📱 Flutter Driver App - API Integration Guide

Target Audience: Mobile Engineering 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 Header: `Authorization: Bearer <ACCESS_TOKEN>`
Version: 4.5.0 (MMT Integration Complete)
Last Updated: January 23, 2026
Last Verified: January 23, 2026

What's New in v4.5.0

- **MMT Integration Complete** - Full tracking sync for all trip events
  - Automatic sync: start, arrived, boarded, alight, not-boarded
  - Location updates every 30 seconds via `POST /trips/:id/location`
  - Proper `booking_id` storage from MMT paid endpoint
- **Referral System** - Drivers get unique referral codes to invite other drivers
- **Enhanced Onboarding** - Full KYC submission via `/drivers/new-driver-onboard`
- **Bank Account Details** - Required for payout processing
- **Vehicle Documents** - Chassis, VIN, insurance dates

> ***Note:*** *All requests go through the API Gateway. The gateway routes to 6 microservices (Auth, Driver, Vehicle, Assignment, Trip, Notification).*

📑 Table of Contents

---

1. Authentication

1.1 New Driver Registration (Onboarding)

The new onboarding flow allows drivers to self-register via the mobile app. It replaces the old "Admin-only" creation model.

Step 1: Check Eligibility & Send OTP

Endpoint: `POST /users/drivers/verify`

Use this to screen the phone number.

- If user exists -> Returns error (User already registered).
- If new -> Sends OTP via SMS.

```
// Request
{ "phone": "9876543210" }

// Response
{ "message": "OTP sent successfully" }
```

Step 2: Verify OTP

Endpoint: `POST /users/drivers/verifyOtp`

Verify the OTP entered by the driver.

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

// Response
{ "message": "OTP verified successfully" }
```

Step 3: Create Account (Signup)

Endpoint: `POST /users/drivers/signup`

Creates the `User` and empty `Driver` profile. Does NOT require Auth token (Public).

```
// Request
{
    "name": "Amit Kumar",
    "phone": "9876543210"
}

// Response
{
    "id": "u-123",
    "name": "Amit Kumar",
    "role": "DRIVER",
    "token": "..." // Auto-login after signup
}
```

> ***Note:*** *If the signup endpoint does not return a token, you must call normal Login (* `POST /auth/verify-otp` *) immediately after signup to get the session token.*

Step 4: KYC Profile Completion (New Onboarding Flow)

Once signup is complete, use the dedicated onboarding endpoint to submit full KYC:

Endpoint: `POST /drivers/new-driver-onboard` **Auth Required:** No (Public endpoint, uses userId from signup)

Request Body:

```
{
  "userId": "uuid-from-signup-response",
  "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",
    "firstName": "Raj",
    "lastName": "Kumar",
    "kycStatus": "PENDING"
  },
  "message": "Documents Uploaded, KYC pending for verification"
}
```

Alternative: Update Existing Profile

Use `PATCH /drivers/:id` to update individual fields after onboarding.

Personal Information:

- `firstName`, `lastName`, `mobile`
- `email`
- `dob` (Date of Birth)
- `address`, `city`, `pincode`
- `profilePic`

KYC Values (Document Numbers):

- `aadharNumber` - 12-digit Aadhaar number
- `panNumber` - 10-character PAN
- `dlNumber` - Driving License number
- `licenseNumber` - Alternative DL field
- `gstNumber` - GST number (optional)

Bank Account Details (for payouts):

- `bankAccountNumber` - Bank account number
- `bankIfscCode` - IFSC code
- `bankAccountName` - Account holder name

Document Uploads (via S3 Presigned URLs):

- `licenseFront`, `licenseBack` - Driving License images
- `aadharFront`, `aadharBack` - Aadhaar card images
- `panCardImage` - PAN card image
- `livePhoto` - Live selfie for verification
- `bankIdProof` - Bank passbook/statement image

Vehicle Documents (if driver has assigned vehicle):

- `rcFrontImage`, `rcBackImage` - RC images
- `fitnessImage`, `fitnessExpiry` - Fitness certificate
- `insuranceImage`, `insuranceStart`, `insuranceExpiry` - Insurance documents
- `chassisNumber` - Vehicle chassis number
- `vinNumber` - Vehicle Identification Number

1.2 Send OTP

Endpoint: `POST /auth/send-otp`
Auth Required: No

Request Body:

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

Response (200):

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

1.2 Verify OTP

Endpoint: `POST /auth/verify-otp`
Auth Required: No

Request Body:

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

Request Headers (Optional):

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

Note: Set `x-client-type` header to `app` for mobile apps or `web` for web clients. This determines refresh token expiry duration.

Response (200):

```
{
  "success": true,
  "statusCode": 200,
  "data": {
    "accessToken": "eyJhbGciOiJIUzI1NiIs...",
    "refreshToken": "8f8e23...",
    "user": {
      "id": "uuid-user-id",
      "phone": "9876543210",
      "role": "DRIVER"
    }
  }
}
```

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

Action:

1. Check if `user.role === 'DRIVER'` . If not, show "Unauthorized App" error.
2. Store `accessToken` securely (Keychain/Keystore).
3. Store `refreshToken` for silent token renewal.

1.3 Refresh Token

Endpoint: `POST /auth/refresh`
Auth Required: No

Request Body:

