



React Admin Dashboard - API Integration Guide

Target Audience: Web Frontend Team

Base URL (Staging): `https://driversklub-backend.onrender.com`

Base URL (Development): `http://localhost:3000` (API Gateway)

Base URL (Production): AWS Elastic Beanstalk `driversklub-backend-env`

Auth: Requires `Authorization: Bearer <TOKEN>` with Role `SUPER_ADMIN`, `OPERATIONS`, or `MANAGER`

Version: 4.6.0 (MMT Smart ID + Pricing Cleanup)

Last Updated: January 27, 2026

Last Verified: January 27, 2026

What's New in v4.6.0

- **MMT Smart ID Formatting** - Driver/Vehicle IDs compressed to 10 chars for MMT compliance
- **Pricing Cleanup** - Consolidated pricing config, removed unused rules file
- **Endpoint Fix** - MMT uses `/detach` (not `/unassign`), location uses `/update`

What's New in v4.5.0

- **MMT Integration Complete** - Full inbound + outbound tracking
 - Inbound: search, block, paid, cancel, reschedule
 - Outbound: assign, reassign, detach, start, arrived, boarded, alight, not-boarded, location
 - Automatic `providerBookingId` storage from MMT paid endpoint
- **FLEET_ADMIN Role** - Fleet-level administration with scoped access
- **Public Booking API** - Customers can book trips without auth
- **Referral System** - Driver referral tracking and rewards
- **Enhanced KYC** - Full driver/vehicle document management

Note: All requests route through the API Gateway to 6 microservices. The gateway handles authentication and routing automatically.



Table of Contents

1. [Authentication](#)
2. [Dispatch & Trip Operations](#)
3. [Fleet & Asset Management](#)
4. [Operations & Assignments](#)
5. [User Management](#)
6. [Pricing Calculator](#)
7. [Payment & Financial Management](#)
8. [Frontend Implementation Notes](#)
9. [Rapido Operational Monitoring](#)
10. [Maps Service](#)

1. Authentication

1.1 Admin Login Flow

Same as driver authentication but requires `SUPER_ADMIN` or `OPERATIONS` role.

Endpoint: `POST /auth/verify-otp`

Response:

```
{
  "success": true,
  "data": {
    "accessToken": "eyJ...",
    "refreshToken": "def...",
    "user": {
      "id": "uuid",
      "shortId": "USR20260123001",
      "role": "SUPER_ADMIN"
    }
  }
}
```

Action: Check `user.role`. Redirect to dashboard based on role permissions.

Allowed Admin Roles:

- `SUPER_ADMIN` - Full system access (all fleets)
- `FLEET_ADMIN` - Fleet-level admin (scoped to their fleet)
- `OPERATIONS` - Operations team access
- `MANAGER` - Hub/Fleet manager access

Token Expiry:

- Access Token:** 15 minutes (all clients)
- Refresh Token:** 1 day (web clients default)

2. Dispatch & Trip Operations

2.1 Create New Trip

Endpoint: `POST /trips`

Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "tripType": "AIRPORT",
  "bookingType": "PREBOOK",
  "originCity": "Delhi",
  "destinationCity": "Gurgaon",
  "pickupLocation": "T3 Terminal, Gate 4",
  "pickupLat": 28.5562,
```

```
"pickupLng": 77.1000,
"dropLocation": "Cyber Hub, Gurgaon",
"dropLat": 28.4595,
"dropLng": 77.0266,
"pickupTime": "2025-12-25T10:00:00Z",
"vehicleSku": "TATA_TIGOR_EV",
"distanceKm": 45,
"passengerName": "Amit Sharma",
"passengerPhone": "9876543210"
}
```

Field Details:

- `tripType` : AIRPORT , INTER_CITY , RENTAL
- `bookingType` : PREBOOK (future) or INSTANT (immediate)
- `vehicleSku` : Specific vehicle model (e.g., TATA_TIGOR_EV) or use `vehicleType` ("EV")
- `pickupTime` : ISO Date string. **Must be > 2.5 hours from now** for PREBOOK.
- `passengerName` / `passengerPhone` : Optional if booking for someone else.

```
> [!IMPORTANT]
> **Strict Trip Constraints:**
>
> - **Start Window:** Driver can only start trip **2.5 Hours** before pickup
> - **Geofence:** `pickupLat` & `pickupLng` are **MANDATORY** for the app to allow "Arrived"
status (500m radius)
> - **T-1 Constraint:** `pickupTime` must be > 24 hours from now

**Response (201):**

```json
{
 "success": true,
 "data": {
 "id": "uuid",
 "shortId": "TRP20260123001",
 "status": "CREATED",
 "price": 1200,
 "tripType": "AIRPORT",
 "pickupTime": "2025-12-25T10:00:00Z"
 }
}
```

## 2.2 List All Trips (Grid View)

**Endpoint:** GET /admin/trips

**Role:** SUPER\_ADMIN

#### Query Params:

- `page` (default: 1)
- `limit` (default: 10)

- `status` (optional): Filter by status (e.g., `CREATED` , `DRIVER_ASSIGNED` , `STARTED` )

**Response (200):**

```
{
 "success": true,
 "data": {
 "trips": [
 {
 "id": "uuid",
 "shortId": "TRP20260123001",
 "tripType": "AIRPORT",
 "pickupLocation": "T3 Terminal",
 "dropLocation": "Cyber Hub",
 "pickupTime": "2025-12-25T10:00:00Z",
 "status": "CREATED",
 "price": 1200,
 "driver": null,
 "customerPhone": "9876543210",
 "providerMapping": {
 "providerType": "MMT",
 "externalBookingId": "MMT-123"
 }
 }
],
 "total": 150,
 "page": 1,
 "limit": 10
 }
}
```

