

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 12, 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](#)
2. [Daily Attendance](#)
3. [Trip Management](#)
4. [Driver Profile](#)
5. [Error Handling](#)
6. [Payment & Wallet](#)
7. [Rapido Status Management](#)

1. Authentication

1.1 Send OTP

Endpoint: `POST /auth/send-otp`

Auth Required: No

Request Body:

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

Response (200):

```
{  
  "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"
}
```

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
- **Refresh Token:** 7 days

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.amazonaws.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,  
  "cashDeposited": 5000  
}
```

Note: `cashDeposited` is the Amount the driver declares they are submitting (Cash + UPI collection) at day end.

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):

```
{
  "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:

```
{  
  "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):

```
{  
  "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"  
  }  
}
```

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):**

```json
{
 "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:

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

4.4 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"
}
```

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"
}
```

6. Payment & Wallet

6.1 Get Wallet Balance

Endpoint: GET /payment/balance **Auth Required:** Yes **Role:** DRIVER

Response (200):

```
{
  "success": true,
  "data": {
    "driverId": "uuid",
    "currentBalance": 1500.50,
    "currency": "INR",
    "paymentModel": "RENTAL",
    "depositBalance": 5000,
    "pendingDues": 0
  }
}
```

Use Case:

- **Dashboard Display:** Show `currentBalance` at the top of the home screen.
- **Rental Check:** Warn driver if `currentBalance` is negative (RENTAL model).

6.2 Get Transaction History

Endpoint: GET /payment/transactions **Auth Required:** Yes

Query Params:

- `page` (default: 1)
- `limit` (default: 20)
- `type` (optional: `TRIP_PAYMENT` , `INCENTIVE` , `PENALTY` , `DEPOSIT`)

Response (200):

```
{
  "success": true,
  "data": {
    "transactions": [
      {
        "id": "uuid",
        "amount": 450,
        "type": "TRIP_PAYMENT",
        "status": "SUCCESS",
        "createdAt": "2025-12-25T14:30:00Z",
        "description": "Trip payment for Ride #123"
      }
    ],
    "current_page": 1,
    "total_pages": 5
  }
}
```

Use Case:

- **Earnings Report:** Driver checks "Yesterday's Earnings" by filtering transactions.
- **Transparency:** Verify deductions (Penalties) or additions (Incentives).

6.3 Get Incentives

Endpoint: GET /payment/incentives **Auth Required:** Yes

Response (200):

```
{
  "success": true,
  "data": {
    "incentives": [
      {
        "id": "uuid",
        "amount": 500,
        "reason": "Completed 50 Trips",
        "isPaid": false
      }
    ],
    "summary": {
      "totalEarned": 2500,
      "paid": 2000,
      "pending": 500
    }
  }
}
```

```
    }
}
```

Use Case:

- **Motivation:** Show "Progress to Goal" or "Pending Payouts" to encourage more trips.

7. Implementation Notes

6.1 Background Location Tracking

- The backend expects GPS coordinates during status changes
- Implement background location service to track driver position
- Send location updates during trip lifecycle transitions

6.2 Offline Handling

- The API requires online connectivity
- Queue requests locally (SQLite) when offline
- Sync when connection is restored
- Show clear offline indicator to driver

6.3 UI Feedback

- Always show loading indicators during API calls
- API latency can vary (200ms - 2s)
- Implement retry logic for failed requests (max 3 retries)
- Show clear error messages from API responses

6.4 Token Management

```
// Pseudo-code for token refresh
Future<void> refreshTokenIfNeeded() async {
    if (isTokenExpired(accessToken)) {
        final newToken = await api.refreshToken(refreshToken);
        await secureStorage.write('accessToken', newToken);
    }
}

// Intercept 401 responses
if (response.statusCode == 401) {
    await refreshTokenIfNeeded();
    // Retry original request
}
```

6.5 Geofencing Implementation

```

// Check if driver is within 500m of pickup
double distance = Geolocator.distanceBetween(
    driverLat, driverLng,
    pickupLat, pickupLng
);

if (distance > 500) {
    showError("You must be within 500m of pickup location");
    return;
}

```

6.6 Time Constraint Checks

```

// Check if within 2.5h window for start
DateTime now = DateTime.now();
DateTime pickupTime = DateTime.parse(trip.pickupTime);
Duration diff = pickupTime.difference(now);

if (diff.inHours > 2.5) {
    showError("Cannot start trip more than 2.5 hours before pickup");
    return;
}

```

6.7 State Management

- Use Provider/Riverpod/Bloc for state management
- Cache trip list locally
- Implement pull-to-refresh for trip list
- Auto-refresh every 30 seconds when on trip list screen

6.8 Push Notifications

- Implement FCM for trip assignments
- Handle notification when app is in background/killed
- Deep link to specific trip when notification tapped

