📚 Complete API Reference

Version: 4.6.0 (MMT Smart ID + Pricing Cleanup)
Base URL (Development): `http://localhost:3000` (API Gateway)
Base URL (Staging): `https://driversklub-backend.onrender.com`
Base URL (Production): AWS Elastic Beanstalk `driversklub-backend-env`
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:
  - UUID → Truncated (first 10 hex chars after removing hyphens)
  - ShortID (DRV20260127001) → Smart Compressed (D260127001)
- **Pricing Cleanup** - Removed unused `pricing.rules.ts`, consolidated to `pricing.config.ts`
- **Endpoint Fix** - MMT dispatch uses `/detach` (not `/unassign`), location uses `/update` (not `/location`)

What's New in v4.5.0

- **MMT Integration Complete** - Full inbound + outbound tracking with MakeMyTrip
  - Inbound: search, block, paid, cancel, reschedule, booking details
  - Dispatch events: assign, reassign, detach
  - Track events: start, arrived, boarded, alight, not-boarded, location
  - Automatic `providerBookingId` storage from paid endpoint
- **Public Booking API** - Customers can now book trips without authentication
- **Referral System** - Drivers can refer other drivers and earn incentives
- **Enhanced KYC** - Full driver KYC with Aadhar, PAN, Bank details, and vehicle documents
- **FLEET_ADMIN Role** - New role for fleet-level administration
- **Vehicle Identification** - Chassis number, VIN, and insurance date fields

Architecture Overview

This API is built on a **microservices architecture** with 6 independent services behind an API Gateway:

- **API Gateway** (Port 3000) - Routes all requests
- **Auth Service** (Port 3001) - Authentication & user management
- **Driver Service** (Port 3002) - Driver profiles & attendance
- **Vehicle Service** (Port 3003) - Vehicles, fleets & managers
- **Assignment Service** (Port 3004) - Driver-vehicle assignments
- **Trip Service** (Port 3005) - Trips, payments, pricing & partners
- **Notification Service** (Port 3006) - Real-time notifications

Total Endpoints: 120+

> ***Note:*** *All requests should go through the API Gateway. Individual service ports are for internal communication only.*

Table of Contents

Authentication

All authenticated endpoints require a valid JWT token in the `Authorization` header:

```
Authorization: Bearer <access_token>
```

User Roles

- `SUPER_ADMIN` - Full system access
- `FLEET_ADMIN` - Fleet-level administration (manages their own fleet)
- `OPERATIONS` - Operations team access
- `MANAGER` - Fleet/Hub manager access
- `DRIVER` - Driver-specific access

Core Endpoints

Health Check

GET `/health`

Check server and database connectivity.

Authentication: None

Response:

```
{
  "status": "ok",
  "service": "drivers-klub-backend",
  "database": "connected",
  "timestamp": "2025-12-26T15:51:32.202Z"
}
```

Auth

POST `/auth/send-otp`

Send OTP to user's phone number.

Authentication: None

Request Body:

```
{
  "phone": "9876543210"
}
```

Response:

```
{
  "success": true,
  "message": "OTP sent successfully"
}
```

Notes:

- OTP is sent via Exotel SMS
- Set `EXOTEL_BASE_URL` for a custom Exotel subdomain (optional)
- Use `EXOTEL_SMS_ENDPOINT=sendCopy` if your Exotel account requires the sendCopy API
- For India DLT compliance, set `EXOTEL_DLT_ENTITY_ID` and `EXOTEL_DLT_TEMPLATE_ID`
- Ensure `EXOTEL_SMS_BODY_TEMPLATE` matches your registered DLT template and includes `{{otp}}`
- OTP expires in 5 minutes (configurable)

- Maximum 3 verification attempts per OTP

POST `/auth/verify-otp`

Verify OTP and authenticate user.

Authentication: None

Request Body:

```
{
  "phone": "9876543210",
  "otp": "123456"
}
```

Response:

```
{
  "success": true,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...",
    "user": {
      "id": "uuid",
      "shortId": "USR20260123001",
      "name": "Rajesh Kumar",
      "phone": "9876543210",
      "role": "DRIVER",
      "isActive": true
    }
  }
}
```

Security:

- OTP is deleted after successful verification (prevents reuse)

Request Headers (Optional):

```
x-client-type: app
```

Token Expiry:

- **Access Token:** 15 minutes (all clients)
- **Refresh Token:**
  - Mobile App (`x-client-type: app`): 30 days
  - Web Client (`x-client-type: web`): 1 day
  - Default (no header): 1 day

Note: Set `x-client-type` header to control refresh token expiry duration.

POST `/auth/refresh`

Refresh access token using refresh token.

Authentication: None

Request Body:

```json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Response:

```json
{
  "success": true,
  "data": {
    "accessToken": "new_access_token",
    "refreshToken": "new_refresh_token"
  }
}
```

POST `/auth/logout`

Logout user and revoke tokens.

Authentication: Optional (Bearer token)

Request Body:

```json
{
  "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..."
}
```

Response:

```json
{
  "success": true,
  "message": "Logged out successfully"
}
```

Driver Onboarding (New)

POST `/users/drivers/verify`

Check if a driver phone number is already registered or eligible for signup.

Authentication: None

Request Body:

```json
{
  "phone": "9876543210"
}
```

Response (Success):

```json
{
  "success": true,
  "message": "OTP sent successfully"
}
```

Note: If user exists, returns 404/400 Conflict message.

POST `/users/drivers/verifyOtp`

Verify the OTP for driver registration.

Request Body:

```json
{
  "phone": "9876543210",
  "otp": "123456"
}
```

Response:

```json
{
  "success": true,
  "message": "OTP verified successfully"
}
```

POST `/users/drivers/signup`

Create a new Driver account (User + Driver Profile).

Request Body:

```json
{
  "name": "New Driver",
  "phone": "9876543210"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "user-uuid",
    "name": "New Driver",
    "role": "DRIVER",
    "isActive": true
  }
}
```

Users

All user endpoints require authentication (except Public Signup).

POST `/users/drivers/verify`

Check if a phone number is already registered.

Authentication: None (Public)

Request Body:

```
{
  "phone": "9876543210"
}
```

Response:

```
{
  "success": true,
  "message": "OTP sent successfully",
  "data": {
    "message": "OTP sent successfully"
  }
}
```

Note: If user already exists, returns error `404` with message "User already registered".

POST `/users/drivers/verifyOtp`

Verify OTP for signup (if different from Auth OTP).

Authentication: None (Public)

Request Body:

```
{
  "phone": "9876543210",
  "otp": "123456"
}
```

Response:

```
{
  "success": true,
  "message": "OTP Verified"
}
```

POST `/users/drivers/signup`

Create a new Driver account (Public Self-Registration).

Authentication: None (Public)

Request Body:

```
{
  "name": "Raju Pilot",
  "phone": "9876543210",
  "referralCode": "REF123",   // Optional
  "vehicleDetails": {         // Optional (if creating vehicle)
     "vehicleNumber": "DL01ABC1234",
     "model": "Tata Nexon EV"
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "user": { "id": "...", "shortId": "USR...", "role": "DRIVER" },
    "token": "..." // Auto-login token
  }
}
```

POST `/users`

Create a new user.

Authentication: Required
Roles: `SUPER_ADMIN`

Request Body:

```
{
  "name": "John Doe",
  "phone": "9876543210",
  "role": "DRIVER",
  "isActive": true
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "USR20260123002",
    "name": "John Doe",
    "phone": "9876543210",
    "role": "DRIVER",
    "isActive": true,
    "createdAt": "2025-12-26T10:00:00.000Z"
  }
}
```

GET `/users`

Get all users.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "USR20260123003",
      "name": "John Doe",
      "phone": "9876543210",
      "role": "DRIVER",
      "isActive": true
    }
  ]
}
```

GET `/users/:id`

Get user by ID.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "USR20260123003",
    "name": "John Doe",
    "phone": "9876543210",
    "role": "DRIVER",
    "isActive": true
  }
}
```

PATCH `/users/:id/deactivate`

Deactivate a user.

Authentication: Required
Roles: `SUPER_ADMIN`

Response:

```json
{
  "success": true,
  "message": "User deactivated successfully"
}
```

Drivers

All driver endpoints require authentication.

POST `/drivers/new-driver-onboard`

Complete driver onboarding with full KYC (Public - for driver app signup flow).

Authentication: None (Public endpoint)

Request Body:

```json
{
  "userId": "uuid-from-signup",
  "firstName": "Raj",
  "lastName": "Kumar",
  "mobile": "9876543210",
  "email": "raj@example.com",
  "dob": "1990-05-15T00:00:00.000Z",
  "address": "123 Main Street, Sector 29",
  "city": "Gurgaon",
  "pincode": 122001,

  "aadharNumber": 123456789012,
  "panNumber": "ABCDE1234F",
  "drivingLicenceNumber": "DL-0120230012345",
  "gstNumber": "22AAAAA0000A1Z5",

  "aadharFront": "https://s3.aws.com/aadhaar-front.jpg",
  "aadharBack": "https://s3.aws.com/aadhaar-back.jpg",
  "panPhoto": "https://s3.aws.com/pan.jpg",
  "driverAgreement": "https://s3.aws.com/agreement.pdf",
  "policeVerification": "https://s3.aws.com/pcv.jpg",
  "currentAddressProof": "https://s3.aws.com/current-address.jpg",
  "permanentAddressProof": "https://s3.aws.com/permanent-address.jpg",

  "haveVehicle": true,
  "vehicleModel": "Tata Tigor EV",
  "vehicleType": "SEDAN",
  "registrationNumber": "DL01AB1234",
  "fuelType": "ELECTRIC",
  "ownerName": "Raj Kumar",
  "rcFrontImage": "https://s3.aws.com/rc-front.jpg",
  "rcBackImage": "https://s3.aws.com/rc-back.jpg",

  "referralCode": "REF123ABC"
}
```

Response (201):

```
{
  "success": true,
  "data": {
    "id": "driver-uuid",
    "shortId": "DRV20260123001",
    "firstName": "Raj",
    "lastName": "Kumar",
    "mobile": "9876543210",
    "kycStatus": "PENDING",
    "vehicleId": "vehicle-uuid"
  },
  "message": "Documents Uploaded, KYC pending for verification"
}
```

Process:

1. Creates driver profile with all KYC data
2. If `haveVehicle: true`, creates vehicle and links it to driver
3. If `referralCode` provided, creates referral record
4. Generates unique referral code for new driver

POST `/drivers`

Create a new driver profile (Admin).

Authentication: Required
Roles: `SUPER_ADMIN` , `FLEET_ADMIN` , `OPERATIONS` , `MANAGER`

Request Body:

```json
{
  "userId": "uuid",
  "fleetId": "uuid",
  "firstName": "Rajesh",
  "lastName": "Kumar",
  "mobile": "9876543210",
  "licenseNumber": "DL-0120230012345",
  "profilePic": "https://example.com/photo.jpg",
  "licenseFront": "https://example.com/license-front.jpg",
  "licenseBack": "https://example.com/license-back.jpg",
  "aadharFront": "https://example.com/aadhaar-front.jpg",
  "aadharBack": "https://example.com/aadhaar-back.jpg",
  "panCardImage": "https://example.com/pan.jpg",
  "livePhoto": "https://example.com/live.jpg",
  "bankIdProof": "https://example.com/bank-proof.jpg",
  "kycStatus": "PENDING",
  "status": "ACTIVE",
  "isAvailable": true
}
```

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "DRV20260123002",
    "userId": "uuid",
    "fleetId": "uuid",
    "firstName": "Rajesh",
    "lastName": "Kumar",
    "mobile": "9876543210",
    "licenseNumber": "DL-0120230012345",
    "kycStatus": "PENDING",
    "status": "ACTIVE",
    "isAvailable": true
  }
}
```

GET `/drivers/fleet/:fleetId`

Get all drivers for a specific fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "firstName": "Rajesh",
      "lastName": "Kumar",
      "mobile": "9876543210",
      "status": "ACTIVE",
      "isAvailable": true
    }
  ]
}
```

GET `/drivers/hub/:hubId`

Get all drivers for a specific hub.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "firstName": "Rajesh",
      "lastName": "Kumar",
      "mobile": "9876543210",
      "hubId": "uuid"
    }
  ]
}
```

GET `/drivers/me`

Get current driver's profile.

Authentication: Required
Roles: `DRIVER`

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "firstName": "Rajesh",
    "lastName": "Kumar",
    "mobile": "9876543210",
    "licenseNumber": "DL-0120230012345",
    "kycStatus": "APPROVED",
    "status": "ACTIVE",
    "isAvailable": true,
    "fleet": {
      "id": "uuid",
      "name": "Delhi Cabs Pvt Ltd",
      "city": "DELHI"
    },
    "assignments": [
      {
        "id": "assignment-uuid",
        "status": "ACTIVE",
        "startDate": "2026-01-12T00:00:00Z",
        "vehicle": {
          "id": "vehicle-uuid",
          "vehicleNumber": "DL10CA1234",
          "vehicleName": "Tata Tigor EV",
          "fuelType": "ELECTRIC",
          "vehicleType": "SEDAN"
        }
      }
    ]
  }
}
```

Note: The `assignments` array contains the currently assigned vehicle. If no vehicle is assigned, it will be empty.

~~~

GET `/drivers/:id`

Get driver by ID.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "firstName": "Rajesh",
    "lastName": "Kumar",
    "mobile": "9876543210",
    "licenseNumber": "DL-0120230012345",
    "kycStatus": "APPROVED",
    "status": "ACTIVE"
~~~

```
    }
  }
```

PATCH `/drivers/:id`

Update driver details.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Request Body:

```
{
  "firstName": "Rajesh",
  "lastName": "Kumar",
  "mobile": "9876543210",
  "email": "rajesh@example.com",
  "dob": "1990-05-15T00:00:00.000Z",
  "address": "123 Main Street",
  "city": "Delhi",
  "pincode": "110001",
  "profilePic": "https://example.com/photo.jpg",

  "aadharNumber": "123456789012",
  "panNumber": "ABCDE1234F",
  "dlNumber": "DL-0120230012345",
  "licenseNumber": "DL-0120230012345",
  "gstNumber": "22AAAAA0000A1Z5",

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

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

  "rcFrontImage": "https://example.com/rc-front.jpg",
  "rcBackImage": "https://example.com/rc-back.jpg",
  "fitnessImage": "https://example.com/fitness.jpg",
  "fitnessExpiry": "2026-12-31T00:00:00.000Z",
  "insuranceImage": "https://example.com/insurance.jpg",
  "insuranceStart": "2024-01-01T00:00:00.000Z",
  "insuranceExpiry": "2025-12-31T00:00:00.000Z",
  "chassisNumber": "MA1AB2CD3EF456789",
  "vinNumber": "1HGBH41JXMN109186"
}
```

Field Categories:

Category	Fields
Basic Info	`firstName` , `lastName` , `mobile` , `email` , `dob` , `address` , `city` , `pincode` , `profilePic`
KYC Values	`aadharNumber` , `panNumber` , `dlNumber` , `licenseNumber` , `gstNumber`
Bank Details	`bankAccountNumber` , `bankIfscCode` , `bankAccountName`
KYC Attachments	`licenseFront` , `licenseBack` , `aadharFront` , `aadharBack` , `panCardImage` , `livePhoto` , `bankIdProof`
Vehicle Documents	`rcFrontImage` , `rcBackImage` , `fitnessImage` , `fitnessExpiry` , `insuranceImage` , `insuranceStart` , `insuranceExpiry` , `chassisNumber` , `vinNumber`

Note: Vehicle document fields (`rcFrontImage` , `rcBackImage` , `fitnessImage` , `insuranceImage` , `chassisNumber` , `vinNumber` , etc.) are only updated if the driver has an assigned vehicle.

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "firstName": "Rajesh",
    "lastName": "Kumar"
  }
}
```

PATCH `/drivers/:id/status`

Update driver status.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

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

Possible Status Values:

- `ACTIVE`
- `INACTIVE`
- `SUSPENDED`

PATCH `/drivers/:id/availability`

Update driver availability.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Request Body:

```
{
  "isAvailable": true
}
```

Fleet Management

All fleet endpoints require authentication.

POST `/fleets`

Create a new fleet.

Authentication: Required
Roles: `SUPER_ADMIN`

Request Body:

```
{
  "name": "Demo Fleet Pvt Ltd",
  "city": "GURGAON",
  "mobile": "9876543210",
  "fleetAdminName": "Fleet Admin",
  "panNumber": "ABCDE1234F",
  "fleetType": "COMPANY",
  "status": "ACTIVE"
}
```

Notes:

- `fleetAdminName` is required to create the Fleet Admin user
- Fleet Admin mobile uses the same `mobile` as the Fleet
- `modeId` is optional; defaults to `CAB` if omitted

Fleet Types:

- `COMPANY`
- `INDIVIDUAL`

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Demo Fleet Pvt Ltd",
    "city": "GURGAON",
    "mobile": "9876543210",
    "status": "ACTIVE"
  }
}
```

GET `/fleets`

Get all fleets.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Demo Fleet Pvt Ltd",
      "city": "GURGAON",
      "status": "ACTIVE"
    }
  ]
}
```

GET `/fleets/:id`

Get fleet by ID.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Demo Fleet Pvt Ltd",
    "city": "GURGAON",
    "mobile": "9876543210",
    "panNumber": "ABCDE1234F",
    "status": "ACTIVE"
  }
}
```

PATCH `/fleets/:id/deactivate`

Deactivate a fleet.

Authentication: Required
Roles: `SUPER_ADMIN`

Response:

```json
{
  "success": true,
  "message": "Fleet deactivated successfully"
}
```

POST `/fleets/:id/hubs`

Create a hub for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```json
{
  "name": "Gurgaon Hub",
  "location": "Sector 29, Gurgaon",
  "city": "GURGAON"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Gurgaon Hub",
    "fleetId": "uuid",
    "location": "Sector 29, Gurgaon",
    "city": "GURGAON"
  }
}
```

GET `/fleets/:id/hubs`

Get all hubs for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Gurgaon Hub",
      "location": "Sector 29, Gurgaon",
      "city": "GURGAON"
    }
  ]
}
```

POST `/fleets/:id/hub-managers`

Create a hub manager for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "userId": "uuid",
  "name": "Manager Name",
  "phone": "9876543210"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "userId": "uuid",
    "fleetId": "uuid",
    "name": "Manager Name",
    "phone": "9876543210"
  }
}
```

GET `/fleets/:id/hub-managers`

Get all hub managers for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Manager Name",
      "phone": "9876543210"
    }
  ]
}
```

GET `/fleets/hub-manager/:id`

Get hub manager by ID.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "name": "Manager Name",
    "phone": "9876543210",
    "fleetId": "uuid"
  }
}
```

POST `/fleets/hubs/:hubId/assign-manager`

Assign a manager to a hub.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "managerId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Manager assigned to hub successfully"
}
```

POST `/fleets/hubs/:id/add-vehicle`

Add a vehicle to a hub.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "vehicleId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Vehicle added to hub successfully"
}
```

POST `/fleets/hubs/:id/add-driver`

Add a driver to a hub.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "driverId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Driver added to hub successfully"
}
```

POST `/fleets/hubs/:id/remove-vehicle`

Remove a vehicle from a hub.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "vehicleId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Vehicle removed from hub successfully"
}
```

POST `/fleets/hubs/:id/remove-driver`

Remove a driver from a hub.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "driverId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Driver removed from hub successfully"
}
```

GET `/drivers/:id/preference`

Get current driver preferences.

Authentication: Required **Roles:** `DRIVER`

Response:

```
{
  "success": true,
  "data": {
    "preferences": {
      "smokingAllowed": false,
      "musicPreference": "Rock",
      "acPreferred": true
    }
  }
}
```

POST `/drivers/:id/preference/update`

Request a change in driver preferences.

Authentication: Required **Roles:** `DRIVER`

Request Body:

```
{
  "requestedPreference": {
    "smokingAllowed": true
  }
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "status": "PENDING",
    "requestedPreference": { "smokingAllowed": true }
  },
  "message": "Preference change request submitted successfully"
}
```

GET `/drivers/preference/pending-requests`

Get all pending preference change requests.

Authentication: Required **Roles:** `SUPER_ADMIN` , `OPERATIONS`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "driverId": "uuid",
      "driverName": "Rajesh Kumar",
      "currentPreference": { "smokingAllowed": false },
      "requestedPreference": { "smokingAllowed": true },
      "status": "PENDING",
      "requestAt": "2026-01-08T10:00:00Z"
    }
  ]
}
```

POST `/drivers/preference/update-status`

Approve or reject a preference change request.