### 2.3 Assign Driver (Dispatch)

**Endpoint:** `POST /admin/trips/assign` **Role:** `OPERATIONS` , `MANAGER`

**Request Body:**

```
{
 "tripId": "uuid-trip-id",
 "driverId": "uuid-driver-id"
}
```

**Response (201):**

```
{
 "success": true,
 "message": "Driver assigned successfully",
 "data": {
 "id": "uuid-trip-id",
 "shortId": "TRP20260123001",
 }
}
```

```
"status": "DRIVER_ASSIGNED",
"tripType": "AIRPORT",
"pickupLocation": "T3 Terminal, Gate 4",
"dropLocation": "Cyber Hub, Gurgaon",
"pickupTime": "2025-12-25T10:00:00Z",
"price": 1200,
"driverId": "uuid-driver-id",
"vehicleSku": "TATA_TIGOR_EV"
}
}
```

## 2.4 Reassign Driver

**Endpoint:** POST /admin/trips/reassign **Role:** OPERATIONS , MANAGER

**Use Case:** Switch driver for an already assigned trip (before or during trip).

**Logic:**

- Atomically swaps current driver with new driver
- **For MMT trips:** Sends MMT `reassign` event (NOT unassign + assign)
- Keeps trip status as `DRIVER_ASSIGNED`

**Request Body:**

```
{
 "tripId": "uuid-trip-id",
 "driverId": "uuid-new-driver-id"
}
```

**Response (200):**

```
{
 "success": true,
 "message": "Driver reassigned successfully",
 "data": {
 "id": "uuid-trip-id",
 "shortId": "TRP20260123001",
 "status": "DRIVER_ASSIGNED",
 "tripType": "AIRPORT",
 "pickupLocation": "T3 Terminal, Gate 4",
 "dropLocation": "Cyber Hub, Gurgaon",
 "pickupTime": "2025-12-25T10:00:00Z",
 "price": 1200,
 "driverId": "uuid-new-driver-id",
 "vehicleSku": "TATA_TIGOR_EV"
 }
}
```

**Side Effects:**

- Old driver becomes available
  - New driver becomes unavailable
  - **MMT Trip:** Calls `POST /dispatch/{booking_id}/reassign` to MMT
- 

## 2.5 Unassign Driver

**Endpoint:** `POST /admin/trips/unassign` **Role:** `OPERATIONS` , `MANAGER`

**Use Case:** Remove driver from trip without assigning a new one immediately.

**Logic:**

- Reverts trip status to `CREATED`
- Makes driver available for other trips
- Supports detaching even if trip status is `STARTED` (useful for breakdown scenarios)
- **Does NOT** send MMT unassign event (as per MMT integration requirements)

**Request Body:**

```
{
 "tripId": "uuid-trip-id"
}
```

**Response (200):**

```
{
 "success": true,
 "message": "Driver unassigned successfully"
}
```

**Note:** For MMT trips during testing, use `/detach` endpoint instead if you need to notify MMT.

---

## 2.6 Detach Driver (MMT Only)

**Endpoint:** `POST /admin/trips/detach` **Role:** `OPERATIONS` , `MANAGER`

**Use Case:** For MMT trips only - send detach event to MMT without changing internal trip state.

**Description:** This endpoint is specifically for MMT integration testing scenarios (Test Cases 4, 5, 6). It sends the MMT `detach/unassign` event while optionally keeping the trip available for reassignment internally.

**Request Body:**

```
{
 "tripId": "uuid-trip-id",
 "reason": "Driver unavailable"
}
```

**Response (200):**

```
{
 "success": true,
 "message": "Driver detached and partner notified (if applicable)"
}
```

#### Side Effects:

- **MMT Trip:** Calls `POST /dispatch/{booking_id}/unassign` to MMT (if configured) or detach.
- **Internal:** No change to trip status (trip remains available).

*[!NOTE] **Vendor Cancellation:** This endpoint maps to MMT's **Vendor Cancellation** flow ( `/detach` ). Use this when you cannot fulfill the booking.*

#### How Detach Flow Matches Our System:

- **Manual Process:** Currently, this is a **manual admin process**. When an admin determines a trip cannot be serviced (e.g., no drivers available), they must use the "Detach Driver" function.
- **Automation Gap:** There is currently **no auto-detach** logic (e.g., timeout). If required, this logic would need to call this same `detachBooking` method.

#### Test Cases:

- Test Case 4: Assign > Start > Arrived > Detach
- Test Case 5: Assign > Detach
- Test Case 6: Detach (no prior assignment)

#### Side Effects:

- Status: Reverts to `CREATED` (trip available for re-dispatch)
- Driver marked as `isAvailable: true`
- Assignment status updated to `UNASSIGNED`
- **If MMT Trip:** Calls `POST /dispatch/{booking_id}/detach` to MMT

#### Allowed Trip Statuses:

- `CREATED` - Trip not yet assigned (no-op)
- `DRIVER_ASSIGNED` - Driver assigned but hasn't started
- `STARTED` - Driver started trip but customer hasn't boarded yet

*[!WARNING] Cannot detach after passenger has boarded ( `BOARDED` status) or trip is completed/cancelled.*

## 2.6 Partner Bookings (MMT)

**Identification:** Trips originating from MakeMyTrip will have:

- `provider` : "MMT"
- `providerBookingId` : "BKS88888800926" (MMT's booking ID used for all tracking)
- `providerMapping.externalBookingId` : Same as above
- `providerMeta.otp` : OTP for passenger verification

```
{
 "provider": "MMT",
 "providerBookingId": "BKS88888800926",
}
```

```

"providerMapping": {
 "providerType": "MMT",
 "externalBookingId": "BKS88888800926",
 "providerStatus": "CONFIRMED"
},
"providerMeta": {
 "otp": "1056"
}
}