```
{
  "refreshToken": "8f8e23..."
}
```

Response (200):

```
{
  "success": true,
  "data": {
    "accessToken": "eyJ..."
  }
}
```

Implementation: Call this automatically when you receive `401 Unauthorized` on any protected endpoint.

---

2. Daily Attendance

2.1 Check In (Start Shift)

Endpoint: `POST /attendance/check-in`
Auth Required: Yes
Role: DRIVER

Request Body:

```
{
  "driverId": "uuid-driver-id-from-profile",
  "lat": 28.4595,
  "lng": 77.0266,
  "odometer": 10500,
  "selfieUrl": "https://s3.aws.com/bucket/selfie.jpg"
}
```

Important: Upload selfie to S3/Cloudinary first, then send the URL.

Response (201):

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "status": "PENDING_APPROVAL",
    "checkInTime": "2025-12-25T08:00:00Z"
  }
}
```

Side Effects:

- **Rapido Status**: Driver is automatically marked **ONLINE** on Rapido (if no other conflicts exist).

2.2 Check Out (End Shift)

Endpoint: `POST /attendance/check-out`
Auth Required: Yes
Role: DRIVER

Request Body:

```
{
  "driverId": "uuid-driver-id",
  "odometer": 10650,
  "odometerImageUrl": "https://s3.aws.com/bucket/odometer.jpg",
  "cashDeposited": 5000
}
```

Note:

- `cashDeposited` is the Amount the driver declares they are submitting (Cash + UPI collection) at day end.
- `odometerImageUrl` is optional. Upload odometer image to S3 first using `/drivers/upload-url` , then send the URL.

Response (200):

```
{
  "success": true,
  "message": "Check-out successful",
  "data": {
    "checkOutTime": "2025-12-25T18:00:00Z",
    "totalKm": 150
  }
}
```

Error Responses (400):

- **Invalid Odometer:** `message: "Odometer reading cannot be less than start reading (10500)"`
- **Invalid Cash:** `message: "Invalid cash deposit amount"`

Side Effects:

- **Rapido Status**: Driver is forced **OFFLINE** on Rapido immediately.

2.3 Start Break (Start Break in Shift)

Endpoint: `POST /attendance/start-break`
Auth Required: Yes
Role: DRIVER

Request Body:

```
{
  "driverId": "uuid-driver-id"
}
```

Response (200):

```
{
    "success": true,
    "statusCode": 200,
    "message": "Break started successfully",
    "data": {
        "id": "1d0c84d0-a593-4079-8ed3-2ce274ad378d",
        "driverId": "b16dd8ec-a030-44a9-9175-ebd77013dbd0",
        "checkInTime": "2025-12-25T10:06:20.674Z",
        "checkOutTime": null,
        "status": "APPROVED",
        "approvedBy": null,
        "adminRemarks": null,
        "checkInLat": 19.076,
        "checkInLng": 72.8777,
        "selfieUrl": "https://cdn.example.com/selfies/driver_98765.jpg",
        "odometerStart": 45230,
        "odometerEnd": null,
        "breakStartTime": "2025-12-25T10:07:31.911Z",
        "breakEndTime": null,
        "createdAt": "2025-12-25T10:06:20.674Z",
        "updatedAt": "2025-12-25T10:07:31.920Z"
    }
}
```

2.4 End Break (End Break in shift)

Endpoint: `POST /attendance/end-break`
Auth Required: Yes
Role: DRIVER

Request Body:

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

Response (200):

```
{
    "success": true,
    "statusCode": 200,
    "message": "Break ended successfully",
    "data": {
        "id": "1d0c84d0-a593-4079-8ed3-2ce274ad378d",
        "driverId": "b16dd8ec-a030-44a9-9175-ebd77013dbd0",
        "checkInTime": "2025-12-25T10:06:20.674Z",
        "checkOutTime": null,
        "status": "APPROVED",
        "approvedBy": null,
        "adminRemarks": null,
        "checkInLat": 19.076,
        "checkInLng": 72.8777,
        "selfieUrl": "https://cdn.example.com/selfies/driver_98765.jpg",
        "odometerStart": 45230,
        "odometerEnd": null,
        "breakStartTime": "2025-12-25T10:07:31.911Z",
        "breakEndTime": "2025-12-25T10:08:14.312Z",
        "createdAt": "2025-12-25T10:06:20.674Z",
        "updatedAt": "2025-12-25T10:08:14.316Z"
    }
}
```

