Borla Tracker (Waste Management System) API Documentation

Purpose of the API

This API provides a **unified backend for managing waste collection operations**. Its purpose is to digitize and streamline workflows for both private and company-employed collectors, supervisors, and clients. Key capabilities include:

- **Efficient assignment of collectors** for on-demand and scheduled collection requests.
- **Supervisor oversight** of routes, progress, and collector performance.
- **Client interaction** for creating requests, tracking collections, and managing wallets.
- **Automated business logic** for incentives, penalties, and compliance monitoring.
- **Audit-ready records** to ensure accountability, reporting, and operational transparency

Actors

- **Client** → End-user who creates requests (on-demand or scheduled), pays for services, and tracks status.
- **Collector** → Worker who performs waste collection. Can be:
    - **Private collector** (self-registered).
    - **Company collector** (created/approved by a company and supervised).
- **Supervisor** → Oversees company collectors, monitors routes, approves registrations, and ensures compliance.
- **Company** → Organizational entity that creates supervisors, manages collectors, and owns routes.

- **Admin** → System-level actor who manages roles, monitors statistics, and ensures platform integrity.

🔗 CLICK HERE TO VIEW USE CASE DIAGRAM

Domain Models

UML (Fields Only)

Each class lists its attributes for clarity. Methods are documented separately.

- **User**
  - Fields: username, phone_number, email, role, profile_photo, address, is_verified, is_active, is_staff, created_at.
- **Client**
  - Fields: first_name, last_name, alternate_phone, wallet_balance, segregation_compliance_percent, registration_date.
- **Collector**
  - Fields: first_name, last_name, company, supervisor, is_private_collector, vehicle_number, vehicle_type, assigned_area_zone, daily_wage_or_incentive_rate, bank_account_details, average_rating, total_collections, last_known_latitude, last_known_longitude.
- **Company**
  - Fields: name, registration_number, address, created_at.
- **Supervisor**
  - Fields: first_name, last_name, assigned_zone, created_at.
- **OnDemandRequest**

    - Fields: pickup_date, pickup_time_slot, address_line1, landmark, area_zone, city, latitude, longitude, location, bag_count, bin_size_liters, waste_type, special_instructions, waste_image, quoted_price, final_price, payment_status, request_status, requested_at, accepted_at, completed_at, cancelled_at, cancellation_reason, created_at, updated_at.
- **ScheduledRequest**
    - Fields: pickup_date, pickup_time_slot, recurrence, request_status, created_at, updated_at.
- **Route**
    - Fields: route_date, start_time, end_time, actual_start, actual_end, completion_percent, status, total_distance_km, estimated_duration, created_at, updated_at.
- **RouteStop**
    - Fields: order, expected_minutes, actual_start, actual_end, status, notes, created_at, updated_at.
- **CollectionRecord**
    - Fields: collection_id, payment_method, amount_paid, collection_type, scheduled_date, collection_start, collection_end, collected_at, bag_count, bin_size_liters, estimated_volume_liters, waste_type, photo_before, photo_after, segregation_score, status, latitude, longitude, notes, created_at, updated_at, duration_minutes.

Methods

- **User** → save() (auto-generate username), __str__().
- **Client** → __str__().
- **Collector** → full_name(), __str__().
- **OnDemandRequest** → calculate_quoted_price(), save() (auto-populate fields), __str__().

- **ScheduledRequest** → __str__().
- **Route** → update_distance_and_duration(), update_completion_status(), save(), __str__().
- **RouteStop** → __str__().
- **CollectionRecord** → save() (timestamps/duration), get_volume_description(), verify_location(), __str__().

Relationships

- **Inheritance** → Client, Collector, Company, Supervisor all extend User.
- **Company → Supervisor/Collector** → Company creates supervisors and collectors.
- **Client → Requests** → Client creates OnDemandRequest and ScheduledRequest.
- **Collector → Requests/Records** → Collector executes requests and logs CollectionRecords.
- **Supervisor → Routes** → Supervisor oversees routes and collectors.
- **Route → RouteStop** → Route contains stops linked to requests.
- **CollectionRecord → Client/Collector/RouteStop** → Immutable evidence tying together actors and operations.