```

### MMT Tracking Events (Automatic):

When you perform admin actions, the system automatically sends tracking events to MMT:

Admin Action	MMT API Called	Endpoint
Assign Driver	POST /dispatch/{booking_id}/assign	Sends driver + vehicle details
Reassign Driver	POST /dispatch/{booking_id}/reassign	Sends new driver + vehicle
Unassign Driver	POST /dispatch/{booking_id}/unassign	Removes driver from booking

### Driver App Actions → MMT:

Driver Action	MMT API Called
Start Trip	POST /track/{booking_id}/start
Arrived	POST /track/{booking_id}/arrived
Onboard (OTP verified)	POST /track/{booking_id}/boarded
Complete Trip	POST /track/{booking_id}/alight
No Show	POST /track/{booking_id}/not-boarded
Location Update	PUT /track/{booking_id}/location (every 30s)

### Operational Rules:

- Priority Assignment:** MMT trips should be assigned promptly as they have SLA requirements.
- Cancellation:** Unassigning triggers /dispatch/{booking\_id}/unassign webhook to MMT.
- Reassignment:** Triggers /dispatch/{booking\_id}/reassign webhook with new driver details.
- Vehicle Required:** Driver must have an assigned vehicle before being assigned to MMT trips.

### Environment Variables for MMT:

```

Inbound (MMT → DriversKlub)
MMT_INBOUND_USERNAME=mmt_inbound_service
MMT_INBOUND_PASSWORD=your_secure_password

Outbound (DriversKlub → MMT)
MMT_TRACKING_URL=https://cabs-partners-staging.makemytrip.com/tracking/pp2/api/partner/v1

```



```
MMT_TRACKING_USER=your_mmt_tracking_username
MMT_TRACKING_PASS=your_mmt_tracking_password
```

## 3. Fleet & Asset Management

### 3.1 Fleets (Operators)

*[!TIP] **Rapido Integration:** Rapido Captains are managed as a Fleet. Their status is synced automatically.*

#### Create Fleet

**Endpoint:** POST /fleets

**Role:** SUPER\_ADMIN

#### Request Body:

```
{
 "name": "Delhi Cabs Pvt Ltd",
 "mobile": "9999988888",
 "city": "DELHI",
 "fleetType": "COMPANY",
 "panNumber": "ABCDE1234F"
}
```

#### Response (201):

```
{
 "success": true,
 "data": {
 "id": "uuid",
 "shortId": "FLT20260123001",
 "name": "Delhi Cabs Pvt Ltd",
 "city": "DELHI"
 }
}
```

#### List Fleets

**Endpoint:** GET /fleets

**Roles:** SUPER\_ADMIN, OPERATIONS

#### Response (200):

```
{
 "success": true,
 "data": [
 {
```

```
 "id": "uuid",
 "name": "Delhi Cabs Pvt Ltd",
 "city": "DELHI",
 "fleetType": "COMPANY",
 "status": "ACTIVE"
 }
]
```

## Get Fleet Details

### 3.1.1 Hub Management

#### Manage Hubs

**Create Hub:** POST /fleets/:id/hubs **List Hubs:** GET /fleets/:id/hubs **Get Hub Details:** GET /fleets/hubs/:id

#### Manage Hub Managers

**Note:** Hub Managers are regular Users with role `MANAGER` and a linked `fleetId`.

**Create Manager:** POST /fleets/:id/hub-managers (Creates User with role `MANAGER`) **List Managers:** GET /fleets/:id/hub-managers **Assign Manager:** POST /fleets/hubs/:hubId/assign-manager

#### Fleet Resources (Hub Context)

**Add Vehicle to Hub:** POST /fleets/hubs/:id/add-vehicle **Remove Vehicle from Hub:** POST /fleets/hubs/:id/remove-vehicle **Add Driver to Hub:** POST /fleets/hubs/:id/add-driver **Remove Driver from Hub:** POST /fleets/hubs/:id/remove-driver

## 3.2 Vehicles (Cars)

#### Add Vehicle

**Endpoint:** POST /vehicles

**Roles:** `SUPER_ADMIN`, `OPERATIONS`

#### Request Body:

```
{
 "fleetId": "uuid-fleet-id",
 "vehicleNumber": "DL10CA1234",
 "vehicleName": "Tata Tigor EV",
 "fuelType": "ELECTRIC",
 "ownership": "OWNED"
}
```

#### Response (201):

```
{
 "success": true,
 "data": {
```

```
{
 "id": "uuid",
 "vehicleNumber": "DL10CA1234",
 "vehicleName": "Tata Tigor EV"
}
```

### List Vehicles by Fleet

**Endpoint:** GET /vehicles/fleet/:fleetId

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Response (200):

```
{
 "success": true,
 "data": [
 {
 "id": "uuid",
 "shortId": "VEH20260123001",
 "vehicleNumber": "DL10CA1234",
 "vehicleName": "Tata Tigor EV",
 "fuelType": "ELECTRIC",
 "status": "ACTIVE"
 }
]
}
```

### Update Vehicle Documents

**Endpoint:** PATCH /vehicles/:id/docs

**Roles:** SUPER\_ADMIN , OPERATIONS

#### Request Body:

```
{
 "rcUrl": "https://s3.aws.com/rc.pdf",
 "insuranceUrl": "https://s3.aws.com/insurance.pdf"
}
```

### Update Vehicle Status

**Endpoint:** PATCH /vehicles/:id/status

**Roles:** SUPER\_ADMIN , OPERATIONS

#### Request Body:

```
{
 "status": "ACTIVE"
}
```

#### Status Values:

- `ACTIVE` - Vehicle is operational
- `INACTIVE` - Vehicle temporarily unavailable
- `MAINTENANCE` - Vehicle under maintenance