2.5 Get Attendance History

Endpoint: `GET /attendance/history?driverId={uuid}`
Auth Required: Yes

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "checkInTime": "2025-12-25T08:00:00Z",
      "checkOutTime": "2025-12-25T18:00:00Z",
      "status": "APPROVED",
      "odometerStart": 10500,
      "odometerEnd": 10650
    }
  ]
}
```

---

3. Trip Management

3.1 Get My Assigned Trips

Endpoint: `GET /trips?status=DRIVER_ASSIGNED`
Auth Required: Yes
Role: DRIVER

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "uuid-trip-id",
      "tripType": "AIRPORT",
      "originCity": "Delhi",
      "pickupLocation": "T3 Terminal, Gate 4",
      "pickupLat": 28.5562,
      "pickupLng": 77.1000,
      "dropLocation": "Cyber Hub, Gurgaon",
      "pickupTime": "2025-12-25T10:00:00Z",
      "status": "DRIVER_ASSIGNED",
      "price": 1200,
      "distanceKm": 45
    }
  ]
}
```

3.2 Get Trip Details

Endpoint: `GET /trips/:id`
Auth Required: Yes

Response (200):

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "tripType": "AIRPORT",
    "pickupLocation": "T3 Terminal, Gate 4",
    "pickupLat": 28.5562,
    "pickupLng": 77.1000,
    "dropLocation": "Cyber Hub",
    "pickupTime": "2025-12-25T10:00:00Z",
    "status": "DRIVER_ASSIGNED",
    "price": 1200,
    "customerPhone": "9876543210",
    "customerName": "John Doe",
    "provider": "MMT"
  }
}
```

3.3 🚦 Trip Lifecycle State Machine

Perform these actions **strictly in order**. Send GPS coordinates with every status change.

> *[!IMPORTANT] **Trip Types:***
>
> - ***Internal Trips:** Created by admin, standard flow*
> - ***External Trips (MMT):** From MakeMyTrip, includes `provider: "MMT"`, `providerBookingId`, and `providerMeta.otp`*

Step A: Start Trip (En-route to Pickup)

Endpoint: `POST /trips/:id/start`
Auth Required: Yes

Request Body:

```
{
  "lat": 28.5500,
  "lng": 77.0900
}
```

⚠️ **STRICT CONSTRAINT:**

- Can ONLY start within **2.5 hours** of `pickupTime`
- Error if too early: `400 "Cannot start trip more than 2.5 hours before pickup"`

Response (200) - Internal Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Trip started successfully",
  "data": {
    "id": "uuid-trip-id",
    "tripType": "AIRPORT",
    "originCity": "Delhi",
    "destinationCity": "Noida",
    "pickupLocation": "Terminal 1, IGI Airport",
    "pickupLat": 28.5550838,
    "pickupLng": 77.0844015,
    "dropLocation": "Noida, Sector 62",
    "pickupTime": "2026-01-24T16:00:00.000Z",
    "distanceKm": 40,
    "price": 950,
    "status": "STARTED",
    "startedAt": "2026-01-24T05:44:45.105Z",
    "provider": null,
    "providerBookingId": null
  }
}
```

Response (200) - MMT Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Trip started successfully",
  "data": {
    "id": "uuid-trip-id",
    "tripType": "AIRPORT",
    "originCity": "Delhi",
    "destinationCity": "Noida",
    "pickupLocation": "Terminal 1, IGI Airport",
    "pickupLat": 28.5550838,
    "pickupLng": 77.0844015,
    "dropLocation": "Noida, Sector 62",
    "pickupTime": "2026-01-24T16:00:00.000Z",
    "distanceKm": 40,
    "price": 950,
    "status": "STARTED",
    "startedAt": "2026-01-24T05:44:45.105Z",
    "provider": "MMT",
    "providerBookingId": "BKS88888800933",
    "providerMeta": {
      "otp": "4963"
    }
  }
}
```

Side Effects:

- Status: `DRIVER_ASSIGNED` → `STARTED`
- **MMT Trip:** Webhook `POST /track/{providerBookingId}/start` sent to MMT with driver location

Step B: Arrived (At Pickup Location)

Endpoint: `POST /trips/:id/arrived`
Auth Required: Yes

Request Body:

```
{
  "lat": 28.5562,
  "lng": 77.1000
}
```

⚠️ **STRICT CONSTRAINTS:**

1. **Geofence:** Must be within **500m** of `pickupLat` / `pickupLng`
2. **Time:** Must be within **30 minutes** of `pickupTime`

Errors:

- `400 "Driver not within 500m geofence"` - Too far from pickup
- `400 "Cannot arrive more than 30 minutes before pickup"` - Too early

Response (200) - Internal Trip:

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

Response (200) - MMT Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Driver marked as arrived",
  "data": {
    "id": "uuid-trip-id",
    "status": "ARRIVED_EVENT_SENT",
    "arrivedAt": "2026-01-24T05:53:24.389Z",
    "pickupLocation": "Terminal 1, IGI Airport",
    "pickupLat": 28.5550838,
    "pickupLng": 77.0844015,
    "provider": "MMT",
    "providerBookingId": "BKS88888800933",
    "providerMeta": {
      "otp": "4963"
    }
  }
}
```

