Waste Collector Service Implementation Guide

Project Overview

The Waste Collector Service is a critical component of our RWDP (Recycling Waste Disposal Platform) that manages all aspects of waste collector operations. This service handles collector profiles, driver management, vehicle tracking, service offerings, and pickup request assignment.

Architecture Overview

The Waste Collector Service follows our microservice architecture pattern:

- RESTful API endpoints for synchronous operations
- Kafka event consumers/producers for asynchronous workflows
- PostgreSQL database for persistent storage
- Redis for caching frequently accessed data

Database Schema

Main Tables

waste_collector

Column	Type	Description
collector_id	SERIAL	Primary key
user_id	INTEGER	Foreign key to users table
company_name	VARCHAR(255)	Company name
license_number	VARCHAR(100)	Waste collection license
authorized_categories	TEXT	Categories of waste handled (comma-separated)
capacity	INTEGER	Collection capacity in kg

is_verified	BOOLEAN	Whether collector is verified
service_rating	FLOAT	Overall service rating
license_expiry	DATE	License expiration date
documents_verified	BOOLEAN	Whether documents are verified
avg_rating	FLOAT	Average rating from all trips

service_category

Column	Type	Description
category_id	SERIAL	Primary key
collector_id	INTEGER	Foreign key to waste_collector
waste_type	VARCHAR(100)	Type of waste accepted
price_per_kg	FLOAT	Cost per kg for this waste type
maximum_capacity	FLOAT	Max capacity for this waste type
handling_requirements	TEXT	Special handling instructions

vehicle

Column	Type	Description
vehicle_id	SERIAL	Primary key
collector_id	INTEGER	Foreign key to waste_collector
vehicle_number	VARCHAR(50)	Vehicle registration number
vehicle_type	VARCHAR(50)	Type of vehicle
capacity	FLOAT	Vehicle capacity in kg
maintenance_date	DATE	Last maintenance date
is_active	BOOLEAN	Whether vehicle is in service

gps_tracking_id	VARCHAR(100)	GPS tracking device ID
assigned_driver_id	INTEGER	Currently assigned driver
registration_document	VARCHAR(255)	Path to registration document
registration_expiry	DATE	Registration expiration date

driver

Column	Type	Description
driver_id	SERIAL	Primary key
user_id	INTEGER	Foreign key to users table
collector_id	INTEGER	Foreign key to waste_collector
license_number	VARCHAR(100)	Driver's license number
license_expiry	DATE	License expiration date
assigned_vehicle_id	INTEGER	Currently assigned vehicle
is_active	BOOLEAN	Whether driver is available for trips
rating	FLOAT	Driver's performance rating
joining_date	DATE	Date when driver joined

driver_location

Column	Type	Description
id	SERIAL	Primary key
driver_id	INTEGER	Foreign key to driver
latitude	FLOAT	Current latitude
longitude	FLOAT	Current longitude

timestamp	TIMESTAMP	When location was recorded
is_active	BOOLEAN	Whether driver is active
trip_id	INTEGER	Current trip ID (if any)
vehicle_id	INTEGER	Current vehicle ID

Required API Endpoints

Collector Management

1. `GET /collectors` - List waste collectors with filtering options
2. `GET /collectors/:id` - Get specific collector details
3. `PUT /collector/profile` - Update collector profile (authenticated)
4. `GET /collector/dashboard` - Get collector dashboard stats (authenticated)

Service Categories

5. `GET /collectors/:id/service-categories` - Get services offered by a collector
6. `POST /collector/service-categories` - Add a new service category (authenticated)
7. `PUT /collector/service-categories/:id` - Update service category (authenticated)
8. `DELETE /collector/service-categories/:id` - Delete service category (authenticated)

Vehicle Management

9. `GET /collectors/:id/vehicles` - Get vehicles owned by a collector
10. `POST /collector/vehicles` - Add a new vehicle (authenticated)
11. `PUT /collector/vehicles/:id` - Update vehicle details (authenticated)
12. `PUT /collector/vehicles/:id/activate` - Activate vehicle (authenticated)
13. `PUT /collector/vehicles/:id/deactivate` - Deactivate vehicle (authenticated)

Driver Management

14. `GET /collectors/:id/drivers` - List all drivers for a waste collector

15. `GET /collector/drivers/:id` - Get specific driver details (authenticated)
16. `POST /collector/drivers` - Register a new driver (authenticated)
17. `PUT /collector/drivers/:id` - Update driver details (authenticated)
18. `PUT /collector/drivers/:id/assign-vehicle` - Assign vehicle to driver (authenticated)
19. `POST /driver/location` - Update driver's current location (driver authentication)

Trip and Pickup Management

20. `GET /collector/pickup-requests` - List pickup requests assigned to collector (authenticated)
21. `GET /collector/pickup-requests/:id` - Get pickup request details (authenticated)
22. `PUT /collector/pickup-requests/:id/assign` - Assign driver and vehicle to pickup request (authenticated)
23. `GET /driver/trips` - Get trips assigned to a driver (driver authentication)
24. `PUT /driver/trips/:id/status` - Update trip status (driver authentication)

Event Handling

Consumed Events

1. `PickupRequestCreated` - When a business creates a pickup request
2. `TripScheduled` - When a trip is scheduled in the pickup service
3. `TripCompleted` - When a trip is marked as completed
4. `UserVerified` - When a collector's verification status changes

Produced Events

1. `PickupRequestAssigned` - When a collector assigns a driver/vehicle to a request
2. `DriverAssigned` - When a driver is assigned to a pickup request
3. `DriverLocationUpdated` - When a driver's location is updated
4. `VehicleStatusUpdated` - When a vehicle's status changes

Implementation Requirements

Models (internal/wastecollector/models.go)

- Define all data structures corresponding to database tables
- Define request/response structures for API endpoints

- Define event payloads for Kafka integration

Repository Layer (internal/wastecollector/repository.go)

- Implement CRUD operations for all entities
- Use prepared statements for all database queries
- Implement filtering and pagination for list operations
- Use transactions for operations that update multiple tables

Service Layer (internal/wastecollector/service.go)

- Implement business logic for all operations
- Handle validation and error checking
- Implement trip assignment algorithms
- Manage event production and consumption

Handler Layer (internal/wastecollector/handler.go)

- Implement REST API endpoints
- Handle request parsing and validation
- Format responses according to API standards
- Implement authentication/authorization checks

Event Handler (internal/wastecollector/events.go)

- Implement Kafka event consumption
- Process event payloads
- Trigger appropriate service methods
- Handle error conditions and retries

Trip Assignment Algorithm

The waste collector service should implement a trip assignment algorithm that:

1. Identifies available drivers and vehicles for a given pickup request
2. Considers proximity to pickup location (using driver_location data)
3. Factors in vehicle capacity vs. estimated waste volume
4. Considers driver ratings and performance history
5. Accounts for vehicle capabilities for the waste type
6. Optimizes for efficiency (minimizing travel distance and time)

Authentication and Authorization

- All `/collector/*` endpoints require waste collector role authentication
- All `/driver/*` endpoints require driver role authentication
- A collector can only manage their own data (vehicles, drivers, etc.)
- A driver can only update their own location and trips