🔗 **CLICK HERE TO VIEW UML DIAGRAM**

Authentication & Authorization

This API uses **JSON Web Tokens (JWT)** to provide secure, stateless authentication across all protected endpoints.

Upon successful login, the system issues two tokens:

- **Access Token** → Short-lived token used to authenticate API requests.
- **Refresh Token** → Long-lived token used to obtain a new access token when the current one expires.

Clients must include the access token in the Authorization header for all authenticated requests:

Authorization: Bearer <access_token>

Token Lifecycle

- Access tokens expire automatically after a defined duration.
- Refresh tokens are used to generate new access tokens without re-authentication.
- During logout, refresh tokens are invalidated to prevent reuse.

Role-Based Authorization

Access to resources and actions is strictly controlled based on user roles:

- **Clients** → Can create and track their own requests and manage payments.
- **Collectors** → Can view and update only their assigned requests.
- **Supervisors** → Can assign collectors, monitor routes, and view company-level data.
- **Companies** → Can manage supervisors, collectors, and routes.
- **Admins** → Have system-wide visibility and control.

Authorization rules are enforced at the API level, ensuring users cannot access or modify unauthorized data.

Security Considerations

The system applies multiple security controls to protect data integrity and operational trust:

- User credentials are securely hashed and never stored in plain text.
- JWT tokens are validated on every protected request.
- Role-based access control prevents unauthorized operations.

- GPS proximity validation reduces fraudulent collection confirmations.
- Collection records are immutable to support auditing and accountability.

These measures ensure the platform remains secure, reliable, and resistant to misuse.

Error Handling & Status Codes

The API follows standard HTTP status codes to communicate request outcomes clearly:

Status Code	Description
200	Request completed successfully
201	Resource created successfully
400	Invalid request or validation error
401	Unauthorized (missing or invalid token)
403	Forbidden (insufficient permissions)
404	Resource not found
500	Internal server error

Error responses are returned in a consistent JSON format to support predictable client-side handling.

Access Control Model

Authentication is handled globally using JWT, while authorization is enforced through role-aware permissions and data filtering.

- Users can only access resources relevant to their role.
- Query results are automatically filtered to prevent data leakage.
- Sensitive actions require elevated privileges (e.g., supervisor or admin roles).

This layered approach ensures both security and operational correctness.

Non-Functional Requirements (Critical Only)

This section outlines the key non-functional requirements that guide the expected performance, scalability, security, maintainability, and reliability of the system. These requirements represent design targets rather than strict guarantees and reflect realistic expectations for an academic project.

Performance

- **Response Time:**
  The system is designed to respond to API requests within **approximately two (2) seconds** under typical operating conditions.

Scalability

- **Concurrent Requests:**
  The system is intended to support **up to 1 thousand (1,000) active requests per day per zone** without significant performance degradation.
- **Collector Tracking:**
  Real-time GPS tracking is designed to scale to **approximately one thousand (1,000) concurrent collectors per city**, assuming normal usage patterns.

Security

- **Role-Based Access Control (RBAC):**
  Access to system features is controlled based on user roles, including **Client, Collector, Supervisor, Company, and Admin**, and is enforced at the API level.
- **Data Protection:**
  Sensitive data such as wallet information, payment details, and GPS coordinates is protected through encryption during data transmission and storage.
- **Audit Logging:**
  Collection records are treated as immutable once created, helping to support transparency and accountability.

Maintainability

- **Modular Design:**
  The system follows a modular design approach, with Django applications organized by domain (e.g., Requests, Routes, Collectors, Companies) to simplify development and future changes.
- **Documentation:**
  API documentation generated through **Swagger UI and ReDoc** is maintained to reflect the current state of the system and support frontend development and testing.