Side Effects:

- Status: `STARTED` → `ARRIVED_EVENT_SENT`
- SMS sent to customer: "Driver Arrived"
- **MMT Trip:** Webhook `POST /track/{providerBookingId}/arrived` sent to MMT

Step C: Passenger Onboard (Ride Begins)

Endpoint: `POST /trips/:id/onboard`
Auth Required: Yes

Request Body:

```
{
  "otp": "4963",
  "lat": 28.5550838,
  "lng": 77.0844015
}
```

Field Details:

- `otp` (optional): For MMT trips, this OTP is required for passenger verification. Get from `providerMeta.otp`
- `lat`, `lng` (optional): Current location at boarding

Response (200) - Internal Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Passenger boarded",
  "data": {
    "id": "uuid-trip-id",
    "status": "STARTED",
    "boardedAt": "2026-01-24T05:54:15.234Z"
  }
}
```

Response (200) - MMT Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Passenger boarded",
```

```json
  "data": {
    "id": "uuid-trip-id",
    "status": "STARTED",
    "boardedAt": "2026-01-24T05:54:15.234Z",
    "provider": "MMT",
    "providerBookingId": "BKS88888800933"
  }
}
```

Side Effects:

- Status: `ARRIVED_EVENT_SENT` → `STARTED` (boarded state)
- **MMT Trip:** Webhook `POST /track/{providerBookingId}/boarded` sent to MMT with boarding confirmation

---

Step D: Complete (Dropoff)

Endpoint: `POST /trips/:id/complete`
Auth Required: Yes

Request Body - Internal Trip (Basic):

```json
{
  "lat": 28.5355,
  "lng": 77.3910,
  "fare": 1200
}
```

Request Body - MMT Trip (With Extra Charges):

```json
{
  "lat": 28.5355,
  "lng": 77.3910,
  "fare": 1200,
  "extraCharges": {
    "toll": 150,
    "parking": 50
  }
}
```

Field Details:

- `lat` , `lng` : Drop-off location coordinates
- `fare` (optional): Final fare amount
- `extraCharges` (optional): Additional charges breakdown
  - `toll` (number): Toll charges in rupees
  - `parking` (number): Parking charges in rupees
  - `extraKms` (number): Extra kilometer charges
  - `extraMinutes` (number): Extra time charges

Response (200) - Internal Trip:

```json
{
  "success": true,
  "statusCode": 200,
  "message": "Trip completed successfully",
  "data": {
    "id": "uuid-trip-id",
    "tripType": "AIRPORT",
    "originCity": "Delhi",
    "destinationCity": "Noida",
    "pickupLocation": "Terminal 1, IGI Airport",
    "dropLocation": "Noida, Uttar Pradesh, India",
    "pickupTime": "2026-01-24T16:00:00.000Z",
    "distanceKm": 40,
    "price": 950,
    "status": "COMPLETED",
    "completedAt": "2026-01-24T05:54:32.102Z",
    "startedAt": "2026-01-24T05:44:45.105Z",
    "dropLat": 28.5355161,
    "dropLng": 77.3910265
  }
}
```

Response (200) - MMT Trip (With Extra Charges):

```json
{
  "success": true,
  "statusCode": 200,
  "message": "Trip completed successfully",
  "data": {
    "id": "uuid-trip-id",
    "tripType": "AIRPORT",
    "originCity": "Delhi",
    "destinationCity": "Noida",
    "pickupLocation": "Terminal 1, IGI Airport",
    "dropLocation": "Noida, Uttar Pradesh, India",
    "pickupTime": "2026-01-24T16:00:00.000Z",
    "distanceKm": 40,
    "price": 950,
    "status": "COMPLETED",
    "completedAt": "2026-01-24T05:54:32.102Z",
    "startedAt": "2026-01-24T05:44:45.105Z",
    "dropLat": 28.5355161,
    "dropLng": 77.3910265,
    "provider": "MMT",
    "providerBookingId": "BKS88888800933",
    "providerMeta": {
      "otp": "4963"
    }
  }
}
```

Side Effects:

- Status: `STARTED` → `COMPLETED`
- Driver becomes available for next assignment
- **MMT Trip:** Webhook `POST /track/{providerBookingId}/alight` sent to MMT with:
    - Final coordinates
    - Extra charges (if provided): `extra_charge: 200` (toll + parking)
    - Detailed breakdown in `extra_fare_breakup` object

Alternative: No Show

Endpoint: `POST /trips/:id/noshow`
Auth Required: Yes

Request Body:

```
{
  "lat": 28.5550838,
  "lng": 77.0844015,
  "reason": "Customer not reachable"
}
```

Field Details:

- `lat` , `lng` : Current location when marking no-show
- `reason` (optional): Reason for no-show

⚠️ **STRICT CONSTRAINT:**

- Can ONLY mark no-show **AFTER** driver has arrived (`ARRIVED_EVENT_SENT` status)
- Best practice: Wait **15-30 minutes** past `pickupTime` before marking no-show

