Drive Execution Controller API Documentation

Overview

The Drive Execution Controller manages real-time drive operations during milk delivery. It handles the actual execution phase of drives, including customer interactions, sales recording, QR code scanning, and drive progress tracking. This controller is primarily used by delivery personnel during active delivery operations.

Endpoints

1. GET /drive-execution/:id - Get Drive Execution Details

Purpose: Retrieve comprehensive execution details for an ongoing drive, including customer list, delivery status, and progress metrics.

Path Parameters:

- `id` : Drive ID

Business Rules:

- Only returns data for ongoing drives
- Shows delivery status for each customer in the route
- Calculates real-time progress metrics

Sample Request:

```
GET /api/drive-execution/1
Authorization: Bearer your-token-here
```

Sample Response:

```
{
  "drive": {
    "drive_id": 1,
    "route_id": 1,
    "delivery_guy_id": 1,
    "stock": 100,
    "sold": 45,
    "returned": 0,
    "status": "ongoing",
    "name": "Morning Route A",
    "start_time": "2025-07-18T06:00:00Z",
    "route_name": "Downtown Route",
    "delivery_guy_name": "John Doe"
  },
  "customers": [
    {
      "customer_id": 1,
      "name": "Alice Johnson",
      "phone": "9876543210",
      "location": "Downtown",
      "address": "123 Main St",
      "price": 25.0,
      "default_quantity": 2,
      "delivery_status": "success",
      "delivered_quantity": 2,
      "delivered_amount": 50.0
    },
    {
      "customer_id": 2,
      "name": "Bob Smith",
      "phone": "9876543211",
      "location": "Downtown",
      "address": "456 Oak Ave",
      "price": 25.0,
      "default_quantity": 3,
      "delivery_status": "pending",
      "delivered_quantity": 0,
      "delivered_amount": 0
    }
  ],
  "progress": {
    "total_customers": 15,
    "servedCustomers": 8,
```

```
    "remainingCustomers": 7,
    "completionPercentage": 53.33,
    "quantity_delivered": 45,
    "total_amount": 1125.0,
    "successful_deliveries": 7,
    "failed_deliveries": 1
  }
}
```

2. **POST /drive-execution/:id/record-sale** - Record Customer Sale

Purpose: Record a successful delivery/sale to a customer during drive execution.

Path Parameters:

- `id` : Drive ID

Request Body:

```
{
  "customer_id": 1,
  "quantity": 2,
  "price": 25.0,
  "status": "success"
}
```

Business Rules:

- Drive must be in "ongoing" status
- Customer must be in the drive's route
- Automatically updates drive's sold count
- Supports both new sales and updates to existing sales
- Quantity defaults to 1 if not provided

Sample Request:

```
POST /api/drive-execution/1/record-sale
Authorization: Bearer your-token-here
Content-Type: application/json

{
  "customer_id": 1,
  "quantity": 2,
  "price": 25.0,
  "status": "success"
}
```

Sample Response:

```
{
  "message": "Sale recorded successfully",
  "sale": {
    "drive_customer_sale_id": 15,
    "drive_id": 1,
    "customer_id": 1,
    "quantity": 2,
    "price": 25.0,
    "total_amount": 50.0,
    "status": "success",
    "qr_id": null,
    "remarks": null,
    "created_at": "2025-07-18T08:30:00Z",
    "updated_at": "2025-07-18T08:30:00Z"
  }
}
```

3. **POST /drive-execution/:id/skip-customer** - Skip Customer Delivery

Purpose: Mark a customer as skipped/failed delivery with a reason.

Path Parameters:

- `id` : Drive ID

Request Body:

```
{
  "customer_id": 1,
  "reason": "Customer not available"
}
```

Business Rules:

- Drive must be in "ongoing" status
- Customer must be in the drive's route
- Records as failed delivery with 0 quantity
- Supports updating existing skipped entries

Sample Request:

```
POST /api/drive-execution/1/skip-customer
Authorization: Bearer your-token-here
Content-Type: application/json

{
  "customer_id": 1,
  "reason": "Customer not available"
}
```

Sample Response:

```
{
  "message": "Customer skipped successfully",
  "skippedDelivery": {
    "drive_customer_sale_id": 16,
    "drive_id": 1,
    "customer_id": 1,
    "quantity": 0,
    "price": 25.0,
    "total_amount": 0,
    "status": "failed",
    "remarks": "Customer not available",
    "created_at": "2025-07-18T08:45:00Z"
  }
}
```

4. **POST /drive-execution/:id/scan-qr** - Scan QR Code for Delivery

Purpose: Process delivery using customer's QR code for quick and accurate service.