Reliability

- **Availability:**
  The system aims to achieve **high availability** for core request handling and assignment features during normal operational periods.

API Endpoints

This API exposes its endpoints through **auto-generated OpenAPI documentation**, which serves as the authoritative reference for all available resources, request/response schemas, and authentication requirements.

Swagger UI (Interactive Documentation)

Swagger UI provides an interactive interface for exploring and testing API endpoints in real time.

- Allows developers to browse all available endpoints grouped by resource
- Displays required headers, parameters, and request body schemas
- Shows example responses and HTTP status codes
- Supports authenticated testing using access tokens (where enabled)

Access Swagger UI:

https://kyei-ernest.github.io/wms/documentation/swagger/

ReDoc (Reference Documentation)

ReDoc provides a clean, read-only reference view of the API specification, optimized for clarity and long-form reading.

- Presents endpoints in a structured, easy-to-navigate format
- Clearly documents request and response models
- Highlights authentication and authorization requirements
- Suitable for external partners and non-technical stakeholders

Access ReDoc:

https://kyei-ernest.github.io/wms/documentation/redoc/

Both Swagger UI and ReDoc expose the **complete OpenAPI schema**, including:

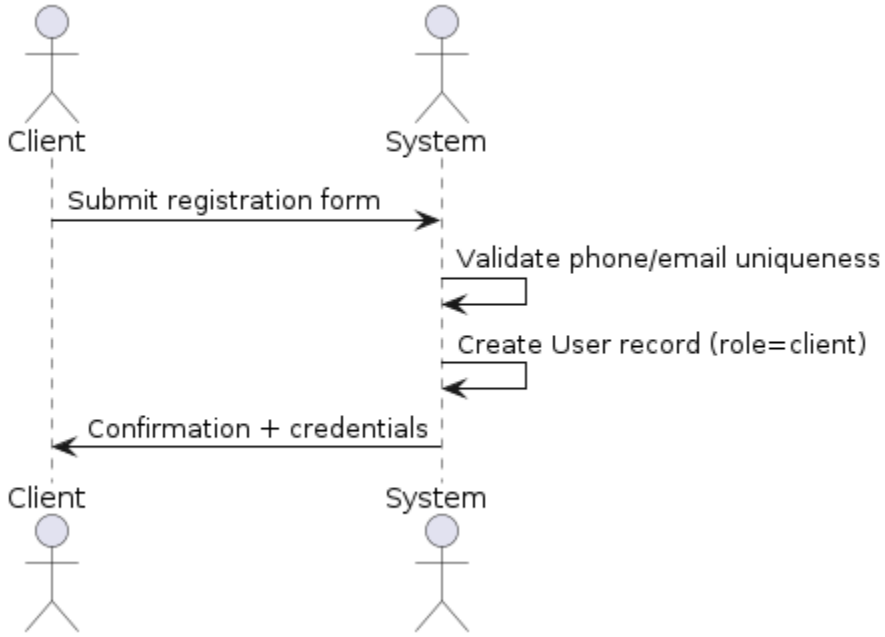
- All available API endpoints
- HTTP methods (GET, POST, PUT, PATCH, DELETE)
- Request parameters and body schemas
- Response payloads and error formats
- Authentication mechanisms (e.g., JWT, API keys)
- Role-based access constraints (e.g., Admin, Supervisor, Collector)

These documentation links should be considered the **single source of truth** for endpoint definitions and usage.