Response (200) - Internal Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Trip marked as No Show",
  "data": {
    "id": "uuid-trip-id",
    "status": "NO_SHOW",
    "pickupLocation": "Terminal 1, IGI Airport"
  }
}
```

Response (200) - MMT Trip:

```
{
  "success": true,
  "statusCode": 200,
  "message": "Trip marked as No Show",
  "data": {
    "id": "uuid-trip-id",
    "status": "NO_SHOW",
    "pickupLocation": "Terminal 1, IGI Airport",
    "provider": "MMT",
    "providerBookingId": "BKS88888800929"
  }
}
```

Side Effects:

- Status: `ARRIVED_EVENT_SENT` → `NO_SHOW`
- Driver becomes available for next assignment
- **MMT Trip:** Webhook `POST /track/{providerBookingId}/not_boarded` sent to MMT

3.4 Update Live Location

Endpoint: `POST /trips/:id/location` **Auth Required:** Yes

Request Body:

```
{
  "lat": 28.5500,
  "lng": 77.0900
}
```

Response (200):

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

Implementation Note:

- Call this endpoint every **30-60 seconds** while trip is in progress (`STARTED` , `ARRIVED_EVENT_SENT`)
- **MMT Trips:** Location updates are automatically synced to MMT via `PUT /track/{providerBookingId}/location` webhook
- Provides real-time tracking for customers and admin

3.5 Get Live Tracking

Endpoint: `GET /trips/:id/tracking`
Auth Required: Yes

Response (200):

```
{
  "success": true,
  "data": {
    "currentLat": 28.5500,
    "currentLng": 77.0900,
    "lastUpdated": "2025-12-25T09:45:00Z"
  }
}
```

3.6 Partner Trips (MMT)

Identification: Trips assigned from MakeMyTrip can be identified by:

- `provider`: `"MMT"` (in Trip Details)
- `providerBookingId`: `"BKS88888800926"` (MMT's booking ID)
- `tripType`: `"AIRPORT"` or `"OUTSTATION"`

Special Handling Rules:

1. **Prepaid/Zero Payment**:
    - MMT trips are prepaid.
    - **Do NOT collect cash** from the customer even if `price` is shown.
    - Show "PREPAID" tag in the UI.
2. **Mandatory OTP**:
    - MMT requires a valid OTP for onboarding.
    - Ensure the driver enters the exact 4-digit OTP provided by the customer.
    - OTP is stored in `providerMeta.otp` field.
    - Sending "0000" or invalid OTP may cause MMT to reject the 'Onboard' status.
3. **Location Updates (CRITICAL)**:
    - MMT strictly tracks vehicle movement.
    - **Must call `POST /trips/:id/location` every 30 seconds** from Start to Alight.
    - Failure to send location updates may result in penalties from MMT.
4. **Cancellation**:
    - If a driver cancels an MMT trip, it triggers an immediate unassign webhook to MMT.
    - **Avoid frequent cancellations** to maintain fleet rating.

MMT Trip Event Flow:

```
1. Admin assigns driver → MMT receives /dispatch/{booking_id}/assign
2. Driver taps "Start" → MMT receives /track/{booking_id}/start
3. Driver taps "Arrived" → MMT receives /track/{booking_id}/arrived
4. Driver enters OTP, taps "Onboard" → MMT receives /track/{booking_id}/boarded
5. Every 30 seconds during trip → MMT receives /track/{booking_id}/location
6. Driver taps "Complete" → MMT receives /track/{booking_id}/alight
```

No-Show Flow:

```
1. Driver arrives, waits 30+ minutes after pickup time
2. Driver taps "No Show" → MMT receives /track/{booking_id}/not-boarded
```

---

4. Driver Profile

4.1 Get My Profile

Endpoint: `GET /drivers/me`
Auth Required: Yes
Role: DRIVER

Response (200):

```json
{
  "success": true,
  "data": {
    "id": "uuid",
    "firstName": "Raj",
    "lastName": "Kumar",
    "mobile": "9876543210",
    "licenseNumber": "DL-12345-67890",
    "kycStatus": "APPROVED",
    "status": "ACTIVE",
    "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",
          "status": "ACTIVE"
        }
      }
    ]
  }
}
```

Note:

- `assignments` array contains the currently assigned vehicle (if any)

- If no vehicle is assigned, `assignments` will be an empty array `[]`
- Use `assignments[0].vehicle` to access the assigned vehicle details

4.2 Get Driver Profile by ID

Endpoint: `GET /drivers/:id` **Auth Required:** Yes **Role:** `SUPER_ADMIN` , `OPERATIONS` , `MANAGER`

Response (200):

```
{
  "success": true,
  "data": {
    "id": "uuid",
    "firstName": "Raj",
    "lastName": "Kumar",
    // ... other fields
  }
}
```

4.3 Update Driver Profile

Endpoint: `PATCH /drivers/:id` **Auth Required:** Yes **Role:** `SUPER_ADMIN` , `OPERATIONS` , `MANAGER` , `DRIVER`

Request Body (all fields optional):

