr, drop_lat, drop_lng, drop_addr, status ENUM(pending,offered,accepted,arrived,started,completed,cancelled), offered_price, driver_price, final_price, distance_km, duration_min, started_at, completed_at, created_at

- indexes: user_id, driver_id, status, created_at, spatial (pickup_lat,pickup_lng)

## ride_offers (for counter-offers & history)

- id, ride_id, request_id, driver_id, offered_price, accepted_by (user|driver|null), action, created_at

## parcels

- id, sender_id, pickup_, *drop_*, weight, dimensions, parcel_type, assigned_driver_id, status, pod_url, price, created_at

### service_centers

- id, name, manager_name, phone, address, lat, lng, docs, approved (bool), ratings

### wallets

- id, owner_type ENUM(user,driver), owner_id, balance_decimal

### wallet_transactions

- id, wallet_id, type ENUM(credit,debit), amount, meta(json), created_at

### loans

- id, driver_id, amount, term_months, interest_rate, emi_amount, outstanding_amount, status ENUM(pending,approved,rejected,completed), created_at

### payouts

- id, driver_id, amount, method, status, requested_at, processed_at

### notifications

- id, user_id, title, body, data(json), read_at, created_at

### sos_alerts

- id, user_id|driver_id, location_lat, location_lng, ride_id?, status, responded_by, response_notes, created_at

### audits / logs

- ride_actions, payment_logs, kyc_audit

---

## 12) Smart Roadmap for Implementation (phases)

**Phase 0 — Planning & infra**

- Finalize models & API contract (we're doing that now)
- Choose hosting: AWS (RDS, ECS/EC2), Redis, WebSockets (ElastiCache + WebSocket server)
- Set up CI/CD, code style, env secrets

**Phase 1 — Core MVP**

- Auth, basic user & driver profiles
- Ride request → driver accept flow (no counter-offers initially) + WebSockets
- Basic payments (Razorpay/Stripe) – capture after ride
- Basic admin panel (users, drivers, rides)

**Phase 2 — Counter-offer + Parcel**

- Implement counter-offer logic, ride_offers table
- Parcel create/assign flow
- Driver wallet & payouts

**Phase 3 — Service Centers + KYC + SOS**

- Service centers registration & listing
- KYC flows + admin document review
- SOS + nearest-driver notification

**Phase 4 — Loans, Analytics & Scaling**

- Driver loan eligibility & application
- EMI schedule + auto deductions
- Reporting dashboards, fraud detection, ML based surge

**Phase 5 — Polishing**

- Multi-language, maps optimizations, payments settlement integration with NBFCs, driver incentives.

---

## 13) Key Business Rules & Edge Cases

- **Timeouts**: Driver counter-offer window (e.g., 30s) — if no response, move to next driver.
- **Concurrent offers**: Use `ride_offers` with optimistic locking; first accepted wins.
- **Location stale**: If driver location > 60s old, mark unavailable.
- **Cancel penalties**: cancellation fee based on when and by whom.
- **Fraud detection**: flag users/drivers with repeated cancellations or low ratings.
- **Driver eligibility for loan**: scheduled job to check `joined_at <= now - 6 months` && rides_count >= 200 && rating >= 4.0 && no fraud flags.

---

## 14) Cron Jobs & Scheduler

- `daily:check_loan_eligibility` — mark drivers as eligible
- `cron:deduct_emi` — run weekly/monthly to deduct EMI from wallet; create dues if insufficient.

- `cron:cleanup_stale_requests` — expire pending requests older than X seconds
- `cron:reconcile_payments` — reconcile gateway payments
- `cron:generate_reports` — generate daily summary for admin

---

## 15) Admin Panel — navigation routes & features

Design as Filament / Nova / custom admin SPA.

**Top Nav / Sidebar**

- Dashboard (overview: active rides, SOS, earnings)
- Users
  - All Users
  - KYC Pending
- Drivers
  - All Drivers
  - KYC Pending
  - Loan Applications
  - Wallet & Payouts
- Rides
  - Active Rides
  - Past Rides
  - Cancellations

- Parcels
    - Active Parcels
    - PODs
- Service Centers
    - Approvals
    - Offers & Coupons
- SOS
    - Active Alerts
    - History
- Payments
    - Transactions
    - Reconciliation
    - Refunds
- Reports
    - Daily Summary
    - Driver Performance
    - Parcel Metrics
- Settings
    - Fare Config (base fare, per km, surge)
    - Vehicle Types
    - Notification Templates
    - Integrations (Payment, NBFC)

- Support / Tickets
- Logs / Audit

**Admin API endpoints (examples)**

```
GET /api/v1/admin/dashboard
GET /api/v1/admin/drivers?filter=kyc_pending
POST /api/v1/admin/drivers/:id/kyc-verify { status, note }
GET /api/v1/admin/loans
POST /api/v1/admin/loans/:id/approve
POST /api/v1/admin/sos/:id/respond
POST /api/v1/admin/settings/fare
```

## 16) Security & Compliance (important)

- Use HTTPS everywhere, secure JWT secret rotation.
- Rate-limit API per IP & per user.
- Store files (docs, POD) on S3 with signed URLs.
- Encrypt sensitive data at rest (PII like phone/email optional).
- Audit logs for financial operations.
- KYC docs retention policy (compliance).
- Implement 2FA for admin.