#### Deactivate Vehicle

**Endpoint:** `PATCH /vehicles/:id/deactivate`

**Roles:** `SUPER_ADMIN`, `OPERATIONS`

---

### 3.3 Drivers (Profiles)

#### Onboard Driver

**Endpoint:** `POST /drivers`

**Roles:** `SUPER_ADMIN`, `OPERATIONS`

#### Request Body:

```
{
 "fleetId": "uuid-fleet-id",
 "firstName": "Raj",
 "lastName": "Kumar",
 "mobile": "9812345678",
 "email": "raj@example.com",
 "licenseNumber": "DL-12345-67890"
}
```

#### Response (201):

```
{
 "success": true,
 "data": {
 "id": "uuid",
 "shortId": "DRV20260123001",
 "firstName": "Raj",
 "lastName": "Kumar",
 "mobile": "9812345678"
 }
}
```

#### Update Driver (Full KYC)

**Endpoint:** `PATCH /drivers/:id`

**Roles:** `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

#### Request Body (All fields optional):

```
{
 "firstName": "Raj",
}
```

```

"lastName": "Kumar",
"mobile": "9812345678",
"email": "raj@example.com",
"dob": "1990-05-15T00:00:00.000Z",
"address": "123 Main Street",
"city": "Delhi",
"pincode": "110001",

"aadharNumber": "123456789012",
"driverAgreement": "https://s3.aws.com/agreement.pdf",
"policeVerification": "https://s3.aws.com/pcv.jpg",
"currentAddressProof": "https://s3.aws.com/current.jpg",
"permanentAddressProof": "https://s3.aws.com/permanent.jpg",
"panNumber": "ABCDE1234F",
"dlNumber": "DL-12345-67890",
"gstNumber": "22AAAAA0000A1Z5",

"bankAccountNumber": "1234567890123456",
"bankIfscCode": "HDFC0001234",
"bankAccountName": "Raj Kumar",

"licenseFront": "https://s3.aws.com/license-front.jpg",
"licenseBack": "https://s3.aws.com/license-back.jpg",
"aadharFront": "https://s3.aws.com/aadhaar-front.jpg",
"aadharBack": "https://s3.aws.com/aadhaar-back.jpg",
"panCardImage": "https://s3.aws.com/pan.jpg",
"bankIdProof": "https://s3.aws.com/bank-proof.jpg",

"rcFrontImage": "https://s3.aws.com/rc-front.jpg",
"rcBackImage": "https://s3.aws.com/rc-back.jpg",
"fitnessImage": "https://s3.aws.com/fitness.jpg",
"fitnessExpiry": "2026-12-31",
"insuranceImage": "https://s3.aws.com/insurance.jpg",
"insuranceStart": "2024-01-01",
"insuranceExpiry": "2025-12-31",
"chassisNumber": "MA1AB2CD3EF456789",
"vinNumber": "1HGBH41JXMN109186"
}

```

#### Field Categories:

Category	Fields	Description
Basic Info	firstName , lastName , mobile , email , dob , address , city , pincode	Personal details
KYC Values	aadharNumber , panNumber , dlNumber , gstNumber	Document numbers
Bank Details	bankAccountNumber , bankIfscCode , bankAccountName	For payouts
KYC Attachments	licenseFront , licenseBack , aadharFront , aadharBack , panCardImage , bankIdProof	Document images (upload via S3)
Vehicle Docs	rcFrontImage , rcBackImage , fitnessImage , fitnessExpiry , insuranceImage , insuranceStart , insuranceExpiry ,	Updated on driver's assigned vehicle

	chassisNumber , vinNumber	
--	---------------------------	--

**Note:** Vehicle fields are only saved if the driver has an assigned vehicle.

#### List Drivers by Fleet

**Endpoint:** GET /drivers/fleet/:fleetId

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Response (200):**

```
{
 "success": true,
 "data": [
 {
 "id": "uuid",
 "shortId": "DRV20260123001",
 "firstName": "Raj",
 "lastName": "Kumar",
 "mobile": "9812345678",
 "status": "ACTIVE",
 "kycStatus": "APPROVED"
 }
]
}
```

#### Get Driver Details

**Endpoint:** GET /drivers/:id

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

### 3.4 Driver Operations (Status & Availability)

#### Update Status

**Endpoint:** PATCH /drivers/:id/status **Roles:** SUPER\_ADMIN , OPERATIONS

**Request Body:**

```
{
 "status": "SUSPENDED"
}
```

#### Update Availability

**Endpoint:** PATCH /drivers/:id/availability **Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Request Body:**

```
{
 "isAvailable": false
}
```

```
}
```

### 3.5 Image Upload Service (S3 Presigned URLs)

**Endpoint:** `GET /drivers/upload-url`

**Roles:** `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`, `DRIVER`

**Description:** Generate secure presigned URLs for uploading images and documents directly to S3. This is used for driver documents, vehicle photos, and other assets.