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.)
-
-

5. Payment & Wallet

5.1 Get Balance & Rental Status

Endpoint: GET /payment/balance

Auth Required: Yes

Role: DRIVER

Response (200):

```
{
  "depositBalance": 5000,
  "paymentModel": "RENTAL",
  "hasActiveRental": true,
  "rental": {
    "planName": "Weekly Plan",
    "startDate": "2025-12-23T00:00:00.000Z",
    "expiryDate": "2025-12-30T00:00:00.000Z",
    "daysRemaining": 3,
    "isExpired": false
  }
}
```

UI Display:

- Show deposit balance prominently at top of wallet screen
 - Display rental validity like a SIM card validity (days remaining)
 - Show warning when rental expires in < 2 days
-

5.2 Get Available Rental Plans

Endpoint: GET /payment/rental/plans

Auth Required: Yes

Role: DRIVER

Response (200):

```
{
  "success": true,
  "data": [
    {
      "id": "uuid",
      "name": "Weekly Plan",
      "rentalAmount": 3000,
```

```
        "depositAmount": 5000,
        "validityDays": 7,
        "isActive": true
    }
]
}
```

5.3 Get Transaction History

Endpoint: GET /payment/transactions?page=1&limit=20

Auth Required: Yes

Role: DRIVER

Query Parameters:

- `page` (number, default: 1)
- `limit` (number, default: 20)
- `type` (optional): DEPOSIT, RENTAL, INCENTIVE, PENALTY, PAYOUT
- `status` (optional): PENDING, SUCCESS, FAILED

Response (200):

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

UI Implementation:

- Implement infinite scroll or pagination
- Show transaction type with icons (deposit, rental, incentive, penalty)
- Color code: Green (credit), Red (debit)

5.4 Get Incentives

Endpoint: GET /payment/incentives

Auth Required: Yes

Role: DRIVER

Response (200):

```
{  
  "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  
  }  
}
```

UI Display:

- Show total unpaid incentives prominently
- List all incentives with paid/unpaid status
- Show payout date for paid incentives

5.5 Get Penalties

Endpoint: GET /payment/penalties

Auth Required: Yes

Role: DRIVER

Response (200):

```
{  
  "penalties": [  
    {  
      "id": "uuid",  
      "type": "MONETARY",  
      "amount": 200,  
      "reason": "Late for pickup",  
      "isPaid": true,  
      "isWaived": false,  
      "deductedFromDeposit": true,  
      "depositDeductionAmount": 200,  
      "createdAt": "2025-12-25T00:00:00.000Z"  
    }  
  ]  
}
```

```
        "createdAt": "2025-12-24T00:00:00.000Z"
    }
]
}
```

Penalty Types:

- MONETARY - Financial penalty (auto-deducted from deposit)
- WARNING - Verbal/written warning
- SUSPENSION - Temporary suspension
- BLACKLIST - Permanent ban

UI Display:

- Show penalties with clear reason
- Indicate if waived (show in green)
- Show deposit deduction for monetary penalties

5.6 Initiate Deposit Payment

Endpoint: POST /payment/deposit

Auth Required: Yes

Role: DRIVER

Request Body:

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

Response (200):

```
{
  "transactionId": "uuid",
  "paymentUrl": "https://testpay.easebuzz.in/pay/...",
  "txnId": "TXN_1735123456_ABC123"
}
```

Implementation:

- Open `paymentUrl` in WebView or external browser
- Handle success/failure callbacks
- Refresh balance after payment

5.7 Initiate Rental Payment

Endpoint: POST /payment/rental

Auth Required: Yes

Role: DRIVER

Request Body:

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

Response (200):

```
{  
    "transactionId": "uuid",  
    "paymentUrl": "https://testpay.easebuzz.in/pay/...",  
    "txnId": "TXN_1735123456_XYZ789"  
}
```

Implementation:

- Show available rental plans first
- Open payment URL in WebView
- Activate rental after successful payment

5.7 Get Daily Collections (Payout Model Only)

Endpoint: GET /payment/collections

Auth Required: Yes

Role: DRIVER

Query Parameters:

- `startDate` (optional): ISO date
- `endDate` (optional): ISO date

Response (200):

```
{  
    "collections": [  
        {  
            "id": "uuid",  
            "date": "2025-12-29T00:00:00.000Z",  
            "qrCollectionAmount": 3000,  
            "cashCollectionAmount": 2000,  
            "totalCollection": 5000,  
            "qrCollectionAmount": 3000,  
            "cashCollectionAmount": 2000,  
            "totalCollection": 5000,  
            "isPaid": false // Status tracked via Manual Payouts  
        }  
    ],  
    "summary": {  
        "totalDays": 30,  
        "totalCollections": 150000, // Gross Earnings  
    }  
}
```

```
        "totalPayout": 108000,           // Actual Paid (via CSV)
        "deductions": 42000,            // Commission/Platform Fee (Derived)
        "balance": 0                  // No pending dues
    }
}
```

Note: Only visible for drivers on PAYOUT model

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.