Authentication: Required **Roles:** `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "requestId": "uuid",
  "status": "APPROVED",
  "rejectionReason": "Optional reason if rejected"
}
```

Response:

```
{
  "success": true,
  "message": "Preference request status updated successfully"
}
```

Vehicles

All vehicle endpoints require authentication.

POST `/vehicles`

Create a new vehicle.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "fleetId": "uuid",
  "vehicleNumber": "DL01AB1234",
  "vehicleName": "Tata Tigor EV",
  "vehicleModel": "Tigor EV 2024",
  "vehicleColor": "White",
  "fuelType": "ELECTRIC",
```

```json
  "ownership": "OWNED",
  "ownerName": "Fleet Owner Name",
  "status": "ACTIVE",

  "chassisNumber": "MA1AB2CD3EF456789",
  "vinNumber": "1HGBH41JXMN109186",

  "rcFrontImage": "https://example.com/rc-front.jpg",
  "rcBackImage": "https://example.com/rc-back.jpg",
  "permitImage": "https://example.com/permit.jpg",
  "permitExpiry": "2034-12-31",
  "fitnessImage": "https://example.com/fitness.jpg",
  "fitnessExpiry": "2026-12-31",
  "insuranceImage": "https://example.com/insurance.jpg",
  "insuranceStart": "2024-01-01",
  "insuranceExpiry": "2025-12-31"
}
```

Vehicle Identification Fields:

Field	Description
`chassisNumber`	Vehicle chassis number (stamped on frame)
`vinNumber`	Vehicle Identification Number (17-character unique identifier)

Insurance Fields:

Field	Description
`insuranceImage`	URL to insurance document image
`insuranceStart`	Insurance policy start date
`insuranceExpiry`	Insurance policy end/expiry date

Fuel Types:

- `PETROL`
- `DIESEL`
- `CNG`
- `ELECTRIC`

Ownership Types:

- `OWNED`
- `LEASED`

Vehicle Status:

- `ACTIVE`
- `INACTIVE`
- `MAINTENANCE`

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "VEH20260123001",
    "vehicleNumber": "DL01AB1234",
    "vehicleName": "Tata Tigor EV",
    "status": "ACTIVE"
  }
}
```

GET `/vehicles/fleet/:fleetId`

Get all vehicles for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "VEH20260123001",
      "vehicleNumber": "DL01AB1234",
      "vehicleName": "Tata Tigor EV",
      "status": "ACTIVE"
    }
  ]
}
```

GET `/vehicles/hub/:hubId`

Get all vehicles for a hub.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "VEH20260123001",
      "vehicleNumber": "DL01AB1234",
      "vehicleName": "Tata Tigor EV",
      "hubId": "uuid"
    }
  ]
}
```

GET `/vehicles/:id`

Get vehicle by ID.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "VEH20260123001",
    "vehicleNumber": "DL01AB1234",
    "vehicleName": "Tata Tigor EV",
    "vehicleModel": "Tigor EV 2024",
    "fuelType": "ELECTRIC",
    "status": "ACTIVE"
  }
}
```

PATCH `/vehicles/:id`

Update vehicle details.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "vehicleName": "Tata Tigor EV",
  "vehicleModel": "Tigor EV 2024",
  "vehicleColor": "White"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "vehicleName": "Tata Tigor EV"
  }
}
```

GET `/drivers/:id/preference`

Get driver preferences.

Authentication: Required
Roles: `DRIVER`

Response:

```json
{
  "success": true,
  "statusCode": 200,
  "message": "Driver preferences retrieved successfully",
  "data": [
    {
      "key": "prefer_airport_rides",
      "displayName": "Prefer airport rides",
      "description": "Prioritize airport pickup and drop trips",
      "category": "TRIP",
      "approvalRequired": true,
      "value": true
    }
  ]
}
```

POST `/drivers/:id/preference/update`

Request preference change.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "prefer_airport_rides": true,
  "accept_rentals": true,
  "auto_assign_rides": true
}
```

Response:

```json
{
  "success": true,
  "message": "Preference change request submitted successfully"
}
```

GET `/drivers/preference/pending-requests`

Get pending preference requests.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Pending preference requests retrieved successfully",
  "data": [
    {
      "id": "bd3c2df9-d58d-4b5a-8d20-2ecd8db1b63e",
      "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"
    }
  ]
}
```

POST `/drivers/preference/update-status`

Update preference request status.

Authentication: Required
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"
}
```

Response:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Preference change request updated successfully",
  "data": {
    "id": "bd3c2df9-d58d-4b5a-8d20-2ecd8db1b63e",
    "status": "REJECTED",
    "rejectionReason": "demo test"
  }
}
```

PATCH `/vehicles/:id/docs`

Update vehicle documents.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

```
{
  "rcFrontImage": "https://example.com/rc-front.jpg",
  "rcBackImage": "https://example.com/rc-back.jpg",
  "permitImage": "https://example.com/permit.jpg",
  "permitExpiry": "2034-12-31"
}
```

Response:

```
{
  "success": true,
  "message": "Vehicle documents updated successfully"
}
```

PATCH `/vehicles/:id/status`

Update vehicle status.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

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

PATCH `/vehicles/:id/deactivate`

Deactivate a vehicle.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Response:

```
{
  "success": true,
  "message": "Vehicle deactivated successfully"
}
```

PATCH_DOCS `/vehicles/:id/docs`

Update vehicle documents.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

```
{
  "rcFrontImage": "https://example.com/rc-front.jpg",
  "rcBackImage": "https://example.com/rc-back.jpg",
  "permitImage": "https://example.com/permit.jpg",
  "permitExpiry": "2034-12-31",
  "fitnessImage": "https://example.com/fitness.jpg",
  "fitnessExpiry": "2026-12-31",
  "insuranceImage": "https://example.com/insurance.jpg",
  "insuranceExpiry": "2025-12-31"
}
```

Response:

```
{
  "success": true,
  "message": "Vehicle documents updated successfully"
}
```

PATCH_STATUS `/vehicles/:id/status`

Update vehicle status.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

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

Status Values:

- `ACTIVE`
- `INACTIVE`
- `MAINTENANCE`

Response:

```
{
  "success": true,
  "message": "Vehicle status updated successfully"
}
```

PATCH_DEACTIVATE `/vehicles/:id/deactivate`

Deactivate a vehicle.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

```
{
  "success": true,
  "message": "Vehicle deactivated successfully"
}
```

Assignments

All assignment endpoints require authentication.

POST `/assignments`

Create a driver-vehicle assignment.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Request Body:

```json
{
  "fleetId": "uuid",
  "driverId": "uuid",
  "vehicleId": "uuid",
  "status": "ACTIVE"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "fleetId": "uuid",
    "driverId": "uuid",
    "vehicleId": "uuid",
    "status": "ACTIVE"
  }
}
```

GET `/assignments/fleet/:fleetId`

Get all assignments for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "driver": {
        "id": "uuid",
        "firstName": "Rajesh",
        "lastName": "Kumar"
      },
      "vehicle": {
        "id": "uuid",
        "vehicleNumber": "DL01AB1234"
      },
      "status": "ACTIVE"
    }
  ]
}
```

GET `/assignments/trip/:tripId`

Get assignments for a specific trip.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "tripId": "uuid",
      "driverId": "uuid",
      "vehicleId": "uuid"
    }
  ]
}
```

PATCH `/assignments/:id/end`

End an assignment.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response:

```json
{
  "success": true,
  "message": "Assignment ended successfully"
}
```

Attendance

All attendance endpoints require authentication.

POST `/attendance/check-in`

Driver check-in.

Authentication: Required
Roles: Any authenticated user

Request Body:

```
{
  "driverId": "uuid",
  "location": {
    "latitude": 28.4595,
    "longitude": 77.0266
  },
  "timestamp": "2025-12-26T10:00:00.000Z"
}
```

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "ATT20260123001",
    "driverId": "uuid",
    "checkInTime": "2025-12-26T10:00:00.000Z",
    "status": "PENDING"
  }
}
```

GET `/drivers/upload-url`

Generate presigned S3 URL for secure file uploads.

Authentication: Required
Roles: `DRIVER` , `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Query Parameters:

Parameter	Type	Required	Description
folder	string	Yes	`selfies` , `odometer` , `documents` , `profiles` , `vehicles`
fileType	string	Yes	`jpg` , `jpeg` , `png` , `pdf`

Request Example:

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

Response (200):

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

Upload Flow:

1. Request presigned URL from this endpoint
2. Upload file to `uploadUrl` using PUT request
3. Use the `url` field in other APIs (e.g., `selfieUrl` , `odometerImageUrl`)

Notes:

- Presigned URLs expire in 5 minutes
- Direct S3 upload - no file passes through backend

POST `/attendance/check-out`

Driver check-out.

Authentication: Required
Roles: Any authenticated user

Request Body:

```
{
  "attendanceId": "uuid",
  "location": {
    "latitude": 28.4595,
    "longitude": 77.0266
  },
  "odometer": 10500,
  "odometerImageUrl": "https://s3.aws.com/bucket/odometer.jpg",
  "cashDeposited": 500,
  "timestamp": "2025-12-26T18:00:00.000Z"
}
```

Note: `odometerImageUrl` is optional. Upload image using `/drivers/upload-url` first.

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "checkOutTime": "2025-12-26T18:00:00.000Z"
  }
}
```

POST `/attendance/start-break`

Start a break during active attendance.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "attendanceId": "uuid"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "BRK20260123001",
    "attendanceId": "uuid",
    "startTime": "2025-12-26T12:00:00.000Z"
  }
}
```

POST `/attendance/end-break`

End current break.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "breakId": "uuid"
}
```

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "BRK20260123001",
    "endTime": "2025-12-26T12:30:00.000Z"
  }
}
```

GET `/attendance/history`

Get attendance history.

Notes:

- Drivers can call this without `driverId` to fetch their own history

Authentication: Required
Roles: Any authenticated user

Query Parameters:

- `driverId` (optional) - Filter by driver
- `startDate` (optional) - Start date
- `endDate` (optional) - End date

Response:

```json
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "ATT20260123001",
      "driverId": "uuid",
      "checkInTime": "2025-12-26T10:00:00.000Z",
      "checkOutTime": "2025-12-26T18:00:00.000Z",
      "status": "APPROVED"
    }
  ]
}
```

POST `/attendance/:id/approve`

Approve attendance record.

Authentication: Required
Roles: `SUPER_ADMIN`, `MANAGER`

Response:

```json
{
  "success": true,
  "message": "Attendance approved successfully"
}
```

POST `/attendance/:id/reject`

Reject attendance record.

Authentication: Required
Roles: `SUPER_ADMIN`, `MANAGER`

Request Body:

```
{
  "reason": "Invalid check-in location"
}
```

Response:

```
{
  "success": true,
  "message": "Attendance rejected successfully"
}
```

Trips

All trip endpoints require authentication.