**Query Parameters:**

- `folder` (required): `selfies`, `odometer`, `documents`, `profiles`, `vehicles`
- `fileType` (required): `jpg`, `jpeg`, `png`, `pdf`

**Request Example:**

```
GET /drivers/upload-url?folder=documents&fileType=pdf
Authorization: Bearer <ACCESS_TOKEN>
```

**Response (200):**

```
{
 "success": true,
 "data": {
 "uploadUrl": "https://s3.amazonaws.com/driversklub-assets/documents/uuid.pdf?X-Amz-...",
 "key": "documents/uuid.pdf",
 "url": "https://driversklub-assets.s3.ap-south-1.amazonaws.com/documents/uuid.pdf"
 },
 "message": "Upload URL generated successfully"
}
```

**Upload Flow:**

- Request Upload URL:** Call this endpoint
- Upload File:** Send `PUT` request to `uploadUrl` with file as binary body
- Store URL:** Save the `url` field to database or send to other APIs

**Example Upload (JavaScript/React):**

```
// Step 1: Get presigned URL
const response = await fetch(
 `${baseUrl}/drivers/upload-url?folder=documents&fileType=pdf`,
 {
 headers: { 'Authorization': `Bearer ${token}` }
 }
);
const { data } = await response.json();
```

```
// Step 2: Upload file to S3
await fetch(data.uploadUrl, {
 method: 'PUT',
 body: file,
 headers: { 'Content-Type': 'application/pdf' }
});

// Step 3: Use the final URL
const documentUrl = data.url;
```

#### Use Cases:

- Upload driver license/Aadhaar during onboarding
- Upload vehicle RC/insurance documents
- Upload odometer photos during attendance
- Upload profile pictures

**Note:** Presigned URLs expire in 5 minutes.

---

## 4. Operations & Assignments

---

### 4.1 Attendance Management

#### Approve Attendance

**Endpoint:** POST /attendance/:id/approve **Roles:** SUPER\_ADMIN , MANAGER

#### Request Body:

```
{
 "remarks": "Approved by Ops"
}
```

#### Reject Attendance

**Endpoint:** POST /attendance/:id/reject **Roles:** SUPER\_ADMIN , MANAGER

#### Get Attendance Details

**Endpoint:** GET /attendance/:id **Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

### 4.2 Daily Vehicle Assignment (Roster)

**Endpoint:** POST /assignments

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Description:** Link a driver to a car for the day

#### Request Body:



```
{
 "driverId": "uuid-driver",
 "vehicleId": "uuid-vehicle",
 "fleetId": "uuid-fleet"
}
```

**Response (201):**

```
{
 "success": true,
 "data": {
 "id": "uuid",
 "shortId": "ASN20260123001",
 "driverId": "uuid",
 "vehicleId": "uuid",
 "startDate": "2025-12-25T00:00:00Z"
 }
}
```

**Goal:** Driver cannot receive trips without this active link.

---

### 4.3 Get Assignments by Fleet

**Endpoint:** GET /assignments/fleet/:fleetId

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Response (200):**

```
{
 "success": true,
 "data": [
 {
 "id": "uuid",
 "shortId": "ASN20260123001",
 "driver": {
 "firstName": "Raj",
 "lastName": "Kumar"
 },
 "vehicle": {
 "vehicleNumber": "DL10CA1234"
 },
 "startDate": "2025-12-25T00:00:00Z",
 "status": "ACTIVE"
 }
]
}
```

---

## 4.4 End Assignment

**Endpoint:** PATCH /assignments/:id/end

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

---

## 4.5 Driver Preference Management

### Get Pending Requests

**Endpoint:** GET /drivers/preference/pending-requests

**Roles:** SUPER\_ADMIN , OPERATIONS

**Response (200):**

```
{
 "success": true,
 "statusCode": 200,
 "message": "Pending preference requests retrieved successfully",
 "data": [
 {
 "id": "bd3c2df9-d58d-4b5a-8d20-2ecd8db1b63e",
 "shortId": "REQ20260123001",
 "driverId": "ad8324ca-2dea-4618-ba5e-3095fa123d06",
 "currentPreference": {
 "accept_rentals": false,
 "prefer_airport_rides": false
 },
 "requestedPreference": {
 "accept_rentals": true,
 "prefer_airport_rides": true
 },
 "status": "PENDING",
 "requestAt": "2026-01-08T04:38:38.415Z"
 }
]
}
```

### Update Request Status

**Endpoint:** POST /drivers/preference/update-status

**Roles:** SUPER\_ADMIN , OPERATIONS

**Request Body (Approve):**

```
{
 "id": "bd3c2df9-d58d-4b5a-8d20-2ecd8db1b63e",
 "status": "APPROVED"
}
```

**Request Body (Reject):**

```
{
 "id": "bd3c2df9-d58d-4b5a-8d20-2ecd8db1b63e",
 "status": "REJECTED",
 "rejection_reason": "demo test"
}
```

---

## 5. User Management

### 5.1 Create User

**Endpoint:** POST /users

**Role:** SUPER\_ADMIN

**Request Body:**

```
{
 "phone": "9876543210",
 "role": "DRIVER",
 "name": "Raj Kumar"
}
```

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER , DRIVER

---

### 5.2 List All Users

**Endpoint:** GET /users

**Roles:** SUPER\_ADMIN , OPERATIONS

---

### 5.3 Deactivate User

**Endpoint:** PATCH /users/:id/deactivate

**Role:** SUPER\_ADMIN

---

---

## 6. Pricing Calculator

### 6.1 Preview Pricing

**Endpoint:** POST /pricing/preview

**Auth Required:** Yes

**Description:** "Get Estimate" button on Create Trip form. Uses client-provided distance to calculate fare.

**Request Body:**

```
{
 "pickup": "Connaught Place, New Delhi",
 "drop": "Cyber City, Gurgaon",
 "tripType": "INTER_CITY",
 "tripDate": "2024-05-20T10:00:00.000Z",
 "bookingDate": "2024-05-19T10:00:00.000Z",

 // Vehicle specification (choose one):
 "vehicleType": "EV", // Direct type
 "vehicleSku": "TATA_TIGOR_EV", // Or use SKU (auto-detected)

 "distanceKm": 25.5 // Optional: Fallback if Google Maps fails
}
```

#### Response (200):

```
{
 "success": true,
 "data": {
 "distanceSource": "GOOGLE_MAPS", // or "CLIENT_PROVIDED"
 "billableDistanceKm": 25.5,
 "ratePerKm": 25,
 "baseFare": 637.5,
 "totalFare": 765,
 "breakdown": {
 "distanceFare": 637.5,
 "tripTypeMultiplier": 1.2,
 "bookingTimeMultiplier": 1.0,
 "vehicleMultiplier": 1.0
 },
 "currency": "INR"
 },
 "message": "Fare calculated successfully"
}
```

**[!NOTE] Distance Calculation:** The admin dashboard (or client app) should calculate the distance using Google Maps, Mapbox, or similar service before calling this API. The backend pricing engine uses the `distanceKm` value provided in the request to calculate the fare.