---

## 17) Sample payload flows (quick)

**User creates ride:**

```
POST /api/v1/rides/request
{
 "user_id": 11,
 "pickup": {"lat":28.7041,"lng":77.1025,"address":"Connaught Place"},
 "drop": {"lat":28.5355,"lng":77.3910,"address":"Noida Sector 62"},
 "vehicle_type":"car",
 "offered_price": 400,
 "estimated_fare": 450
}
```

**Driver counter-offer (socket event or API):**

```
{
 "type":"DriverCounterOffer",
 "data":{
   "request_id":"req_123",
   "driver_id":45,
   "driver_price":350,
   "eta":120
 }
}
```

**User accepts driver price:**

```
POST /api/v1/rides/:request_id/user-action
{ action: "accept_counter", driver_id: 45 }
```

---

## 18) Dev Notes & Implementation Tips

- Use Laravel Policies and Gates to limit resource access.
- Use Redis for presence and fast location caching; consider clustering drivers using geohash for fast nearby searches.
- Keep WebSocket events idempotent and use event bus for retries.
- Keep business logic in services not controllers (clean architecture).

- Use queue workers (Redis/Beanstalkd) for notifications, payment verification, heavy tasks.
- Add feature flags for toggling experimental features (counter-offers, loans) per region.

---

## 19) Deliverables I can produce next (pick any, I’ll do immediately)

- Full migration files (Laravel) for the tables above.
- Example Controllers + Services (Laravel) for Ride Request & Counter-offer flow.
- WebSocket events stubs (Laravel WebSockets) + client socket usage (React Native).
- Admin Filament resource definitions.
- Postman collection / OpenAPI (Swagger) for all APIs.

Tables All

| Column | Type | Notes |
|---|---|---|
| id | bigint | PK |
| name | string | |
| email | string | nullable, unique |
| phone | string | unique |
| password | string | |
| role | enum('user','driver','admin') | define role |

| | | |
|---|---|---|
| kyc_status | enum('pending','approved','rejected') | |
| kyc_docs | json | store Aadhaar, PAN, License URLs |
| is_verified | boolean | OTP verified |
| wallet_balance | decimal(10,2) | default 0 |
| rating | decimal(3,2) | default 0 |
| experience_months | int | for driver loan |
| status | enum('active','blocked') | |
| created_at | timestamp | |
| updated_at | timestamp | |

### 🚗 vehicles

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| user_id | int | FK(users.id) |
| vehicle_type | enum('car','bike','auto') | |
| vehicle_number | string | |
| docs | json | RC, insurance, pollution cert |
| status | enum('approved','pending') | |

### 💰 wallets

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| user_id | int | FK(users) |
| balance | decimal(10,2) | |

| | | |
|---|---|---|
| min_limit | decimal(10,2) | e.g. -100 |
| last_updated | timestamp | |

### 💸 wallet_transactions

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| wallet_id | int | FK(wallets) |
| type | enum('credit','debit') | |
| amount | decimal(10,2) | |
| reason | string | e.g. "ride fare", "commission", "loan EMI" |
| reference_id | int | ride_id or loan_id |
| created_at | timestamp | |

### 🚕 rides

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| user_id | int | FK(users) |
| driver_id | int | FK(users) |
| pickup_location | string | |
| drop_location | string | |
| fare | decimal(10,2) | |
| user_price | decimal(10,2) | |

| | | |
|---|---|---|
| driver_offer | decimal(10,2) | |
| final_price | decimal(10,2) | |
| status | enum('requested','countered','confirmed','completed','cancelled') | |
| payment_method | enum('cash','wallet','online') | |
| commission | decimal(10,2) | |
| sos_triggered | boolean | default 0 |
| created_at | timestamp | |

### 🆘 sos_alerts

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| ride_id | int | FK(rides) |
| user_id | int | FK(users) |
| lat | decimal(10,6) | |
| lng | decimal(10,6) | |
| nearest_cabs | json | list of nearby drivers |
| resolved_by_admin | boolean | |
| created_at | timestamp | |

### 🏦 loans

| Column | Type | Notes |
|---|---|---|

| | | |
|---|---|---|
| id | int | PK |
| user_id | int | FK(users.id) |
| amount | decimal(10,2) | |
| emi_amount | decimal(10,2) | |
| total_emis | int | |
| remaining_emis | int | |
| status | enum('active','completed','defaulted') | |
| next_due_date | date | |