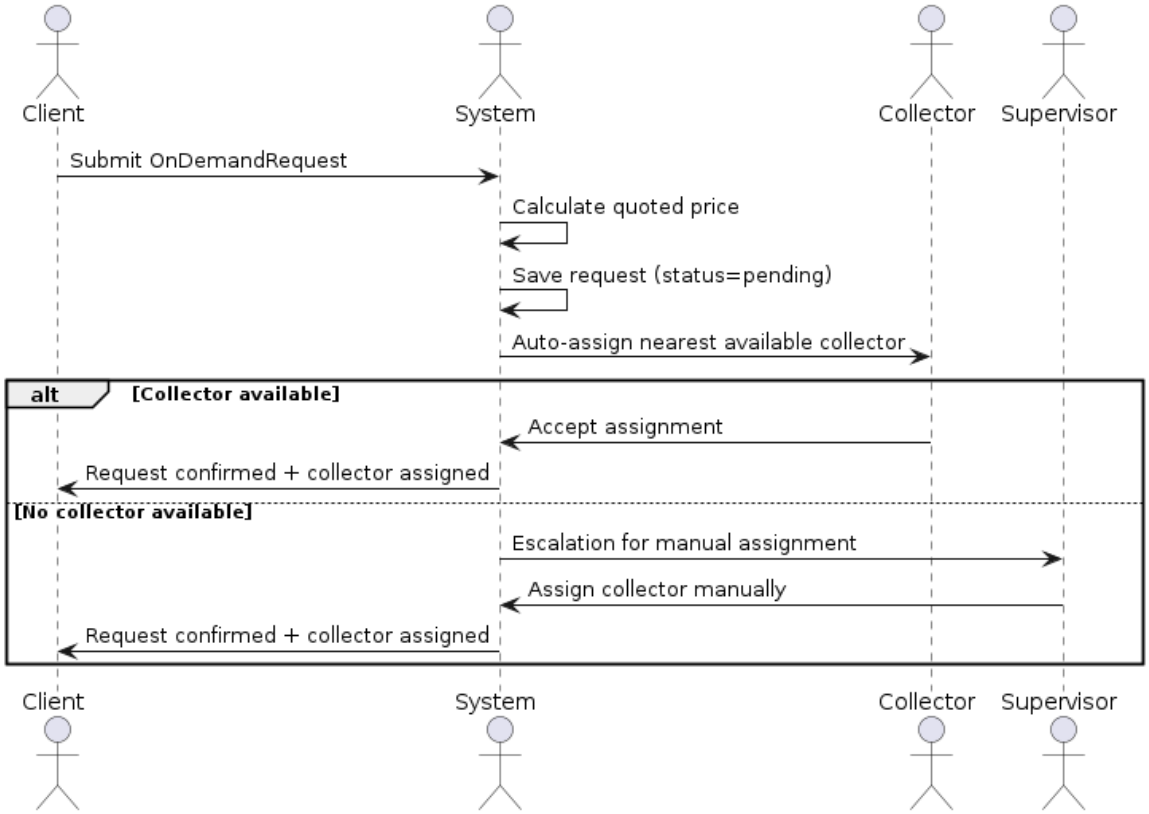
Operational Workflows

1. Client Workflows

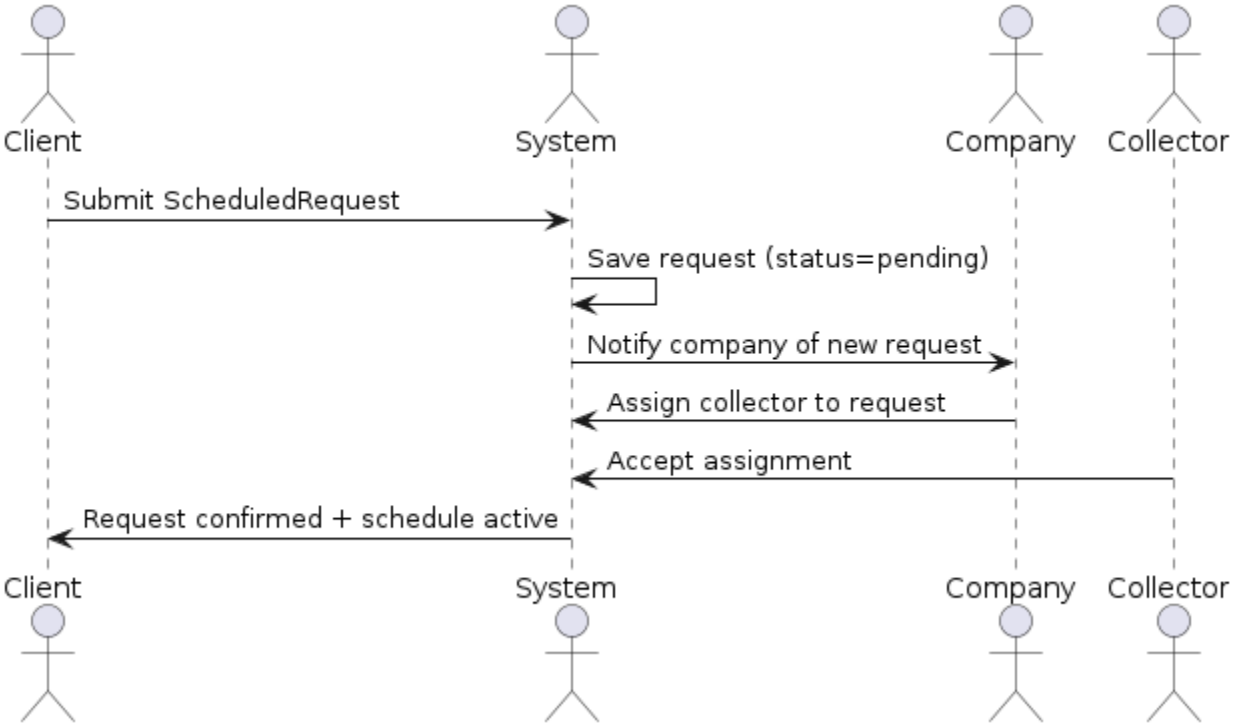
- **Account Registration** → Client