## 7. Payment & Financial Management

### 7.1 Create Rental Plan

**Endpoint:** POST `/payments/admin/rental-plans`

**Roles:** SUPER\_ADMIN, OPERATIONS

**Request Body:**

```
{
 "fleetId": "uuid",
 "name": "Weekly Plan",
 "rentalAmount": 3500,
 "depositAmount": 5000,
 "validityDays": 7
}
```

**Response (201):**

```
{
 "id": "uuid",
 "shortId": "PLN20260123001",
 "fleetId": "uuid",
 "name": "Weekly Plan",
 "rentalAmount": 3500,
 "depositAmount": 5000,
 "validityDays": 7,
 "isActive": true
}
```

**Use Case:**

- **Onboarding:** Create standard plans (e.g., "Weekly Gold") for new drivers to choose from during registration.
- 

## 7.2 Get Rental Plans

**Endpoint:** GET /payments/admin/rental-plans/:fleetId

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Query Parameters:**

- `activeOnly` (boolean, default: true)
- 

## 7.3 Create Penalty

**Endpoint:** POST /payments/admin/penalty

**Roles:** SUPER\_ADMIN , OPERATIONS

**Request Body:**

```
{
 "driverId": "uuid",
 "type": "MONETARY",
 "amount": 500,
 "reason": "Customer complaint",
}
```

```
 "category": "BEHAVIOR"
 }
```

#### Penalty Types:

- **MONETARY** - Financial penalty (auto-deducted from deposit for rental model)
- **WARNING** - Verbal/written warning
- **SUSPENSION** - Temporary suspension (requires `suspensionStartDate` and `suspensionEndDate` )
- **BLACKLIST** - Permanent ban

#### Response (201):

```
{
 "id": "uuid",
 "shortId": "PNL20260123001",
 "driverId": "uuid",
 "type": "MONETARY",
 "amount": 500,
 "reason": "Customer complaint",
 "isPaid": true,
 "deductedFromDeposit": true
}
```

#### Use Case:

- **Quality Control:** Penalize drivers for "No Shows" or poor behavior reported by customers.
- **Deterrence:** Deduct from deposit immediately to enforce compliance.

---

## 7.4 Waive Penalty

**Endpoint:** `POST /payments/admin/penalty/:id/waive`

**Roles:** `SUPER_ADMIN` , `OPERATIONS`

#### Request Body:

```
{
 "waiverReason": "First-time offense, driver apologized"
}
```

#### Response (200):

```
{
 "success": true,
 "message": "Penalty waived successfully"
}
```

#### Side Effects:

- Refunds deposit if already deducted
- Reverses suspension/blacklist status

**Use Case:**

- **Dispute Resolution:** If a driver provides valid proof (e.g., car breakdown), Ops can waive the penalty.
- 

## 7.5 Create Incentive

**Endpoint:** POST /payments/admin/incentive

**Roles:** SUPER\_ADMIN , OPERATIONS

**Request Body:**

```
{
 "driverId": "uuid",
 "amount": 500,
 "reason": "Completed 50 trips this month",
 "category": "MILESTONE"
}
```

**Response (201):**

```
{
 "id": "uuid",
 "shortId": "INC20260123001",
 "driverId": "uuid",
 "amount": 500,
 "reason": "Completed 50 trips this month",
 "isPaid": false
}
```

---

## 7.6 Process Incentive Payout

**Endpoint:** POST /payments/admin/incentive/:id/payout

**Roles:** SUPER\_ADMIN , OPERATIONS

**Response (200):**

```
{
 "success": true,
 "txnId": "TXN_1735123456_PAY123",
 "status": "PENDING",
 "utr": "UTR123456789"
}
```

**Note:** Sends money to driver's bank account via Easebuzz

#### Use Case:

- **Reward Distribution:** Operation team processes the approved incentive to credit the driver's bank account.
- 

### 7.7 Reconcile Daily Collection

**Endpoint:** POST /payments/admin/collection/:id/reconcile

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Request Body:

```
{
 "expectedRevenue": 5000,
 "reconciliationNotes": "All collections verified"
}
```

#### Response (200):

```
{
 "success": true,
 "message": "Collection reconciled successfully"
}
```

#### Side Effects:

- Calculates revenue share
- Applies incentives and penalties
- Prepares for payout

#### Use Case:

- **End-of-Day Ops:** Manager verifies the physical cash collected matches the system's `expectedRevenue` before closing the shift.
- 

### 7.8 Process Daily Payout

POST /payment/admin/collection/:id/payout

**DEPRECATED:** Use Bulk Payout instead. **Roles:** SUPER\_ADMIN , OPERATIONS

---

### 7.9 Bulk Payout (Manual)

**Endpoint:** POST /payments/admin/bulk-payout **Roles:** SUPER\_ADMIN , OPERATIONS

**Request:** multipart/form-data

- `file` : CSV File ( `phone, amount` or `accountNumber, amount` )

#### Response:



```
{
 "total": 10,
 "success": 9,
 "failed": 1,
 "amountDisbursed": 45000
}
```

#### Use Case:

- **Weekly Settlements:** Accountant uploads a CSV of all driver payouts on Monday morning to process them in one batch.