```
{
  "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",

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

  "bankAccountNumber": "1234567890123456",
  "bankIfscCode": "HDFC0001234",
  "bankAccountName": "Rajesh 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
Basic Info	`firstName` , `lastName` , `mobile` , `email` , `dob` , `address` , `city` , `pincode`
KYC Values	`aadharNumber` , `panNumber` , `dlNumber` , `gstNumber`
Bank Details	`bankAccountNumber` , `bankIfscCode` , `bankAccountName`
KYC Attachments	`licenseFront` , `licenseBack` , `aadharFront` , `aadharBack` , `panCardImage` , `bankIdProof`
Vehicle Docs	`rcFrontImage` , `rcBackImage` , `fitnessImage` , `fitnessExpiry` , `insuranceImage` , `insuranceStart` , `insuranceExpiry` , `chassisNumber` , `vinNumber`

Note: Vehicle document fields are only updated if the driver has an assigned vehicle (`vehicleId` is set).

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "planName": "Weekly Plan",
      "rentalAmount": 3000,
      "depositAmount": 5000,
      "validityDays": 7,
      "startDate": "2026-01-10T00:00:00.000Z",
      "expiryDate": "2026-01-17T00:00:00.000Z",
      "isActive": true,
      "status": "ACTIVE"
    },
    {
      "id": "uuid-2",
      "planName": "Monthly Plan",
      "rentalAmount": 10000,
      "depositAmount": 5000,
      "validityDays": 30,
      "startDate": "2025-12-01T00:00:00.000Z",
      "expiryDate": "2025-12-31T00:00:00.000Z",
      "isActive": false,
      "status": "EXPIRED"
    }
  ],
  "message": "Plan history retrieved successfully"
}
```

Status Values:

- `ACTIVE` - Current active plan (not expired)
- `EXPIRED` - Plan has passed expiry date
- `INACTIVE` - Plan was deactivated before expiry

Use Case:

- Show driver's rental history in "My Plans" section
- Display past plans with their validity periods
- Useful for admin to track driver's subscription history

4.6 Driver Preferences

Get My Preferences

Endpoint: `GET /drivers/me/preference`
Auth Required: Yes
Role: DRIVER

> *[!NOTE] **Rapido Status Sync:** Driver availability (`isAvailable`) is managed automatically by the backend based on Login, Trip Status, and Breaks. Manual toggling may be overridden.*

Response (200):

```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
    }
  ]
}
```

Request Preference Change

Endpoint: `POST /drivers/:id/preference/update`
Auth Required: Yes
Role: DRIVER

Request Body:

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

Response (201):

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

4.7 Get Upload URL (S3 Image Upload)

Endpoint: `GET /drivers/upload-url`
Auth Required: Yes
Role: DRIVER

Query Parameters:

- `folder` (required): Folder name - `selfies` , `odometer` , `documents` , `profiles` , `vehicles`
- `fileType` (required): File extension - `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 Upload URL**: Call this endpoint with desired folder and file type

2. **Upload File**: Send a `PUT` request to the `uploadUrl` with the image file as binary body
3. **Use Final URL**: Send the `url` field to other APIs (e.g., `selfieUrl` in check-in, `odometerImageUrl` in check-out)

Example Upload (Dart/Flutter):

```
// Step 1: Get presigned URL
final response = await http.get(
  Uri.parse('$baseUrl/drivers/upload-url?folder=odometer&fileType=jpg'),
  headers: {'Authorization': 'Bearer $token'},
);
final data = jsonDecode(response.body)['data'];

// Step 2: Upload file to S3
final file = File(imagePath);
await http.put(
  Uri.parse(data['uploadUrl']),
  body: await file.readAsBytes(),
  headers: {'Content-Type': 'image/jpeg'},
);

// Step 3: Use the final URL
final imageUrl = data['url'];
```

Allowed Folders:

- `selfies` - Driver check-in selfies
- `odometer` - Odometer reading photos
- `documents` - License, Aadhaar, etc.
- `profiles` - Profile pictures
- `vehicles` - Vehicle photos

Note: Presigned URLs expire in 5 minutes. Upload must be completed within this time.

---

5. Error Handling

5.1 HTTP Status Codes

Code	Error	Meaning	Action
`400`	`VALIDATION_ERROR`	Invalid request body	Check request format
`400`	`TOO_EARLY_START`	Cannot start > 2.5h before pickup	Wait until allowed time
`400`	`TOO_EARLY_ARRIVE`	Cannot arrive > 30min before pickup	Wait until allowed time
`400`	`GEOFENCE_VIOLATION`	Not within 500m of pickup	Move closer to pickup location
`400`	`TOO_EARLY_NOSHOW`	Cannot mark no-show < 30min after pickup	Wait until allowed time

`401`	`UNAUTHORIZED`	Token Invalid/Expired	Call `/auth/refresh` or re-login
`403`	`FORBIDDEN`	Insufficient permissions	Contact admin
`404`	`NOT_FOUND`	Trip/Resource not found	Refresh trip list
`422`	`UNPROCESSABLE_ENTITY`	Business logic violation	Check trip status
`500`	`INTERNAL_SERVER_ERROR`	Backend crash	Retry after few seconds

5.2 Error Response Format