POST `/trips`

Create a new trip.

Authentication: Required

Request Body:

```
{
  "pickupLocation": "Sector 29, Gurgaon",
  "pickupLat": 28.4595,
  "pickupLng": 77.0266,
  "dropLocation": "IGI Airport, Delhi",
  "dropLat": 28.5562,
  "dropLng": 77.1000,
  "pickupTime": "2025-12-26T14:00:00.000Z",
  "tripType": "INTER_CITY",
  "bookingType": "PREBOOK",
  "vehicleSku": "DNA_SEDAN_EV",
  "passengerName": "John Doe",
  "passengerPhone": "9876543210",
  "distanceKm": 25.5
}
```

Booking Types:

- `PREBOOK` - Scheduled (> 2.5 hours in advance)

- `INSTANT` - Immediate booking

Response:

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "TRP20260123001",
    "pickupLocation": {
      "address": "Sector 29, Gurgaon",
      "latitude": 28.4595,
      "longitude": 77.0266
    },
    "dropLocation": {
      "address": "IGI Airport, Delhi",
      "latitude": 28.5562,
      "longitude": 77.1000
    },
    "pickupTime": "2025-12-26T14:00:00.000Z",
    "status": "PENDING",
    "estimatedFare": 625,
    "distance": 25.5
  }
}
```

GET `/trips`

Get trips for current driver.

Authentication: Required
Roles: `DRIVER`

Query Parameters:

- `status` (optional) - Filter by status
- `date` (optional) - Filter by date

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "TRP20260123001",
      "pickupLocation": {
        "address": "Sector 29, Gurgaon"
      },
      "dropLocation": {
        "address": "IGI Airport, Delhi"
      },
      "pickupTime": "2025-12-26T14:00:00.000Z",
      "status": "DRIVER_ASSIGNED",
      "estimatedFare": 625
    }
  ]
}
```

GET `/trips/:id`

Get trip details by ID.

Authentication: Required

Response:

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "shortId": "TRP20260123001",
    "pickupLocation": {
      "address": "Sector 29, Gurgaon",
      "latitude": 28.4595,
      "longitude": 77.0266
    },
    "dropLocation": {
      "address": "IGI Airport, Delhi",
      "latitude": 28.5562,
      "longitude": 77.1000
    },
    "pickupTime": "2025-12-26T14:00:00.000Z",
    "status": "DRIVER_ASSIGNED",
    "driver": {
      "id": "uuid",
      "shortId": "DRV20260123001",
      "firstName": "Rajesh",
      "lastName": "Kumar",
      "mobile": "9876543210"
    },
    "vehicle": {
      "id": "uuid",
      "shortId": "VEH20260123001",
      "vehicleNumber": "DL01AB1234"
    },
    "estimatedFare": 625,
    "distance": 25.5
  }
}
```

POST `/trips/:id/assign`

Assign driver to trip.

Authentication: Required

Request Body:

```
{
  "driverId": "uuid"
}
```

Response:

```
{
  "success": true,
  "message": "Driver assigned successfully",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "driverId": "uuid",
    "status": "DRIVER_ASSIGNED"
  }
}
```

POST `/trips/:id/start`

Start a trip (driver en route to pickup).

Authentication: Required
Roles: `DRIVER`

Request Body:

```
{
  "location": {
    "latitude": 28.4595,
    "longitude": 77.0266
  },
  "timestamp": "2025-12-26T13:30:00.000Z"
}
```

Constraints:

- Must be within 2.5 hours of pickup time
- Trip must be in `DRIVER_ASSIGNED` status

Response:

```
{
  "success": true,
  "message": "Trip started successfully",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "STARTED",
    "pickupTime": "2026-01-24T16:00:00Z",
    "customerName": "Amit Sharma",
    "customerPhone": "9876543210",
    "paymentMethod": "CASH"
  }
}
```

POST `/trips/:id/arrived`

Mark driver as arrived at pickup location.

Authentication: Required
Roles: `DRIVER`

Request Body:

```
{
  "location": {
    "latitude": 28.4595,
    "longitude": 77.0266
  },
  "timestamp": "2025-12-26T13:55:00.000Z"
}
```

Constraints:

- **Geofence**: Must be within **500 meters** of pickup location.
- **Time Window**: Cannot arrive more than **30 minutes** before `pickupTime` .

Error Responses:

- `400 Bad Request` : "You are Xm away from pickup. Please reach within 500m."
- `400 Bad Request` : "Too early to mark Arrived. Please wait until 30 mins before pickup."
- `403 Forbidden` : "Not authorized to update this trip"

Response:

- Must be within 30 minutes of pickup time
- Must be within 500m radius of pickup location
- Trip must be in `STARTED` status

Response:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Driver marked as arrived",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "ARRIVED_EVENT_SENT",
    "arrivedAt": "2026-01-24T05:53:24.389Z",
    "pickupLocation": "Terminal 1, IGI Airport",
    "pickupLat": 28.5550838,
    "pickupLng": 77.0844015
  }
}
```

POST `/trips/:id/onboard`

Mark passenger as onboarded.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "otp": "1234",
  "timestamp": "2025-12-26T14:00:00.000Z"
}
```

Response:

```json
{
  "success": true,
  "message": "Passenger onboarded successfully",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "ONBOARDED"
  }
}
```

POST `/trips/:id/noshow`

Mark trip as no-show.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "reason": "Passenger did not arrive",
  "timestamp": "2025-12-26T14:30:00.000Z"
}
```

Constraints:

- **Time Window**: Can only mark No Show **30 minutes** after `pickupTime` .

Error Responses:

- `400 Bad Request` : "Cannot mark No Show yet. Wait until 30 mins after pickup time."
- `404 Not Found` : "Trip not found"

Response:

```json
{
  "success": true,
  "message": "Trip marked as no-show",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "NO_SHOW"
  }
}
```

POST `/trips/:id/complete`

Complete a trip.

Authentication: Required
Roles: `DRIVER`

Request Body:

```json
{
  "location": {
    "latitude": 28.5562,
    "longitude": 77.1000
  },
  "actualDistance": 26.2,
  "timestamp": "2025-12-26T15:00:00.000Z"
}
```

Note: Extra charges (Waiting, Night, Airport) are **auto-calculated** by the backend based on timestamps and location. Manual values in `extraCharges` for these fields will be overridden.

Response:

```json
{
  "success": true,
  "message": "Trip completed successfully",
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "COMPLETED",
    "actualDistance": 26.2,
    "finalFare": 650
  }
}
```

GET `/trips/:id/tracking`

Get real-time trip tracking information.

Authentication: Required

Response:

```
{
  "success": true,
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "ONBOARDED",
    "driver": {
      "name": "Rajesh Kumar",
      "phone": "9876543210",
      "currentLocation": {
        "latitude": 28.5000,
        "longitude": 77.0500
      }
    },
    "vehicle": {
      "number": "DL01AB1234",
      "model": "Tata Tigor EV"
    },
    "eta": "2025-12-26T15:00:00.000Z"
  }
}
```

Admin Trips

All admin trip endpoints require authentication and `SUPER_ADMIN` role.

GET `/admin/trips`

Get all trips (admin view).

Authentication: Required
Roles: `SUPER_ADMIN`

Query Parameters:

- `status` (optional) - Filter by status
- `date` (optional) - Filter by date
- `driverId` (optional) - Filter by driver
- `fleetId` (optional) - Filter by fleet

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "shortId": "TRP20260123001",
      "pickupLocation": {
        "address": "Sector 29, Gurgaon"
      },
      "dropLocation": {
        "address": "IGI Airport, Delhi"
      },
      "status": "COMPLETED",
      "driver": {
        "id": "uuid",
        "firstName": "Rajesh",
        "lastName": "Kumar"
      },
      "estimatedFare": 625,
      "finalFare": 650
    }
  ]
}
```

POST `/admin/trips/assign`

Manually assign driver to trip (admin).

Authentication: Required
Roles: `SUPER_ADMIN`

Request Body:

```
{
  "tripId": "uuid",
  "driverId": "uuid"
}
```

Response:

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

POST `/admin/trips/unassign`

Unassign driver from trip.

Authentication: Required
Roles: `SUPER_ADMIN`

Request Body:

```json
{
  "tripId": "uuid"
}
```

Response:

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

POST `/admin/trips/reassign`

Reassign trip to different driver.

Authentication: Required
Roles: `SUPER_ADMIN`

Request Body:

```json
{
  "tripId": "uuid",
  "newDriverId": "uuid"
}
```

Response:

```json
{
  "success": true,
  "message": "Trip reassigned successfully"
}
```

POST `/admin/trips/detach`

Detach a driver from a trip (Vendor Cancellation). This triggers an irreversible cancellation flow with external partners (e.g., MMT) if applicable.

Authentication: Required
Roles: `SUPER_ADMIN` , `FLEET_ADMIN` , `OPERATIONS` , `MANAGER`

Request Body:

```json
{
  "tripId": "uuid",
  "reason": "Driver vehicle breakdown",
  "comment": "Optional comment for audit"
}
```

Response:

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

Pricing

POST `/pricing/preview`

Get pricing preview for a trip.

Authentication: None

Request Body:

```
{
  "pickup": "Sector 29, Gurgaon",
  "drop": "IGI Airport, Delhi",
  "tripType": "AIRPORT",
  "tripDate": "2025-12-26T14:00:00.000Z",
  "bookingDate": "2025-12-25T14:00:00.000Z",
  "vehicleType": "EV",
  "vehicleSku": "TATA_TIGOR_EV" // Optional if vehicleType is provided
}
```

Response:

```
{
  "success": true,
  "data": {
    "estimatedFare": 625,
    "distance": 25.5,
    "duration": 45,
    "breakdown": {
      "baseFare": 100,
      "distanceFare": 450,
      "timeFare": 75,
      "taxes": 0,
      "total": 625
    }
  }
}
```

Public Booking API

Public endpoints for customers to book trips without authentication.

POST `/public/trips/pricing`

Get pricing estimate for a trip (no authentication required).

Authentication: None

Request Body:

```json
{
  "pickupLocation": "Connaught Place, New Delhi",
  "dropLocation": "IGI Airport, Terminal 3",
  "tripDate": "2026-01-25",
  "tripTime": "10:00",
  "tripType": "AIRPORT"
}
```

Trip Types: `AIRPORT`, `INTER_CITY`, `RENTAL`

Response (200):

