Project: Urban Waste Collection Management System

1. Introduction

The purpose of this document is to provide a detailed Low-Level Design (LLD) for the **Urban Waste Collection Management System**. This system supports municipalities or urban service providers in managing waste collection zones, assigning vehicles and workers, scheduling pickups, and monitoring waste collection activities effectively.

This design supports both Java (Spring Boot) and .NET (ASP.NET Core) frameworks.

2. Module Overview

The system consists of the following functional modules:

2.1 Zone and Route Management

Defines and manages collection zones and optimizes waste collection routes.

2.2 Vehicle Assignment and Tracking

Assigns garbage trucks to specific zones and tracks their movement in real time.

2.3 Pickup Scheduling

Allows scheduling of daily or weekly waste pickups based on predefined frequencies.

2.4 Worker Management

Manages sanitation workers' shifts, attendance, and zone-wise assignments.

2.5 Waste Collection Logs

Records waste pickup data including weight, time, and disposal location for audit and analysis.

3. Architecture Overview

3.1 Architectural Style

- **Frontend**: Angular or React
- **Backend**: REST API-based architecture
- **Database**: Relational Database (MySQL/SQL Server)

3.2 Component Interaction

- Frontend interacts with backend via secure HTTP/HTTPS.
- Backend communicates with the database for CRUD operations.
- Backend handles business logic and data validation.

- Frontend handles UI rendering and user inputs.

4. Module-Wise Design

4.1 Zone and Route Management Module

4.1.1 Features

- Define geographic waste collection zones
- Create and manage optimized collection routes per zone

4.1.2 Data Flow

- Admin configures zones/routes via frontend.
- Backend saves route details into the database.
- Updated route data is pushed to vehicle assignment module.

4.1.3 Entities

- **Zone**: ZoneID, Name, AreaCoverage
- **Route**: RouteID, ZoneID (FK), PathDetails, EstimatedTime

4.2 Vehicle Assignment and Tracking Module

4.2.1 Features

- Assign vehicles to zones/routes
- Monitor real-time vehicle status and location (simplified tracking)

4.2.2 Data Flow

- Admin assigns trucks via frontend.
- Backend links vehicle to route/zone.
- Vehicle updates (status, location) are polled from simplified endpoint.

4.2.3 Entities

- **Vehicle**: VehicleID, RegistrationNo, Type, Status
- **Assignment**: AssignmentID, VehicleID, RouteID, DateAssigned

4.3 Pickup Scheduling Module

4.3.1 Features

- Configure recurring or ad-hoc pickup schedules
- Notify operations team of upcoming tasks

4.3.2 Data Flow

- Scheduler creates pickup jobs.
- Backend processes and persists schedule.
- Jobs pushed to workers and vehicle modules.

4.3.3 Entities

- **PickupSchedule**: ScheduleID, ZoneID, Frequency (Daily/Weekly), TimeSlot, Status

4.4 Worker Management Module

4.4.1 Features

- Add/update sanitation workers
- Assign workers to zones and routes

4.4.2 Data Flow

- Admin manages workers via frontend.
- Backend updates worker and assignment records.
- Integration with scheduling and assignment modules.

4.4.3 Entities

- **Worker**: WorkerID, Name, ContactInfo, Role
- **WorkerAssignment**: AssignmentID, WorkerID, ZoneID, ShiftTime

4.5 Waste Collection Logs Module

4.5.1 Features

- Log each collection with date, time, weight
- Allow report generation for audits

4.5.2 Data Flow

- Collection data is entered by workers or auto-fed by sensors.
- Backend stores collection log.
- Admin accesses logs for reports.

4.5.3 Entities

- **WasteLog**: LogID, ZoneID, VehicleID, WorkerID, WeightCollected, CollectionTime

5. Deployment Strategy

5.1 Local Deployment

For development and testing, the system can be deployed on local machines.

- **Frontend**: Use ng serve (Angular) or npm start (React)
- **Backend**: Run with Spring Boot (mvn spring-boot:run) or ASP.NET Core (dotnet run)
- **Database**: MySQL/PostgreSQL/SQL Server instance on localhost

6. Database Design

6.1 Tables and Relationships

- **Zone**: ZoneID (PK), Name, AreaCoverage
- **Route**: RouteID (PK), ZoneID (FK), PathDetails, EstimatedTime
- **Vehicle**: VehicleID (PK), RegistrationNo, Type, Status
- **Worker**: WorkerID (PK), Name, ContactInfo, Role
- **Assignment**: AssignmentID (PK), VehicleID (FK), RouteID (FK), DateAssigned
- **WorkerAssignment**: AssignmentID (PK), WorkerID (FK), ZoneID (FK), ShiftTime
- **PickupSchedule**: ScheduleID (PK), ZoneID (FK), Frequency, TimeSlot, Status
- **WasteLog**: LogID (PK), ZoneID (FK), VehicleID (FK), WorkerID (FK), WeightCollected, CollectionTime

7. User Interface Design

7.1 Wireframes

- Admin Dashboard
- Zone/Route Configuration Interface
- Vehicle Assignment Page
- Pickup Scheduler Interface
- Worker Roster Page
- Waste Log Viewer and Report Generator

8. Non-Functional Requirements

8.1 Performance

- Capable of handling operations across 25 zones and up to 50 concurrent users.

8.2 Scalability

- Modular design allows easy scaling to new urban areas or states.

8.3 Security

- Role-based access (Admin, Worker, Supervisor), encrypted communication.

8.4 Usability

- Simple dashboard layout, real-time updates, mobile-responsive UI.

9. Assumptions and Constraints

9.1 Assumptions

- Vehicle tracking is simplified to periodic location updates.
- Waste weights are entered manually or via integrated scales.

9.2 Constraints

- No third-party cloud or IoT integration in current phase.
- Operates in a local network or private server deployment only.