```
{
  "success": false,
  "statusCode": 400,
  "errorCode": "TOO_EARLY_START",
  "message": "Cannot start trip more than 2.5 hours before pickup",
  "timestamp": "2025-12-25T09:00:00Z"
}
```

📝 Checklist for Production

- [ ] Implement token refresh logic
- [ ] Add offline queue mechanism
- [ ] Implement background location tracking
- [ ] Add geofencing validation before API calls
- [ ] Add time constraint validation before API calls
- [ ] Implement retry logic for failed requests
- [ ] Add comprehensive error handling
- [ ] Implement push notifications (FCM)
- [ ] Add analytics/crash reporting (Firebase)
- [ ] Test all edge cases (offline, poor network, etc.)

6. Finance & Rental Plans

6.1 Get Balance & Financial Status

Endpoint: `GET /payments/balance` **Auth Required:** Yes **Role:** DRIVER

Response (200):

```
{
  "success": true,
  "data": {
    "depositBalance": 5100,
    "paymentModel": "RENTAL",
    "hasActiveRental": true,
    "rental": {
      "planName": "Weekly Starter",
      "amount": 2500,
      "startDate": "2026-01-15T00:00:00Z",
      "expiryDate": "2026-01-22T00:00:00Z",
      "daysRemaining": 30,
      "isExpired": false,
      "vehicle": {
        "number": "BR34 QW 1234",
        "model": "TATA Tigor EV"
      }
    }
  }
}
```

6.2 Rental Management

6.2.1 Get Available Plans

Endpoint: `GET /payments/rental/plans` **Description:** List all plans available for subscription.

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Weekly Starter",
      "rentalAmount": 3000,
      "depositAmount": 5000,
      "validityDays": 7,
      "description": "Best for new drivers"
    }
  ]
}
```

6.2.2 Subscribe to Plan

Endpoint: `POST /payments/rental` **Description:** Initiate payment to subscribe to a plan. Requires checking `deposit` balance first.

Request Body:

```
{
  "rentalPlanId": "uuid-plan-id"
}
```

Response (200):

```
{
  "success": true,
  "data": {
    "transactionId": "uuid",
    "paymentUrl": "https://testpay.easebuzz.in/pay/{accessKey}",
    "accessKey": "0c4d0ab671a967784530587dbca8e2c8...",
    "txnId": "TXN_1768492822722_AU6QJR"
  }
}
```

Usage: Open `paymentUrl` in a WebView or browser to complete payment.

6.2.3 Get Active Plan Details

Endpoint: `GET /drivers/:id/active-plan` **Note:** Use Driver ID from `GET /drivers/me`.

Response (200):

```
{
  "success": true,
  "data": {
    "id": "uuid-rental-id",
    "planName": "Weekly Starter",
    "rentalAmount": 2500,
    "depositAmount": 5000,
    "validityDays": 7,
    "startDate": "2026-01-15T00:00:00Z",
    "expiryDate": "2026-01-22T00:00:00Z",
    "isActive": true,
    "daysRemaining": 5,
    "vehicle": {
      "number": "BR34 QW 1234",
      "model": "TATA Tigor EV"
    }
  }
}
```

6.3 Security Deposit

6.3.1 Initiate Top-up

Endpoint: `POST /payments/deposit` **Description:** Add money to security deposit via PG.

Request Body:

```
{
  "amount": 2000
}
```

Response (200):

```json
{
  "success": true,
  "data": {
    "transactionId": "uuid",
    "paymentUrl": "https://testpay.easebuzz.in/pay/{accessKey}",
    "accessKey": "0c4d0ab671a967784530587dbca8e2c8...",
    "txnId": "TXN_1768492822722_AU6QJR"
  }
}
```

Usage: Open `paymentUrl` in a WebView or browser to complete payment.

6.4 Transactions & Summary

6.4.1 Get Transaction History

Endpoint: `GET /payments/transactions` **Query Params:** `page` , `limit` , `type` (DEPOSIT, RENTAL, PENALTY, INCENTIVE)

Response (200):

```json
{
  "success": true,
  "data": {
    "transactions": [
      {
        "id": "uuid",
        "type": "DEPOSIT",
        "amount": 5000,
        "status": "SUCCESS",
        "createdAt": "2025-12-01T10:00:00Z"
      }
    ]
  }
}
```

6.4.2 Get Incentives

Endpoint: `GET /payments/incentives`

6.4.3 Get Penalties

Endpoint: `GET /payments/penalties`

6.4.4 Get Daily Collections (Payout Model)

Endpoint: `GET /payments/collections`

6.4.5 Get Weekly Earnings Summary

Endpoint: `GET /payments/earnings/weekly`
Auth Required: Yes
Role: DRIVER

Query Parameters:

- `weeks` (optional, default: 5): Number of weeks to fetch (1-12)

Response (200):

```json
{
  "success": true,
  "data": {
    "currentWeek": {
      "type": "current",
      "weekNumber": 3,
      "startDate": "2026-01-13",
      "endDate": "2026-01-19",
      "tripCount": 15,
      "tripEarnings": 4500,
      "incentives": 500,
      "penalties": 200,
      "netEarnings": 4800
    },
    "previousWeeks": [
      {
        "weekNumber": 2,
        "startDate": "2026-01-06",
        "endDate": "2026-01-12",
        "tripCount": 12,
        "tripEarnings": 3800,
        "incentives": 300,
        "penalties": 100,
        "netEarnings": 4000
      }
    ],
    "totalEarnings": 8800
  }
}
```

Use Case:

- **Earnings Dashboard**: Show weekly summary card on driver home screen
- **Historical Trends**: Track performance over multiple weeks

---

7. Rapido Status Management

> *[!IMPORTANT] **Automatic Availability Control**:*
>
> *The backend **automatically manages** the driver's Rapido availability based on their internal schedule.*
>
> 1. ***Busy**: When on an internal trip, break, or upcoming assignment (< 45m).*

> 2. ***Available***: *When idle.*
>
> ***Do NOT implement** a manual "Go Online/Offline" toggle for Rapido status in the driver app. If a driver manually overrides this in the Rapido app, the backend will **force them back** to the correct state immediately.*

---

8. Pricing & Utilities

8.1 Check Fare Estimate

Endpoint: `POST /pricing/preview`
Auth Required: Yes
Role: `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 (use one):
  "vehicleType": "EV",              // Option 1
  "vehicleSku": "TATA_TIGOR_EV",    // Option 2

  "distanceKm": 25.5  // Optional fallback
}
```

Response (200):

```
{
  "success": true,
  "data": {
    "distanceSource": "GOOGLE_MAPS", // or "CLIENT_PROVIDED"
    "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"
}
```

> *[!NOTE] **Distance Calculation:** The mobile app should calculate distance using Google Maps SDK, Mapbox, or similar before calling this endpoint. The backend uses the provided `distanceKm` to calculate fare based on trip type, booking advance, and vehicle type.*
>
> ***See Also:** [Pricing Engine Documentation] for complete fare calculation details.*

---

9. Google Maps Service

Base URL: `/maps`

These endpoints allow the app to proxy Google Maps requests through the backend, securing the API Key.

9.1 Autocomplete

Endpoint: `GET /maps/autocomplete` **Query Parameters:** `query` (required) **Use Case:** Address search bar.

9.2 Geocode

Endpoint: `GET /maps/geocode` **Query Parameters:** `address` (required) **Use Case:** Convert address to Lat/Lng.

---

10. Referral System

Drivers can invite other drivers to join the platform and earn rewards.

10.1 How Referrals Work

1. **Every driver gets a unique referral code** automatically after completing signup
2. New drivers can enter a referral code during registration
3. When the referred driver meets activity criteria, the referrer receives an incentive

10.2 Using Referral Code During Signup

When calling `POST /users/drivers/signup`, include the referral code:

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

Validation Errors:

Error	Meaning

"Invalid referral code"	Code doesn't exist
"Referral code belongs to an inactive account"	Referrer is deactivated
"Referral code has reached the maximum usage limit"	Referrer hit 50 referral limit

10.3 Viewing Your Referral Code

Your referral code is available in your profile response (`GET /drivers/me`):

```
{
  "success": true,
  "data": {
    "id": "driver-uuid",
    "firstName": "Raj",
    // ... other fields
    "user": {
      "referralCode": "RAJ123XYZ"
    }
  }
}
```

10.4 Referral Eligibility Criteria

The referred driver must meet these criteria for you to receive the reward:

Criteria	Requirement
Active Days	At least 30 days with approved attendance
Average Rides	At least 8 completed trips per active day

10.5 Referral Rewards

When eligibility is met:

- An **INCENTIVE** of the configured amount (default: ₹500) is created
- Incentive category: `REFERRAL`
- Visible in your incentives list (`GET /payments/incentives`)
- Can be paid out via bank transfer

10.6 UI Implementation Tips

```
// Share referral code
void shareReferralCode(String code) {
  Share.share(
    'Join DriversKlub using my referral code: $code\n'
    'Download the app and start earning today!'
  );
}
```

```dart
// Display referral section
Widget buildReferralCard(String referralCode) {
  return Card(
    child: Column(
      children: [
        Text('Your Referral Code'),
        Text(referralCode, style: TextStyle(fontSize: 24, fontWeight: FontWeight.bold)),
        ElevatedButton(
          onPressed: () => copyToClipboard(referralCode),
          child: Text('Copy Code'),
        ),
        TextButton(
          onPressed: () => shareReferralCode(referralCode),
          child: Text('Share with Friends'),
        ),
      ],
    ),
  );
}
```