## 7.10 Vehicle QR Generation

Generate and manage Easebuzz virtual account QR codes for vehicles.

**Endpoint:** POST /payments/admin/vehicle/:vehicleId/qr

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Response (201):

```
{
 "success": true,
 "data": {
 "virtualAccountId": "VA123456789",
 "qrCodeBase64": "https://api.qrserver.com/v1/create-qr-code/...",
 "upiId": "vehicle@easebuzz"
 }
}
```

**Get Existing QR:** GET /payments/admin/vehicle/:vehicleId/qr

*[!NOTE] Field Name Clarification: The `qrCodeBase64` field can contain either:*

- A **URL** (from Easebuzz or fallback QR generator)
- A **base64 string** (in some cases)

*In test mode, a fallback QR is generated using the UPI ID.*

#### Frontend Usage:

```
// Display QR code - works for both URL and base64
<img
 src={qrCodeBase64?.startsWith('http')
 ? qrCodeBase64
 : `data:image/png;base64,${qrCodeBase64}`}
 alt="Vehicle QR"
/>
```

```
// Simpler: If you're sure it's a URL (current implementation)

```

#### Features:

- Scannable with any UPI app
  - Payments tracked automatically
  - Print for vehicle placement
  - Fallback QR generated in test mode
- 

### 7.10 Get Pending Reconciliations

**Endpoint:** GET /payments/admin/reconciliations/pending

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Response (200):

```
{
 "reconciliations": [
 {
 "id": "uuid",
 "shortId": "REC20260123001",
 "driver": {
 "firstName": "Raj",
 "lastName": "Kumar"
 },
 "date": "2025-12-29T00:00:00.000Z",
 "totalCollection": 5000,
 "isReconciled": false
 }
]
}
```

#### Use Case:

- **Manager Dashboard:** Show a list of drivers whose collections are yet to be verified for the previous day.
- 

### 7.11 Get Pending Payouts

**Endpoint:** GET /payments/admin/payouts/pending

**Roles:** SUPER\_ADMIN , OPERATIONS

#### Response (200):

```
{
 "payouts": [
 {
 "id": "uuid",
 "shortId": "PAY20260123001",
 "driver": {
 "firstName": "Raj",

```

```
 "lastName": "Kumar",
 "bankAccountNumber": "1234567890"
 },
 "date": "2025-12-29T00:00:00.000Z",
 "netPayout": 3800,
 "isPaid": false
}
]
```

#### Use Case:

- **Finance Review:** Finance team reviews all verified collections that are ready for payout before initiating the bank transfer.
- 

### 7.12 Generate Vehicle QR Code

**Endpoint:** POST /payments/admin/vehicle/:id/qr

**Roles:** SUPER\_ADMIN , OPERATIONS

#### Response (201):

```
{
 "id": "uuid",
 "shortId": "VQR20260123001",
 "vehicleId": "uuid",
 "virtualAccountId": "VA123456",
 "virtualAccountNumber": "1234567890123456",
 "ifscCode": "HDFC0000001",
 "qrCodeBase64": "data:image/png;base64,...",
 "upiId": "driversklub.va123456@easebuzz",
 "isActive": true
}
```

#### Use Case:

- **New Car Setup:** Generate a unique QR code sticker for a new vehicle so passengers can pay via UPI directly to the vehicle's virtual account.
- 

### 7.13 Get Vehicle QR Code

**Endpoint:** GET /payments/admin/vehicle/:id/qr

**Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Response (200):

```
{
 "id": "uuid",
 "shortId": "VQR20260123001",
 "vehicleId": "uuid",
 "qrCodeBase64": "data:image/png;base64,...",
 "upiId": "driversklub.va123456@easebuzz",
}
```

```
"isActive": true
}
```

#### Use Case:

- **Reprinting:** Manager retrieves the existing QR code if the physical sticker is damaged or lost.

## 7.14 InstaCollect Orders (Dynamic QR)

### Create Order (Generate Dynamic QR)

**Endpoint:** POST /payments/orders **Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Request Body:

```
{
 "customerName": "John Doe",
 "customerPhone": "9876543210",
 "amount": 2500,
 "description": "Advance Payment for Trip #123"
}
```

#### Response (201):

```
{
 "success": true,
 "data": {
 "id": "uuid-order-id",
 "shortId": "ORD20260123001",
 "totalAmount": 2500,
 "collectedAmount": 0,
 "remainingAmount": 2500,
 "status": "PENDING",
 "virtualAccountId": "VA_ORDER_123",
 "qrCodeBase64": "...",
 "upiId": "driversklub.order123@easebuzz"
 }
}
```

#### Use Case:

- **Ad-Hoc Payments:** Driver or Admin enters an amount on the app/dashboard to generate a **one-time QR code** for a passenger to scan and pay instantly.

