📱 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: 4.0.0 (Microservices)
Last Updated: January 16, 2026
Last Verified: January 16, 2026

> ***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 Send OTP

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

Request Body:

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

Response (200):

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

Dev Mode: When `NODE_ENV !== 'production'`, OTP is printed to server console:

```
==========================================
[DEV OTP] Phone: +919876543210
[DEV OTP] Code : 123456
==========================================
```

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.

Dev Bypass (Development Only):

```
{
  "phone": "9876543210",
  "otp": "000000",
  "verifiedKey": "pass"
}
```

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.

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):

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

Side Effects:

- Status: `DRIVER_ASSIGNED` → `STARTED`
- MMT Webhook triggered (if MMT trip)

Step B: Arrived (At Pickup Location)

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

Request Body:

```json
{
  "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):

```json
{
  "success": true,
  "message": "Arrival confirmed"
}
```

Side Effects:

- Status: `STARTED` → `ARRIVED`
- SMS sent to customer: "Driver Arrived"

Step C: Passenger Onboard (Ride Begins)

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

Request Body:

```
{
  "otp": "1234"
}
```

Note: OTP field is optional. Backend validates if provided.

Response (200):

```
{
  "success": true,
  "message": "Passenger onboarded"
}
```

Side Effects:

- Status: `ARRIVED` → `ONBOARD`

Step D: Complete (Dropoff)

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

Request Body:

```
{
  "distance": 45.5,
  "fare": 1200
}
```

Response (200):

```
{
  "success": true,
  "message": "Trip completed successfully"
}
```

Side Effects:

- Status: `ONBOARD` → `COMPLETED`
- Driver becomes available for next assignment

Alternative: No Show

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

Request Body:

```
{
  "reason": "Customer not reachable"
}
```

⚠️ **STRICT CONSTRAINT:**

- Can ONLY mark no-show **AFTER 30 minutes** past `pickupTime`
- Error if too early: `400 "Cannot mark no-show before 30 minutes past pickup time"`

Response (200):

```
{
  "success": true,
  "message": "Trip marked as no-show"
}
```

Side Effects:

- Status: → `NO_SHOW`

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 the trip is in progress (`STARTED` , `ARRIVED` , `ONBOARD`) to update the driver's live location.

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)
- `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.
    - Sending "0000" or invalid OTP may cause MMT to reject the 'Onboard' status.
3. **Location Updates**:
    - MMT strictly tracks vehicle movement.
    - Ensure `POST /trips/:id/location` is called every **30-60 seconds** without fail.
    - Failure to send location updates may result in penalties from MMT.
4. **Cancellation**:
    - If a driver cancels an MMT trip, it triggers an immediate reassignment webhook.
    - **Avoid frequent cancellations** to maintain fleet rating.

---

4. Driver Profile

4.1 Get My Profile

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

Response (200):

```
{
  "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`

Request Body:

```json
{
  "firstName": "Rajesh",
  "email": "rajesh@example.com"
}
```

Response (200):

```json
{
  "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):

```
{
  "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):

```
{
  "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):

```
{
  "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.