creates a user account.



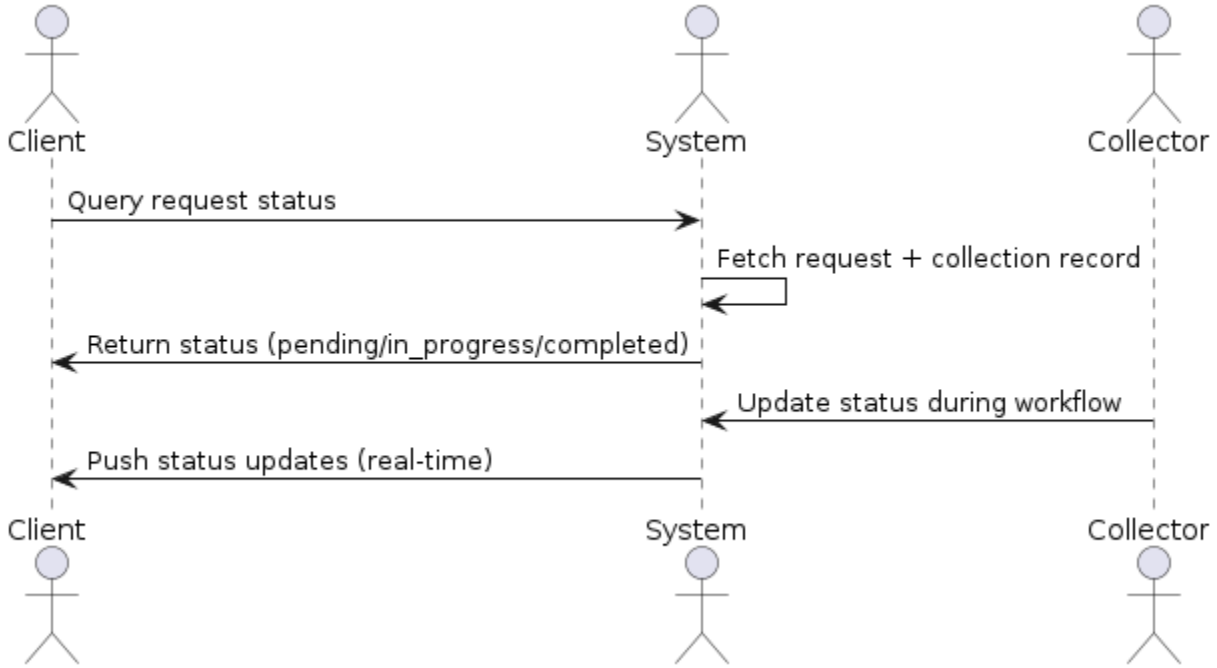
- **Create On-Demand Request** → Client submits one-time pickup request.