Path Parameters:

- `id` : Drive ID

Request Body:

```
{
  "qrCode": "QR_CODE_STRING",
  "quantity": 2
}
```

Business Rules:

- Drive must be in "ongoing" status
- QR code must be active and valid
- Customer must be in the drive's route
- Quantity defaults to customer's default quantity if not provided
- Automatically calculates total amount

Sample Request:

```
POST /api/drive-execution/1/scan-qr
Authorization: Bearer your-token-here
Content-Type: application/json

{
  "qrCode": "CUST_001_QR_2025",
  "quantity": 2
}
```

Sample Response:

```
{
  "message": "QR code scanned and sale recorded successfully",
  "customer": {
    "customer_id": 1,
    "name": "Alice Johnson",
    "phone": "9876543210"
  },
  "sale": {
    "drive_customer_sale_id": 17,
    "drive_id": 1,
    "customer_id": 1,
    "qr_id": 5,
    "quantity": 2,
    "price": 25.0,
    "total_amount": 50.0,
    "status": "success",
    "created_at": "2025-07-18T09:00:00Z"
  }
}
```


5. GET /drive-execution/:id/progress - Get Drive Progress

Purpose: Get real-time progress metrics and statistics for an ongoing drive.

Path Parameters:

- `id` : Drive ID

Sample Request:

```
GET /api/drive-execution/1/progress
Authorization: Bearer your-token-here
```

Sample Response:

```
{
  "progress": {
    "driveStatus": "ongoing",
    "start_time": "2025-07-18T06:00:00Z",
    "end_time": null,
    "elapsedTime": 9600,
    "total_customers": 15,
    "servedCustomers": 8,
    "remainingCustomers": 7,
    "completionPercentage": 53.33,
    "quantity_delivered": 45,
    "total_amount": 1125.0,
    "successful_deliveries": 7,
    "failed_deliveries": 1,
    "stock": 100,
    "remainingStock": 55,
    "lastLocation": {
      "longitude": 77.5946,
      "latitude": 12.9716,
      "time": "2025-07-18T08:45:00Z"
    }
  }
}
```


6. POST /drive-execution/:id/reconcile - Reconcile Drive

Purpose: Reconcile drive data before completion, ensuring accurate stock and sales figures.

Path Parameters:

- `id` : Drive ID

Request Body:


```
{
  "sold": 85,
  "returned": 15,
  "remarks": "Drive completed successfully. Minor discrepancies reconciled."
}
```

Business Rules:

- Drive must be in "ongoing" status
- Automatically calculates values from sales if not provided
- Updates drive with reconciled totals
- Prepares drive for completion

Sample Request:

```
POST /api/drive-execution/1/reconcile
Authorization: Bearer your-token-here
Content-Type: application/json

{
  "sold": 85,
  "returned": 15,
  "remarks": "Drive completed successfully. Stock reconciled."
}
```

Sample Response:


```
{
  "message": "Drive reconciled successfully",
  "drive": {
    "drive_id": 1,
    "route_id": 1,
    "delivery_guy_id": 1,
    "stock": 100,
    "sold": 85,
    "returned": 15,
    "status": "ongoing",
    "name": "Morning Route A",
    "total_amount": 2125.0,
    "remarks": "Drive completed successfully. Stock reconciled.",
    "created_at": "2025-07-18T06:00:00Z",
    "updated_at": "2025-07-18T10:30:00Z"
  }
}
```

Authentication

All endpoints require Bearer token authentication:

```
Authorization: Bearer your-jwt-token-here
```

Error Responses

All endpoints return consistent error responses:

400 Bad Request:

```
{
  "message": "Drive is not currently ongoing"
}
```

404 Not Found:

```
{
  "message": "Drive not found"
}
```

500 Internal Server Error:

```
{
  "message": "Server error"
}
```

Business Rules & Workflow

Drive Execution Flow:

1. **Drive must be "ongoing"** - Most operations only work on ongoing drives
2. **Customer Validation** - All operations validate customer is in the drive's route
3. **Real-time Updates** - Drive totals are automatically updated with each sale
4. **Flexible Quantities** - Quantity defaults are provided but can be overridden
5. **QR Code Integration** - Supports both manual entry and QR code scanning

6. **Progress Tracking** - Real-time progress metrics available at all times

Data Integrity:

- **Transactions** - All database operations use transactions for consistency
- **Conflict Resolution** - Supports updating existing sales records
- **Automatic Calculations** - Total amounts calculated automatically
- **Stock Tracking** - Real-time stock remaining calculations

Mobile App Integration:

- **Offline Capability** - Designed to work with mobile apps that may go offline
- **QR Code Support** - Quick customer identification and delivery
- **Progress Monitoring** - Real-time progress for supervisors
- **Location Tracking** - Integrates with GPS location logging

Database Tables Used

- `drives` : Main drive information and totals
- `drive_customers_sales` : Individual customer delivery records
- `customers` : Customer information and pricing
- `route_customers` : Route-customer relationships
- `qr_codes` : QR code mappings to customers
- `drive_locations_log` : GPS tracking data
- routes: Route information
- `delivery_guys` : Delivery person information

Use Cases

1. **Delivery Personnel** - Record sales, skip customers, scan QR codes
2. **Supervisors** - Monitor progress, track completion rates
3. **Back Office** - Reconcile drives, audit delivery data
4. **Mobile Apps** - Real-time drive execution interface
5. **Analytics** - Performance metrics and completion tracking

This controller is the core of the delivery execution system, providing all necessary functionality for efficient and accurate milk delivery operations.