### Get Order Details

**Endpoint:** GET /payments/orders/:id **Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

#### Response (200):

```
{
 "success": true,
 "data": {
 "id": "uuid-order-id",
 "shortId": "ORD20260123001",
 "customerName": "John Doe",
 "customerPhone": "9876543210",
 "description": "Advance Payment",
 "totalAmount": 2500,
 "collectedAmount": 1000,
 "remainingAmount": 1500,
 "status": "PARTIAL",
 "virtualAccountId": "VA_ORDER_123",
 "qrCodeBase64": "data:image/png;base64...",
 "transactions": [
 {
 "id": "txn-uuid",
 "amount": 1000,
 "status": "SUCCESS",
 "date": "2025-12-25T10:00:00Z"
 }
],
 "createdAt": "2025-12-25T09:00:00Z"
 }
}
```

## List Orders

**Endpoint:** GET /payments/orders **Roles:** SUPER\_ADMIN , OPERATIONS , MANAGER

### Query Params:

- page (default: 1)
- limit (default: 10)
- status (optional): PENDING , PARTIAL , COMPLETED
- search (optional): Filter by Customer Name or Phone

### Response (200):

```
{
 "success": true,
 "data": [
 {
 "id": "uuid-order-1",
 "shortId": "ORD20260123001",
 "customerName": "Alice Smith",
 "totalAmount": 5000,
 "collectedAmount": 5000,
 "status": "COMPLETED",
 "createdAt": "2025-12-24T10:00:00Z"
 },
 {
 "id": "uuid-order-2",
```

```

 "customerName": "Bob Jones",
 "totalAmount": 2000,
 "collectedAmount": 0,
 "status": "PENDING",
 "createdAt": "2025-12-25T11:00:00Z"
 }
],
"pagination": {
 "page": 1,
 "limit": 10,
 "total": 50,
 "totalPages": 5
}
}

```

## 8. Frontend Implementation Notes

### 8.1 CORS

- **Current:** Configured to allow all origins ( \* )
- **Production:** Whitelist specific domains

### 8.2 Date Handling

- Use `date-fns` or `moment` to parse UTC ISO strings from API
- **Always display in User's Local Time**
- Store in UTC, display in local

```

import { format, parseISO } from 'date-fns';

const displayTime = format(parseISO(trip.pickupTime), 'PPpp');
// Output: "Dec 25, 2025, 10:00 AM"

```

### 8.3 State Management

#### Recommendations:

- Cache `Fleets` and `Drivers` lists (TanStack Query recommended) as they change infrequently
- Poll `Trips` list (every 30s) or use a "Refresh" button for operations
- Handle `401 Unauthorized` by redirecting to Login
- Implement optimistic updates for better UX

#### Example with TanStack Query:

```

const { data: trips } = useQuery({
 queryKey: ['trips', { status, page }],
 queryFn: () => fetchTrips({ status, page }),

```

```
 refetchInterval: 30000, // 30 seconds
 });
```

## 8.4 Error Handling

```
try {
 await assignDriver(tripId, driverId);
 toast.success('Driver assigned successfully');
} catch (error) {
 if (error.response?.status === 401) {
 // Redirect to login
 router.push('/login');
 } else {
 toast.error(error.response?.data?.message || 'Failed to assign driver');
 }
}
```

## 8.5 Role-Based UI

```
const canCreateTrip = ['SUPER_ADMIN', 'OPERATIONS'].includes(user.role);
const canApproveAttendance = ['SUPER_ADMIN', 'MANAGER'].includes(user.role);

{canCreateTrip && <Button onClick={openCreateTripModal}>Create Trip</Button>}
```

## 8.6 Pagination Component

```
<Pagination
 currentPage={page}
 totalPages={Math.ceil(total / limit)}
 onPageChange={setPage}
/>
```

## 8.7 Status Badge Component

```
const getStatusColor = (status) => {
 switch (status) {
 case 'CREATED': return 'gray';
 case 'DRIVER_ASSIGNED': return 'blue';
 case 'STARTED': return 'yellow';
 case 'COMPLETED': return 'green';
 case 'CANCELLED': return 'red';
 default: return 'gray';
 }
}
```

```
 }
 };

 <Badge color={getStatusColor(trip.status)}>{trip.status}</Badge>
```

## 8.8 Real-time Updates (Optional)

Consider implementing WebSocket connection for real-time trip status updates:

```
const socket = io('wss://driversklub-backend.onrender.com');

socket.on('trip:updated', (trip) => {
 queryClient.setQueryData(['trip', trip.id], trip);
});
```



## Checklist for Production

- ☐ Implement token refresh logic
- ☐ Add role-based access control to UI
- ☐ Implement pagination for all list views
- ☐ Add loading states for all API calls
- ☐ Add error handling with user-friendly messages
- ☐ Implement date/time formatting (UTC → Local)
- ☐ Add confirmation dialogs for destructive actions
- ☐ Implement search/filter functionality
- ☐ Add export to CSV functionality
- ☐ Test all edge cases (empty states, errors, etc.)

## 9. Rapido Operational Monitoring

### 9.1 Conflict Resolution Logic

The dashboard does not have a manual "Sync Rapido Status" button because the process is fully automated.

- **Logic:** The backend runs a worker every 5 minutes.
- **Conflicts:** If a driver is found ONLINE on Rapido but BUSY internally, the system auto-corrects this.
- **Logs:** All auto-corrections are logged in the backend logs (viewable via server logs).

### 9.2 Manual Override Alert

If a driver manually forces themselves ONLINE in the Rapido app:

1. System detects `status: online` webhook.
2. System checks assignments.



3. If conflict exists, system forces OFFLINE immediately.

---

## 10. Maps Service

---

### 10.1 Location Autocomplete

**Endpoint:** GET /maps/autocomplete

**Auth Required:** Yes

**Role:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Query Parameters:**

- `query` (required): Search text (e.g., "Airport")

**Response (200):**

```
{
 "success": true,
 "data": [
 {
 "description": "Indira Gandhi International Airport, New Delhi",
 "place_id": "ChIJ..."
 }
]
}
```

### 10.2 Geocode Address

**Endpoint:** GET /maps/geocode

**Auth Required:** Yes

**Role:** SUPER\_ADMIN , OPERATIONS , MANAGER

**Query Parameters:**

- `address` (required): Address string

**Response (200):**

```
{
 "success": true,
 "data": {
 "lat": 28.5562,
 "lng": 77.1000,
 "formattedAddress": "Indira Gandhi International Airport..."
 }
}
```