- **Create Scheduled Request** → Client sets up recurring pickups.



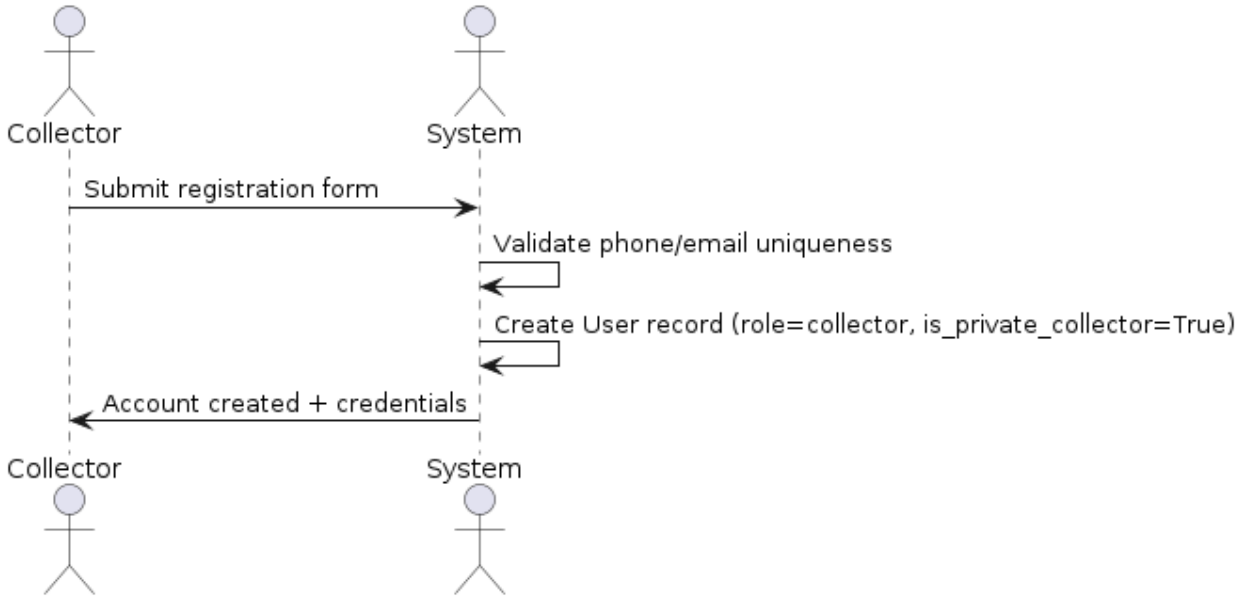
- **Track Request Status** → Client monitors progress of their requests.