## 🧰 service_centers

| Column | Type | Notes |
|---|---|---|
| id | int | PK |
| name | string | |
| location | string | |
| services | json | ["oil change","tyre","wash"] |
| discount_rate | int | e.g. 10% for registered cabs |
| contact | string | |
| approved | boolean | |

## USER FLOW & FEATURES

### 🔹 Step 1: Registration / Login

- User registers via **phone + OTP**
- Optional email/password
- After OTP verification → wallet auto-create

### 🔹 Step 2: KYC (Optional for User)

- Upload **Aadhaar / PAN / ID**
- Status: pending → approved → rejected
- Admin verifies KYC

### 🔹 Step 3: Book a Ride / Parcel

- User selects **pickup & drop location**
- Chooses **service type**: Cab / Parcel / Courier
- App sends **ride request to nearby drivers**

### 🔹 Step 4: Price & Driver Offer

- Drivers can **send counter-offer**
- User sees driver price & confirms
- Once user confirms → ride status becomes **confirmed**

### 🔹 Step 5: Payment

- Options: **Cash / Wallet / Online**
- Wallet auto-deducts fare

- Commission & tax handled automatically

### 🔹 Step 6: Ride Tracking & SOS

- Live **driver tracking** on map
- **SOS button** → nearest drivers + admin notified
- After ride → user can rate driver

### 🔹 Step 7: Wallet & Transaction History

- Check wallet balance
- Top-up wallet via Razorpay/Stripe
- View **wallet transaction history**

---

## 2️⃣ DRIVER FLOW & FEATURES

### 🔹 Step 1: Registration / Login

- Driver registers via **phone + OTP**
- Upload **vehicle documents** + KYC
- Admin approves account

### 🔹 Step 2: Vehicle Registration

- Add **vehicle type & number**
- Upload **RC, Insurance, Pollution certificate**
- Admin approves

### 🔹 Step 3: Ride Offers

- Nearby ride request arrives → driver can **accept or counter-offer price**
- If user confirms → ride status becomes **confirmed**

### 🔹 Step 4: Ride Completion & Wallet

- Complete ride → fare credited to driver wallet **after commission deduction**
- Cash rides → system deducts commission automatically from wallet
- Wallet shows **balance, min_limit, transactions**

### 🔹 Step 5: Loan Eligibility

- Active driver > **6 months** → loan eligibility flag enabled
- Apply for loan → EMI deducted automatically from wallet

### 🔹 Step 6: Service Center

- Driver sees **partner service centers**
- Can book car service → discount applied via wallet
- Maintain service history

### 🔹 Step 7: SOS & Safety

- Respond to nearby SOS alerts
- Admin & user notified in emergency

---

## 3️⃣ ADMIN FLOW & FEATURES

### 🔹 Dashboard

- **View users & drivers**
- Ride summary & revenue stats

## 🔹 Approvals

- Approve/reject **KYC, vehicles, service centers**
- Approve **loan requests**

## 🔹 Wallet Management

- Manual **credit/debit** driver wallet
- Monitor negative balances & limits

## 🔹 Ride Management

- Track all rides in real-time
- Handle disputes

## 🔹 SOS & Alerts

- Resolve SOS
- Notify nearby drivers or authorities

## 🔹 Reports

- Earnings, rides, wallet stats, service usage

## // Main Model

**Indrive Model:**

- Indrive me users aur drivers **fare negotiate** karte hain (counter offer system).
- User mostly **cash** me pay karta hai → to Indrive wallet me sirf **service fee** hoti hai.
- Driver ke wallet se har ride pe **Indrive fee (say ₹20)** deduct hoti hai.
- Jab wallet me balance kam ho jata hai (below min threshold, say ₹-100), driver **new ride accept nahi kar sakta** until top-up.
- Driver **UPI ya card** se wallet top-up karta hai (minimum ₹200, ₹500, etc.)

🧠 Example:

Ride ₹400 (user pays cash to driver)
Indrive fee ₹20 → wallet -20
Driver rides 5 trips → wallet -100
Indrive blocks further rides → driver adds ₹500 → wallet +400.

### 🔁3️⃣Why It’s Used

| Reason | Explanation |
|---|---|
| 💸 Commission Handling | Company apna % cut easily auto-adjust kar sake |
| ⚙️ Cash Ride Settlement | Cash me payment aane ke baad backend me settlement karna easy ho |
| 🚫 Ride Blocking | Wallet limit low hone pe ride block karne ka system |
| 📊 Loan/EMI Adjustments | Loan EMI wallet se auto-deduct |
| 💰 Transparency | Driver ko earning, dues aur bonuses ka clear view milta hai |
| 🔔 Bonus & Refund | Referral ya promo bonus wallet me credit hota hai |