```json
{
  "success": true,
  "data": {
    "vehicleSku": "TATA_TIGOR_EV",
    "vehicleType": "EV",
    "tripType": "AIRPORT",
    "distanceKm": 25.5,
    "billableDistanceKm": 26,
    "ratePerKm": 25,
    "baseFare": 650,
    "totalFare": 780,
    "breakdown": {
      "distanceFare": 650,
      "tripTypeMultiplier": 1.2,
      "bookingTimeMultiplier": 1.0,
      "vehicleMultiplier": 1.0
    },
    "currency": "INR"
  },
  "message": "Fare calculated successfully"
}
```

Constraints Applied:

- T-1 Booking: Trip must be > 24 hours from now
- Distance limits based on trip type
- City-specific rules

POST `/public/trips/create`

Create a trip booking (no authentication required).

Authentication: None

Request Body:

```json
{
  "pickupLocation": "Connaught Place, New Delhi",
  "dropLocation": "IGI Airport, Terminal 3",
  "tripDate": "2026-01-25",
  "tripTime": "10:00",
  "tripType": "AIRPORT",
  "customerName": "John Doe",
  "customerPhone": "9876543210"
}
```

Response (201):

```json
{
  "success": true,
  "data": {
    "tripId": "uuid",
    "shortId": "TRP20260123001",
    "status": "CREATED",
    "vehicleSku": "TATA_TIGOR_EV",
    "vehicleType": "EV",
    "tripType": "AIRPORT",
    "price": 780,
    "pickupLocation": "Connaught Place, New Delhi",
    "dropLocation": "IGI Airport, Terminal 3",
    "tripDate": "2026-01-25T10:00:00.000Z",
    "customerName": "John Doe",
    "customerPhone": "9876543210"
  },
  "message": "Trip created successfully"
}
```

Note: The system automatically:

- Calculates distance via Google Maps
- Extracts city from address
- Gets coordinates for geofencing
- Stores customer info in provider metadata

Referral System

Driver referral system with automated reward processing.

How Referrals Work

1. **Driver A** gets a unique referral code automatically upon signup
2. **Driver B** signs up using Driver A's referral code
3. System tracks Driver B's activity (active days, rides per day)
4. When Driver B meets eligibility criteria, Driver A receives an incentive

Eligibility Criteria (Configurable)

- **Minimum Active Days:** 30 days (with approved attendance)
- **Minimum Rides Per Day:** 8 completed trips on average
- **Max Referrals Per Driver:** 50 (limit per referrer)

Referral Statuses

Status	Description
`PENDING`	Referral created, waiting for eligibility check
`COMPLETED`	Referral reward has been processed
`CANCELLED`	Referral was cancelled
`EXPIRED`	Referral expired without completion

How Referral Code is Used

During driver signup (`POST /users/drivers/signup`), include the referral code:

```
{
  "name": "New Driver",
  "phone": "9876543210",
  "referralCode": "REF123ABC"
}
```

The system validates:

- Referral code exists and is active
- Referrer hasn't exceeded max referrals
- New user isn't already referred

Reward Processing

Rewards are processed automatically via a background worker that:

1. Checks all `PENDING` referrals periodically
2. Validates eligibility criteria for each referred driver
3. Creates `REFERRAL` category incentive for referrer
4. Updates referral status to `COMPLETED`

Partner Integrations

MakeMyTrip (MMT)

Complete integration with MakeMyTrip for cab bookings.

Authentication: Basic Auth
Inbound Credentials: `MMT_INBOUND_USERNAME` / `MMT_INBOUND_PASSWORD`
Outbound Credentials: `MMT_TRACKING_USER` / `MMT_TRACKING_PASS`

MMT Inbound Endpoints (MMT → DriversKlub)

These endpoints receive requests FROM MakeMyTrip TO our system.

POST `/partners/mmt/partnersearchendpoint`

Search for available vehicles and get fare estimates.

Authentication: Basic Auth (`MMT_INBOUND_USERNAME` / `MMT_INBOUND_PASSWORD`)

Request Body:

```json
{
  "start_time": "2026-01-25 20:00:00",
  "trip_type_details": {
    "basic_trip_type": "AIRPORT",
    "airport_type": "AP"
  },
  "one_way_distance": 45,
  "source": {
    "city": "DELHI",
    "lat": 28.5562,
    "lng": 77.1,
    "address": "IGI Airport T3"
  },
  "destination": {
    "city": "GURGAON",
    "lat": 28.4595,
    "lng": 77.0266,
    "address": "Cyber Hub"
  },
  "partner_name": "GOMMT",
  "search_id": "search_123456"
}
```

Response:

```json
{
  "response": {
    "distance_booked": 45,
    "is_instant_search": false,
    "is_instant_available": true,
    "start_time": "2026-01-25 20:00:00",
    "is_part_payment_allowed": true,
    "communication_type": "PRE",
    "verification_type": "OTP",
    "airport_tags": ["AP"],
    "car_types": [
      {
        "sku_id": "TATA_TIGOR_EV",
        "type": "sedan",
        "subcategory": "basic",
        "combustion_type": "Electric",
        "model": "Tata Tigor",
        "carrier": false,
        "make_year_type": "Newer",
        "make_year": 2024,
        "cancellation_rule": "SUPER_FLEXI",
        "min_payment_percentage": 20,
        "pax_capacity": 4,
        "luggage_capacity": 2,
        "zero_payment": false,
        "amenities": {
          "features": {
            "vehicle": ["AC", "Music System", "Charging Point"],
            "driver": ["Vaccinated", "Mask"],
            "services": []
          }
        },
        "fare_details": {
          "base_fare": 1125,
          "per_km_charge": 24,
          "per_km_extra_charge": 25,
          "total_driver_charges": 0,
          "seller_discount": 0,
          "extra_charges": {
            "night_charges": {
              "amount": 1406.25,
              "is_included_in_base_fare": false,
              "is_included_in_grand_total": true,
              "is_applicable": true
            },
            "toll_charges": {
              "amount": 100,
              "is_included_in_base_fare": false,
              "is_included_in_grand_total": false,
              "is_applicable": true
            }
          }
        }
      }
    ]
  }
}
```

POST `/partners/mmt/partnerblockendpoint`

Block a vehicle for booking (holds inventory).

Authentication: Basic Auth

Request Body:

```
{
  "sku_id": "TATA_TIGOR_EV",
  "mmt_ref_id": "BKS88888800926",
  "start_time": "2026-01-25 20:00:00",
  "one_way_distance": 45,
  "source": {
    "city": "DELHI",
    "lat": 28.5562,
    "lng": 77.1,
    "address": "IGI Airport T3"
  },
  "destination": {
    "city": "GURGAON",
    "lat": 28.4595,
    "lng": 77.0266,
    "address": "Cyber Hub"
  }
}
```

Response:

```
{
  "response": {
    "success": true,
    "reference_number": "8863a9d8-b133-4050-9f47-ee8e7534b6bc",
    "status": "BLOCKED",
    "verification_code": "1056"
  }
}
```

Field	Description
`reference_number`	Our internal Trip ID (UUID)
`verification_code`	OTP for passenger verification

POST `/partners/mmt/partnerpaidendpoint`

Confirm booking payment and finalize the trip.

Authentication: Basic Auth

Request Body:

```
{
  "partner_reference_number": "8863a9d8-b133-4050-9f47-ee8e7534b6bc",
  "order_reference_number": "BKS88888800926",
  "total_fare": 1125,
  "amount_to_be_collected": 900,
  "platform_fee": 0,
  "booking_gst": 0,
  "partner_name": "GOMMT",
  "passenger": {
    "name": "John Doe",
    "email": "john@example.com",
    "phone_number": "9876543210"
  }
}
```

Response:

```
{
  "response": {
    "success": true,
    "order_reference_number": "BKS88888800926",
    "status": "CONFIRMED"
  }
}
```

Important: The `order_reference_number` from MMT becomes the `providerBookingId` used for all tracking events.

POST `/partners/mmt/partnercancelendpoint`

Cancel a booking.

Authentication: Basic Auth

Request Body:

```
{
  "order_reference_number": "BKS88888800926",
  "partner_reference_number": "8863a9d8-b133-4050-9f47-ee8e7534b6bc",
  "cancellation_reason": "Customer request",
  "cancelled_by": "CUSTOMER"
}
```

Response:

```
{
  "response": {
    "success": true,
    "status": "CANCELLED"
  }
}
```

GET `/partners/mmt/api/partner/v1/booking/details`

Get booking details by ID.

Authentication: Basic Auth

Query Parameters:

Parameter	Description
`booking_id`	Our internal Trip ID (UUID)
`partner_reference_number`	Our internal Trip ID
`order_reference_number`	MMT's booking ID (BKS...)

Response:

```
{
  "response": {
    "success": true,
    "booking_id": "8863a9d8-b133-4050-9f47-ee8e7534b6bc",
    "status": "CONFIRMED",
    "start_time": "2026-01-25T20:00:00.000Z",
    "source": {
      "city": "DELHI",
      "address": "IGI Airport T3",
      "lat": 28.5562,
      "lng": 77.1
    },
    "destination": {
      "city": "GURGAON",
      "address": "Cyber Hub",
      "lat": 28.4595,
      "lng": 77.0266
    },
    "fare_details": {
      "total_amount": 1125,
      "payable_amount": 900
    },
    "driver_details": {
      "name": "Rajesh Kumar",
      "phone": "9876543210",
      "image": "https://..."
    },
    "vehicle_details": {
      "registration_number": "DL01AB1234",
      "model": "Tata Tigor EV",
      "color": "White"
    },
    "verification_code": "1056"
  }
}
```

POST `/partners/mmt/partnerrescheduleblockendpoint`

Validate reschedule request and get new fare.

Authentication: Basic Auth

Request Body:

```
{
  "order_reference_number": "BKS88888800926",
  "partner_reference_number": "8863a9d8-b133-4050-9f47-ee8e7534b6bc",
  "start_time": "2026-01-26 10:00:00"
}
```

Response:

```
{
  "response": {
    "success": true,
    "verification_code": "1056",
    "fare_details": {
      "total_amount": 1125,
      "payable_amount": 900,
      "fare_difference": 0
    },
    "driver_details": {
      "name": "Rajesh Kumar",
      "phone": "9876543210"
    }
  }
}
```

POST `/partners/mmt/partnerrescheduleconfirmendpoint`

Confirm reschedule after validation.

Authentication: Basic Auth

Request Body:

```
{
  "order_reference_number": "BKS88888800926",
  "partner_reference_number": "8863a9d8-b133-4050-9f47-ee8e7534b6bc"
}
```

Response:

```json
{
  "response": {
    "success": true,
    "status": "RESCHEDULED"
  }
}
```

MMT Outbound Tracking Events (DriversKlub → MMT)

These are **outbound** API calls that DriversKlub makes TO MMT to notify them about ride status changes.

Base URL: `https://cabs-partners-staging.makemytrip.com/tracking/pp2/api/partner/v1`
Authentication: Basic Auth (credentials in environment variables)

Dispatch Events

POST `/dispatch/{booking_id}/assign`

Notify MMT when a driver is assigned to a booking.

Triggered by: `POST /admin/trips/assign` or `POST /trips/:id/assign`

Request Body (sent to MMT):

```json
{
  "booking_id": "BKS88888800922",
  "chauffeur": {
    "id": "56c5c8a269",
    "name": "Rajesh Kumar",
    "mobile_number": "9876543210",
    "image": "https://s3.aws.com/driver-photo.jpg"
  },
  "vehicle": {
    "id": "abc1234567",
    "name": "Tata Tigor EV",
    "color": "White",
    "registration_number": "DL01AB1234",
    "vehicle_type": "sedan"
  }
}
```

Note: `chauffeur.id` and `vehicle.id` are shortened to max 10 characters (MMT requirement).

POST `/dispatch/{booking_id}/reassign`

Notify MMT when a different driver is assigned to an existing booking.

Triggered by: `POST /admin/trips/reassign`

Request Body: Same structure as assign (includes `booking_id`).

POST `/dispatch/{booking_id}/unassign`

Notify MMT when a driver is unassigned from a booking.

Triggered by: `POST /admin/trips/unassign`

Request Body (sent to MMT):

```
{
  "booking_id": "BKS88888800922"
}
```

Track Events

All track events require the following mandatory fields:

- `booking_id` - MMT booking reference (e.g., "BKS88888800922")
- `device_id` - Shortened driver ID (max 10 chars) or "DKAPP001"
- `latitude` - **String** (e.g., "28.4895122")
- `longitude` - **String** (e.g., "77.092306")
- `timestamp` - Unix timestamp in milliseconds

POST `/track/{booking_id}/start`

Notify MMT when the driver starts the trip.

Triggered by: `POST /trips/:id/start`

Request Body (sent to MMT):

```
{
  "booking_id": "BKS88888800922",
  "device_id": "56c5c8a269",
  "latitude": "28.4895122",
  "longitude": "77.092306",
  "timestamp": 1706025600000
}
```

POST `/track/{booking_id}/arrived`

Notify MMT when driver arrives at pickup location.

Triggered by: `POST /trips/:id/arrived`

Request Body: Same structure as start.

POST `/track/{booking_id}/boarded`

Notify MMT when passenger boards the vehicle.

Triggered by: `POST /trips/:id/onboard`

Request Body: Same structure as start.

POST `/track/{booking_id}/alight`

Notify MMT when passenger alights (trip complete).

Triggered by: `POST /trips/:id/complete`

Request Body (sent to MMT):

```
{
  "booking_id": "BKS88888800922",
  "device_id": "56c5c8a269",
  "latitude": "28.5562",
  "longitude": "77.1000",
  "timestamp": 1706029200000,
  "extra_charge": 485,
  "extra_fare_breakup": {
    "toll_charges": {
      "amount": 335,
      "items": [{ "name": "Toll Charges", "amount": 335, "receipt": null }]
    },
    "parking_charges": {
      "amount": 150,
      "items": [{ "name": "Parking Charges", "amount": 150, "receipt": null }]
    }
  }
}
```

Note: `extra_charge` and `extra_fare_breakup` are optional - only included if applicable.

POST `/track/{booking_id}/not-boarded`

Notify MMT when passenger is a no-show.

Triggered by: `POST /trips/:id/noshow`

Request Body (sent to MMT):

```
{
  "booking_id": "BKS88888800922",
  "device_id": "56c5c8a269",
  "latitude": "28.4895122",
  "longitude": "77.092306",
  "timestamp": 1706025600000,
  "reason": "Customer no show"
}
```

PUT `/track/{booking_id}/location`

Send live driver location during an active trip.

Triggered by: `POST /trips/:id/location`

Frequency: Should be called every 30 seconds from `start` until `alight`.

Request Body (sent to MMT):

```
{
  "booking_id": "BKS88888800926",
  "device_id": "b5405f3e0e",
  "latitude": "28.5000",
  "longitude": "77.0500",
  "timestamp": 1769175500000
}
```

MMT Environment Variables

```
# Inbound Authentication (MMT → DriversKlub)
MMT_INBOUND_USERNAME=mmt_inbound_service
MMT_INBOUND_PASSWORD=your_secure_password

# Outbound Tracking (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
```

MMT Complete Booking Flow

```
┌─────────────────────────────────────────────────────────────────┐
│                    MMT BOOKING FLOW                              │
├─────────────────────────────────────────────────────────────────┤
│                                                                  │
│  1. SEARCH      MMT → /partnersearchendpoint                    │
│                 Returns: car_types, fare_details                 │
│                                                                  │
│  2. BLOCK       MMT → /partnerblockendpoint                     │
│                 Returns: reference_number (our Trip ID), OTP     │
│                                                                  │
│  3. PAID        MMT → /partnerpaidendpoint                      │
│                 Stores: order_reference_number (BKS...)          │
│                 Returns: status: CONFIRMED                       │
│                                                                  │
│  4. ASSIGN      Admin assigns driver                             │
│                 DK → MMT /dispatch/{booking_id}/assign           │
│                                                                  │
│  5. START       Driver taps "Start Trip"                         │
│                 DK → MMT /track/{booking_id}/start               │
│                                                                  │
│  6. ARRIVED     Driver reaches pickup                            │
│                 DK → MMT /track/{booking_id}/arrived             │
│                                                                  │
│  7. BOARDED     Passenger boards (OTP verified)                  │
│                 DK → MMT /track/{booking_id}/boarded             │
│                                                                  │
│  8. LOCATION    Every 30 seconds during trip                     │
│                 DK → MMT /track/{booking_id}/location            │
│                                                                  │
│  9. ALIGHT      Trip completed                                   │
│                 DK → MMT /track/{booking_id}/alight              │
```

```
    │                                                                    │
    └────────────────────────────────────────────────────────────────────┘
```

MMT Test Case Flows

Test	Flow	Description
1	Assign → Start → Arrived → Boarded → Alight	Happy path
2	Assign → Reassign → Start → Arrived → Boarded → Alight	Driver change mid-booking
3	Assign → Start → Arrived → Not-Boarded	Customer no-show
4	Assign → Start → Arrived → Unassign	Driver unassigned after arrival
5	Assign → Unassign	Driver unassigned before trip
6	Unassign	Unassign without prior assignment
7	Full flow with extra_charge in Alight	Toll/parking charges
8	Location updates every 30 seconds	Start to Alight

POST `/trips/:id/location`

Update driver live location for a trip.

Authentication: Required **Roles:** `DRIVER`

Request Body:

```
{
  "lat": 28.5355,
  "lng": 77.3910
}
```

Response:

```
{
  "success": true,
  "message": "Location updated successfully"
}
```

Webhooks

Webhooks allow real-time notifications for system events, primarily payment updates.

Easebuzz Payment Webhook

POST `/webhooks/easebuzz/payment`

Webhook endpoint for Easebuzz payment gateway notifications.

Authentication: Signature Verification (Salt-based)

Request Body:

- Standard Easebuzz webhook payload (FormData) including `txnid`, `status`, `amount`, `hash`, etc.

Response:

```
{
  "success": true,
  "message": "Webhook processed successfully"
}
```

POST `/webhooks/easebuzz/virtual-account`

Webhook for Virtual Account (QR Code) payment notifications.

Authentication: Signature Verification (Salt-based)

Description: Called by Easebuzz when a payment is received on a Virtual Account (Vehicle QR or Order QR).

Payment Redirect Handlers

These are public redirect URLs used after Easebuzz payment completion.

GET `/webhooks/payment/success`

Payment success redirect page. Shows success message and redirects to mobile app.

Query Parameters:

- `txnid` - Transaction ID
- `amount` - Payment amount
- `status` - Payment status

Response: HTML page with deep link to `driversklub://payment/success`

GET `/webhooks/payment/failure`

Payment failure redirect page. Shows error message and redirects to mobile app.

Query Parameters:

- `txnid` - Transaction ID
- `error_Message` - Error description

Response: HTML page with deep link to `driversklub://payment/failure`

Error Handling

All API errors follow a consistent format:

```json
{
  "success": false,
  "message": "Error message",
  "error": {
    "code": "ERROR_CODE",
    "details": "Detailed error information"
  }
}
```

Common HTTP Status Codes

- `200` - Success
- `201` - Created
- `400` - Bad Request (validation errors)
- `401` - Unauthorized (missing or invalid token)
- `403` - Forbidden (insufficient permissions)
- `404` - Not Found
- `429` - Too Many Requests (rate limit exceeded)
- `500` - Internal Server Error
- `503` - Service Unavailable (database connection issues)

Common Error Codes

- `INVALID_CREDENTIALS` - Invalid phone or OTP
- `OTP_EXPIRED` - OTP has expired
- `OTP_ATTEMPT_EXCEEDED` - Too many OTP attempts
- `UNAUTHORIZED` - Missing or invalid authentication token
- `FORBIDDEN` - Insufficient permissions
- `NOT_FOUND` - Resource not found
- `VALIDATION_ERROR` - Request validation failed
- `TRIP_CONSTRAINT_VIOLATION` - Trip timing/location constraints not met

Rate Limiting

The API implements rate limiting to prevent abuse:

- **Window:** 15 minutes
- **Limit:** 100 requests per IP
- **Headers:**
  - `X-RateLimit-Limit` - Maximum requests allowed
  - `X-RateLimit-Remaining` - Remaining requests
  - `X-RateLimit-Reset` - Time when limit resets

When rate limit is exceeded:

```json
{
  "success": false,
  "message": "Too many requests from this IP, please try again later."
}
```

Additional Notes

CORS Configuration

- **Development:** All origins allowed (`*`)
- **Production:** Specific origins from `ALLOWED_ORIGINS` environment variable

Timezone

All timestamps are in **IST (Indian Standard Time)** and follow ISO 8601 format.

Pagination

Endpoints returning lists support pagination:

Query Parameters:

- `page` - Page number (default: 1)
- `limit` - Items per page (default: 20, max: 100)

Response:

```json
{
  "success": true,
  "data": [...],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 150,
    "totalPages": 8
  }
}
```

Vehicle QR Generation

POST `/payments/admin/vehicle/:id/qr`

Generate Easebuzz virtual account QR code for a vehicle.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response (201):

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

> ***Note:*** *The `qrCodeBase64` field may contain a URL (from Easebuzz or fallback generator) or base64 string. In test mode, a fallback QR URL is generated from the UPI ID.*

GET `/payments/admin/vehicle/:id/qr`

Get existing QR code. Returns 404 if not found.

Support

For API support and questions:

- **Email:** gourav.singh@triborefin.com
- **Documentation:** `/api-docs` (Coming Soon)
- **Health Check:** `/health`

Payment System

Complete payment and payout system with Easebuzz integration.

Base Path: `/payment`

Documentation: See Payment System Documentation for complete details.

Driver Endpoints

GET `/payment/balance`

Get driver balance & rental status.

Authentication: Required
Roles: `DRIVER`

Response:

```
{
  "success": true,
  "data": {
    "depositBalance": 5000,
    "paymentModel": "RENTAL",
    "hasActiveRental": true,
    "rental": {
      "planName": "Weekly Plan",
      "amount": 2500,
      "startDate": "2025-12-23T00:00:00.000Z",
      "expiryDate": "2025-12-30T00:00:00.000Z",
      "daysRemaining": 3,
      "isExpired": false,
      "vehicle": {
        "number": "KA01 AB 1234",
        "model": "TATA Tigor EV"
      }
    }
  }
}
```

GET `/payment/rental/plans`

Get available rental plans for the driver's fleet.

Authentication: Required
Roles: `DRIVER`

Response:

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Weekly Plan",
      "rentalAmount": 3000,
      "depositAmount": 5000,
      "validityDays": 7,
      "isActive": true
    }
  ]
}
```

GET `/payment/transactions`

Get driver transaction history (paginated).

Authentication: Required
Roles: `DRIVER`

Query Parameters:

- `page` (number, default: 1)
- `limit` (number, default: 20)
- `type` (string, optional): DEPOSIT, RENTAL, TRIP_PAYMENT, INCENTIVE, PENALTY, PAYOUT
- `status` (string, optional): PENDING, SUCCESS, FAILED
- `startDate` (string, optional): ISO date
- `endDate` (string, optional): ISO date

Response:

```
{
  "transactions": [
    {
      "id": "uuid",
      "type": "DEPOSIT",
      "amount": 5000,
      "status": "SUCCESS",
      "paymentMethod": "PG_UPI",
      "description": "Security deposit - ₹5000",
      "createdAt": "2025-12-23T10:00:00.000Z"
    }
  ],
  "pagination": {
    "page": 1,
    "limit": 20,
    "total": 45,
    "totalPages": 3
  }
}
```

GET `/payment/incentives`

Get driver incentives.

Authentication: Required
Roles: `DRIVER`

Query Parameters:

- `isPaid` (boolean, optional)

Response:

```
{
  "incentives": [
    {
      "id": "uuid",
      "amount": 500,
      "reason": "Completed 50 trips",
      "category": "MILESTONE",
      "isPaid": false,
      "createdAt": "2025-12-25T00:00:00.000Z"
    }
  ],
  "summary": {
    "totalIncentives": 10,
    "paidIncentives": 7,
    "unpaidIncentives": 3,
    "totalAmount": 5000,
    "paidAmount": 3500,
    "unpaidAmount": 1500
    }
  }
```

GET `/payment/penalties`

Get driver penalties.

Authentication: Required
Roles: `DRIVER`

Query Parameters:

- `type` (string, optional): MONETARY, WARNING, SUSPENSION, BLACKLIST
- `isPaid` (boolean, optional)
- `isWaived` (boolean, optional)

Response:

```json
{
  "penalties": [
    {
      "id": "uuid",
      "type": "MONETARY",
      "amount": 200,
      "reason": "Late for pickup",
      "isPaid": true,
      "isWaived": false,
      "deductedFromDeposit": true,
      "depositDeductionAmount": 200,
      "createdAt": "2025-12-24T00:00:00.000Z"
    }
  ]
}
```

GET `/payment/collections`

Get daily collections (for payout model drivers).

Authentication: Required
Roles: `DRIVER`

Query Parameters:

- `startDate` (string, optional): ISO date
- `endDate` (string, optional): ISO date
- `isReconciled` (boolean, optional)
- `isPaid` (boolean, optional)

Response:

```json
{
  "collections": [
    {
      "id": "uuid",
      "date": "2025-12-29T00:00:00.000Z",
      "qrCollectionAmount": 3000,
      "cashCollectionAmount": 2000,
      "totalCollection": 5000,
      "revShareAmount": 3500,
      "incentiveAmount": 500,
      "penaltyAmount": 200,
      "netPayout": 3800,
      "isReconciled": true,
      "isPaid": false
    }
  ],
  "summary": {
    "totalDays": 30,
    "totalCollections": 150000,
    "totalRevShare": 105000,
    "totalIncentives": 5000,
    "totalPenalties": 2000,
    "totalPayout": 108000,
    "paidAmount": 75000,
    "unpaidAmount": 33000
  }
}
```

POST `/payment/deposit`

Initiate deposit payment.

Authentication: Required
Roles: `DRIVER`

Request Body:

```
{
  "amount": 5000
}
```

Response:

```
{
  "success": true,
  "data": {
    "transactionId": "uuid",
    "paymentUrl": "https://testpay.easebuzz.in/pay/0c4d0ab67...",
    "accessKey": "0c4d0ab671a967784530587dbca8e2c8...",
    "txnId": "TXN_1735123456_ABC123"
  }
}
```

POST `/payment/rental`

Initiate rental payment.

Authentication: Required
Roles: `DRIVER`

Request Body:

```
{
  "rentalPlanId": "uuid"
}
```

Response:

```
{
  "success": true,
  "data": {
    "transactionId": "uuid",
    "paymentUrl": "https://testpay.easebuzz.in/pay/0c4d0ab67...",
    "accessKey": "0c4d0ab671a967784530587dbca8e2c8...",
    "txnId": "TXN_1735123456_XYZ789"
  }
}
```

Admin Endpoints\r\n\r\n##### POST \r\n\r\nGenerate Easebuzz virtual account QR code for a vehicle.\r\n\r\n**Authentication:** Required \r\n**Roles:** , , \r\n\r\n**URL Parameters:**\r\n- uid=197609(Hello Youtuber) gid=197609 groups=197609 (string, required) - Vehicle UUID\r\n\r\n**Response (201):**\r\n\r\n\r\n\r\n**Notes:** QR code is Base64 PNG. If QR exists, returns existing. Scannable with any UPI app.\r\n\r\n---\r\n\r\n##### GET \r\n\r\nGet existing QR code for a vehicle.\r\n\r\n**Authentication:** Required \r\n**Roles:** , , \r\n\r\n**Response (200):** Same as POST. **Response (404):** QR not found.\r\n\r\n---\r\n\r\n##### POST `/payment/admin/rental-plans`

Create a rental plan.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

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

Response:

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

GET `/payment/admin/rental-plans/:fleetId`

Get rental plans for a fleet.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Query Parameters:

- `activeOnly` (boolean, default: true)

POST `/payment/admin/penalty`

Create a penalty for a driver.

Authentication: Required
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

POST `/payment/admin/penalty/:id/waive`

Waive a penalty.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Request Body:

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

POST `/payment/admin/incentive`

Create an incentive for a driver.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Request Body:

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

POST `/payment/admin/incentive/:id/payout`

Process incentive payout.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`

Response:

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

POST `/payment/admin/collection/:id/reconcile`

Reconcile daily collection.

Authentication: Required
Roles: `SUPER_ADMIN`, `OPERATIONS`, `MANAGER`

Request Body:

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

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

> ***DEPRECATED***: *Use Bulk Payout instead.*

Process daily payout (Legacy).

Authentication: Required **Roles:** `SUPER_ADMIN` , `OPERATIONS`

POST `/payment/admin/bulk-payout`

Upload CSV for automated bulk payouts.

Authentication: Required **Roles:** `SUPER_ADMIN` , `OPERATIONS`

Request: `multipart/form-data`

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

Response:

```json
{
  "total": 10,
  "success": 9,
  "failed": 1,
  "amountDisbursed": 45000,
  "details": [
    { "phone": "9876543210", "status": "SUCCESS", "amount": 5000, "utr": "UTR..." }
  ]
}
```

GET `/payment/admin/reconciliations/pending`

Get pending reconciliations.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

GET `/payment/admin/payouts/pending`

Get pending payouts.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

POST `/payment/admin/vehicle/:id/qr`

Generate virtual QR code for a vehicle.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS`

Response:

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

GET `/payment/admin/vehicle/:id/qr`

Get vehicle QR code.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

InstaCollect Orders (Dynamic QR)

These endpoints manage ad-hoc payment orders where a customer can pay via a dynamic QR code, supporting partial payments.

POST `/payment/orders`

Create a new payment order. This generates a unique virtual account and QR code.

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

Request Body:

```json
{
  "customerName": "John Doe",
  "customerPhone": "9876543210",
  "customerEmail": "john@example.com",
  "description": "Invoice #1234",
  "amount": 5000
}
```

Response:

```json
{
  "id": "uuid",
  "totalAmount": 5000,
  "collectedAmount": 0,
  "remainingAmount": 5000,
  "status": "PENDING",
  "virtualAccountId": "VA123456",
  "virtualAccountNumber": "1234567890",
  "ifscCode": "HDFC0000001",
  "qrCodeBase64": "data:image/png;base64,...",
  "upiId": "driversklub.va123456@easebuzz",
  "paymentLink": "https://testpay.easebuzz.in/pay/..."
}
```

GET `/payment/orders`

List payment orders.

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

Query Parameters:

- `status` (string, optional): `PENDING`, `PARTIAL`, `COMPLETED`, `FAILED`
- `search` (string, optional): Search by customer name or phone

Response:

```
{
  "orders": [
    {
      "id": "uuid",
      "customerName": "John Doe",
      "totalAmount": 5000,
      "collectedAmount": 2000,
      "remainingAmount": 3000,
      "status": "PARTIAL",
      "createdAt": "2025-12-30T10:00:00Z"
    }
  ]
}
```

GET `/payment/orders/:id`

Get detailed order information including transaction history.

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

Response:

```
{
  "id": "uuid",
  "customerName": "John Doe",
  "totalAmount": 5000,
  "collectedAmount": 2000,
  "remainingAmount": 3000,
  "status": "PARTIAL",
  "transactions": [
    {
      "id": "tx_1",
      "amount": 2000,
      "status": "SUCCESS",
      "createdAt": "2025-12-30T10:05:00Z",
      "easebuzzTxnId": "TXN123"
    }
  ]
}
```

Partner Integrations

MakeMyTrip (MMT)

Full integration documentation is available in the dedicated guide: **MMT Integration Guide**.