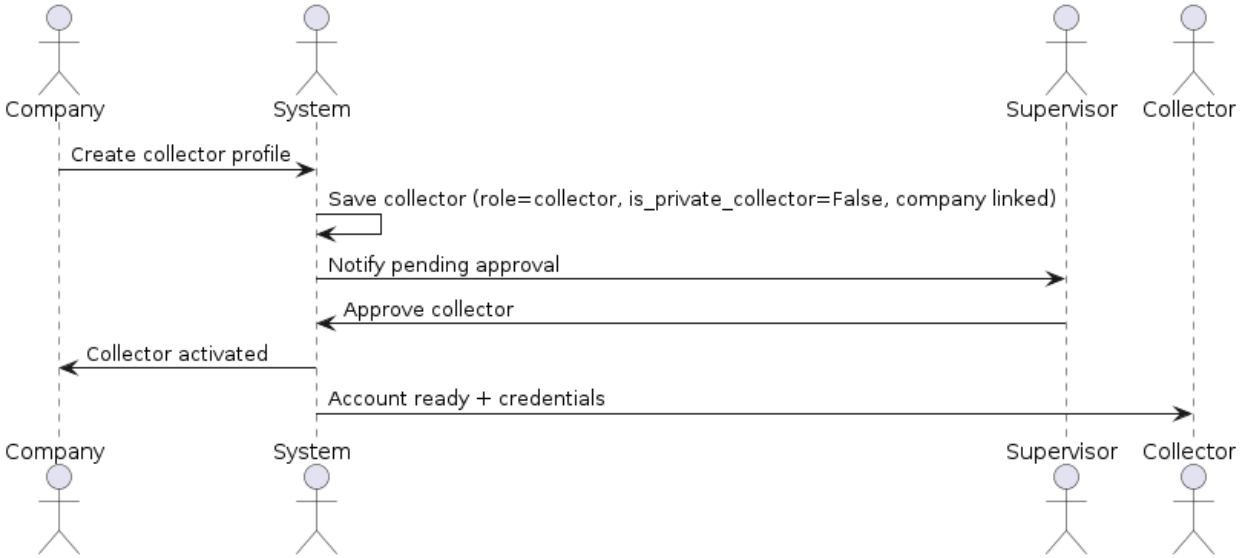
- **Wallet & Payment Flow** → Client pays via cash, mobile money, or bank transfer; wallet balance updated.

2. Collector Workflows

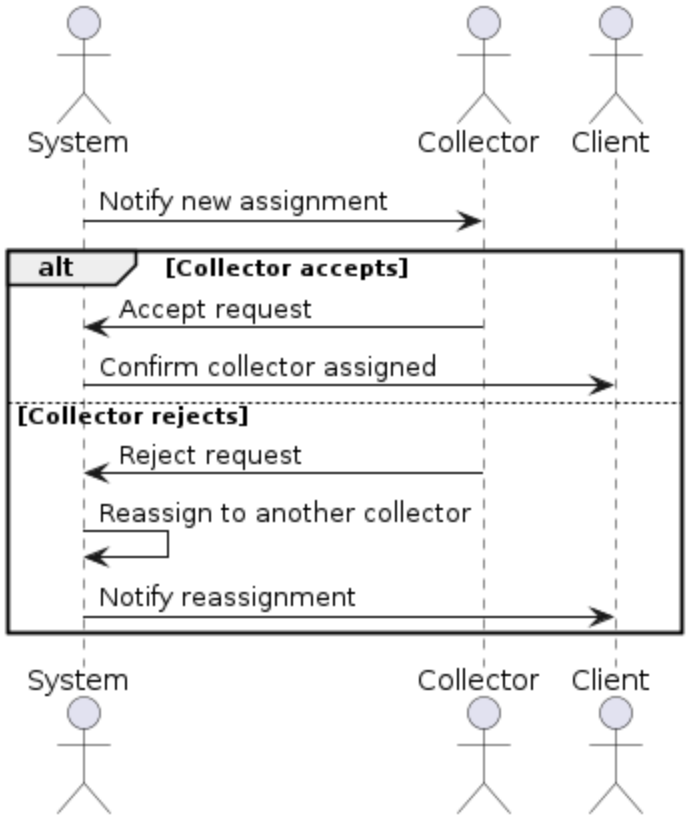
- **Private Collector Registration** → Collector self-registers as independent.