Summary of MMT Endpoints

Endpoint	Method	Description
`/partners/mmt/partnersearchendpoint`	POST	Search availability
`/partners/mmt/partnerblockendpoint`	POST	Block inventory
`/partners/mmt/partnerpaidendpoint`	POST	Confirm booking
`/partners/mmt/partnercancelendpoint`	POST	Cancel booking
`/partners/mmt/partnerrescheduleblockendpoint`	POST	Reschedule check
`/partners/mmt/partnerrescheduleconfirmendpoint`	POST	Reschedule confirm
`/partners/mmt/booking/details`	GET	Check booking status

Rapido (Fleet)

Webhook callbacks from Rapido Fleet API.

POST `/rapido/webhook/order-status`

Callback for Rapido order status updates.

Authentication: None (Public/Webhook)

Request Body:

```
{
  "orderId": "rapido_order_123",
  "status": "started",
  "captainId": "cap_123",
  "timestamp": "2025-12-30T10:00:00Z"
}
```

Response:

```
{
  "success": true,
  "message": "Order status updated"
}
```

POST `/rapido/webhook/captain-status`

Callback for Captain login/availability status.

Authentication: None (Public/Webhook)

Request Body:

```
{
  "captainId": "cap_123",
  "status": "online",
  "location": { "lat": 28.5, "lng": 77.1 }
}
```

Response:

```
{
  "success": true,
  "message": "Captain status updated"
}
```

Pricing Engine

Pricing endpoints for fare calculation.

POST `/pricing/preview`

Calculate fare estimate for a trip.

Authentication: Required
Roles: `SUPER_ADMIN` , `OPERATIONS` , `MANAGER` , `DRIVER`

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 (use one of these):
  "vehicleType": "EV",              // Option 1: Direct type
  "vehicleSku": "TATA_TIGOR_EV",    // Option 2: Vehicle SKU (auto-detects type)

  "distanceKm": 25.5  // Optional: Used as fallback if Google Maps unavailable
}
```

Response:

```
{
  "success": true,
  "data": {
    "distanceSource": "GOOGLE_MAPS",
    "billableDistanceKm": 26,
    "ratePerKm": 25,
    "baseFare": 650,
    "totalFare": 780,
    "breakdown": {
      "distanceFare": 650,
      "tripTypeMultiplier": 1.2,
      "bookingTimeMultiplier": 1.0,
      "vehicleMultiplier": 1.0
    },
    "currency": "INR"
  },
  "message": "Fare calculated successfully"
}
```

Distance Calculation:

- If `pickup` and `drop` are provided AND `GOOGLE_MAPS_API_KEY` is set, the system uses Google Maps Distance Matrix API
- Falls back to `distanceKm` if Google Maps fails or is not configured
- Response includes `distanceSource` field indicating which source was used

See Also: Pricing Engine Documentation for complete details.

Complete Endpoint Reference - All Services

This section provides a comprehensive reference for all 112+ endpoints across all 6 microservices.

Summary Table

Service	Endpoint Count
Authentication	8
Driver	22
Vehicle	23
Assignment	4
Trip	54
Notification	1
TOTAL	**112**

Authentication Service (8 endpoints)

Method	Endpoint	Auth	Roles	Description
POST	`/auth/send-otp`	No	-	Send OTP to phone number
POST	`/auth/verify-otp`	No	-	Verify OTP and get tokens
POST	`/auth/refresh`	No	-	Refresh access token
POST	`/auth/logout`	No	-	Logout user
POST	`/users`	Yes	SUPER_ADMIN	Create new user
GET	`/users`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get all users
GET	`/users/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get user by ID
PATCH	`/users/:id/deactivate`	Yes	SUPER_ADMIN	Deactivate user

Driver Service (22 endpoints)

Method	Endpoint	Auth	Roles	Description
POST	`/drivers`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Create driver
GET	`/drivers/fleet/:fleetId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get drivers by fleet
GET	`/drivers/hub/:hubId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get drivers by hub
GET	`/drivers/me`	Yes	DRIVER	Get my profile
GET	`/drivers/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get driver by ID
PATCH	`/drivers/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update driver
PATCH	`/drivers/:id/status`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update driver status
PATCH	`/drivers/:id/availability`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update availability
GET	`/drivers/:id/preference`	Yes	SUPER_ADMIN, DRIVER, MANAGER	Get driver preferences

POST	`/drivers/:id/preference/update`	Yes	SUPER_ADMIN, DRIVER, MANAGER	Update preference request
GET	`/drivers/preference/pending-requests`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get pending requests
POST	`/drivers/preference/update-status`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update request status
GET	`/drivers/:id/active-plan`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER, DRIVER	Get active rental plan
GET	`/drivers/:id/plan-history`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER, DRIVER	Get plan history
POST	`/attendance/check-in`	Yes	All	Check in attendance
POST	`/attendance/check-out`	Yes	All	Check out attendance
POST	`/attendance/start-break`	Yes	All	Start break
POST	`/attendance/end-break`	Yes	All	End break
GET	`/attendance/history`	Yes	All	Get attendance history
GET	`/attendance/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get attendance by ID
POST	`/attendance/:id/approve`	Yes	SUPER_ADMIN, MANAGER	Approve attendance
POST	`/attendance/:id/reject`	Yes	SUPER_ADMIN, MANAGER	Reject attendance

Vehicle Service (23 endpoints)

Method	Endpoint	Auth	Roles	Description
POST	`/vehicles`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Create vehicle
GET	`/vehicles/fleet/:fleetId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get vehicles by fleet
GET	`/vehicles/hub/:hubId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get vehicles by hub
GET	`/vehicles/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get vehicle by ID
PATCH	`/vehicles/:id`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update vehicle

PATCH	`/vehicles/:id/docs`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update vehicle documents
PATCH	`/vehicles/:id/status`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Update vehicle status
PATCH	`/vehicles/:id/deactivate`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Deactivate vehicle
POST	`/fleets`	Yes	SUPER_ADMIN	Create fleet
GET	`/fleets`	Yes	SUPER_ADMIN, OPERATIONS	Get all fleets
GET	`/fleets/:id`	Yes	SUPER_ADMIN, OPERATIONS	Get fleet by ID
PATCH	`/fleets/:id/deactivate`	Yes	SUPER_ADMIN	Deactivate fleet
POST	`/fleets/:id/hubs`	Yes	SUPER_ADMIN, OPERATIONS	Create fleet hub
GET	`/fleets/:id/hubs`	Yes	SUPER_ADMIN, OPERATIONS	Get all fleet hubs
GET	`/fleets/hubs/:id`	Yes	SUPER_ADMIN, OPERATIONS	Get fleet hub by ID
POST	`/fleets/:id/hub-managers`	Yes	SUPER_ADMIN, OPERATIONS	Create hub manager
GET	`/fleets/:id/hub-managers`	Yes	SUPER_ADMIN, OPERATIONS	Get all hub managers
GET	`/fleets/hub-manager/:id`	Yes	SUPER_ADMIN, OPERATIONS	Get hub manager by ID
POST	`/fleets/hubs/:hubId/assign-manager`	Yes	SUPER_ADMIN, OPERATIONS	Assign manager to hub
POST	`/fleets/hubs/:id/add-vehicle`	Yes	SUPER_ADMIN, OPERATIONS	Add vehicle to hub
POST	`/fleets/hubs/:id/add-driver`	Yes	SUPER_ADMIN, OPERATIONS	Add driver to hub
POST	`/fleets/hubs/:id/remove-vehicle`	Yes	SUPER_ADMIN, OPERATIONS	Remove vehicle from hub
POST	`/fleets/hubs/:id/remove-driver`	Yes	SUPER_ADMIN, OPERATIONS	Remove driver from hub

Assignment Service (4 endpoints)

Method	Endpoint	Auth	Roles	Description
POST	`/assignments`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Create assignment
GET	`/assignments/fleet/:fleetId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get assignments by fleet
GET	`/assignments/trip/:tripId`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	Get assignments by trip

PATCH	`/assignments/:id/end`	Yes	SUPER_ADMIN, OPERATIONS, MANAGER	End assignment

---

Notification Service (1 endpoint)

Method	Endpoint	Auth	Roles	Description
POST	`/notifications/send`	Yes	All	Send notification

Trip Service endpoints (54 total) are documented in previous sections including:

- Trip Operations (11 endpoints)
- Admin Trip Operations (4 endpoints)
- Pricing (1 endpoint)
- Payment Driver Endpoints (8 endpoints)
- Payment Admin Endpoints (13 endpoints)
- InstaCollect Orders (3 endpoints)
- MMT Partner Integration (7 endpoints)
- Rapido Partner Integration (5 endpoints)
- Webhooks (2 endpoints)

Maps Service (2 endpoints)

Method	Endpoint	Auth	Roles	Description
GET	`/maps/autocomplete`	Yes	DRIVER, OPS, MANAGER, SUPER_ADMIN	Get location suggestions
GET	`/maps/geocode`	Yes	DRIVER, OPS, MANAGER, SUPER_ADMIN	Geocode address to lat/lng

---

Last Updated: January 15, 2026
Total Documented Endpoints: 114 across 6 microservices

Google Maps Service (2 endpoints)

Method	Endpoint	Auth	Roles	Description
GET	`/maps/autocomplete`	Yes	DRIVER, OPS, MANAGER, SUPER_ADMIN	Get location suggestions
GET	`/maps/geocode`	Yes	DRIVER, OPS, MANAGER, SUPER_ADMIN	Geocode address to lat/lng

Autocomplete

GET `/maps/autocomplete`

Get location suggestions for a specific query string. Validates against Google Maps Places API.

Authentication: Required

Query Parameters:

- `query` : String (Required) - The search text (e.g. "Airport")

Response:

```json
{
  "success": true,
  "data": [
    {
      "description": "Indira Gandhi International Airport, New Delhi, Delhi 110037, India",
      "place_id": "ChIJ..."
    }
  ]
}
```

Geocoding

GET `/maps/geocode`

Convert an address string into latitude and longitude coordinates.

Authentication: Required

Query Parameters:

- `address` : String (Required) - The address to geocode

Response:

```json
{
  "success": true,
  "data": {
    "lat": 28.5561624,
    "lng": 77.1002807,
    "formattedAddress": "Indira Gandhi International Airport, New Delhi, Delhi 110037, India"
  }
}
```