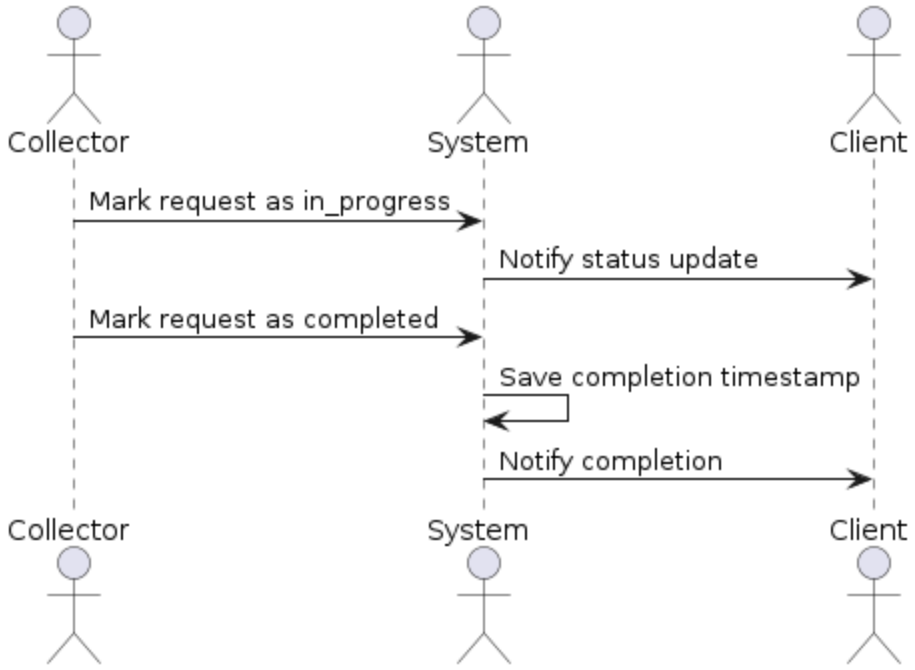
- **Company Collector Onboarding** → Collector created by company, approved by supervisor.



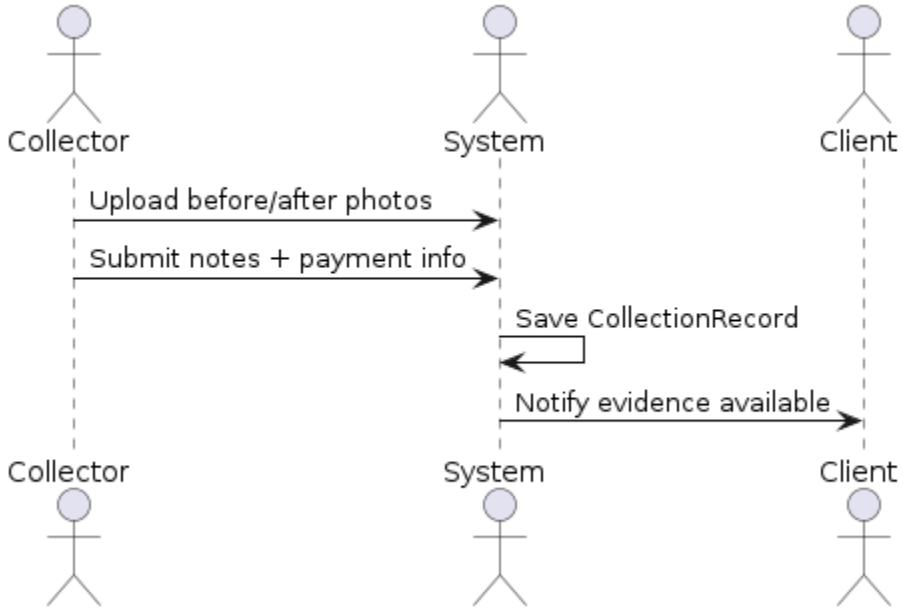
- **Accept Assigned Request** → Collector accepts on-demand or scheduled assignment.



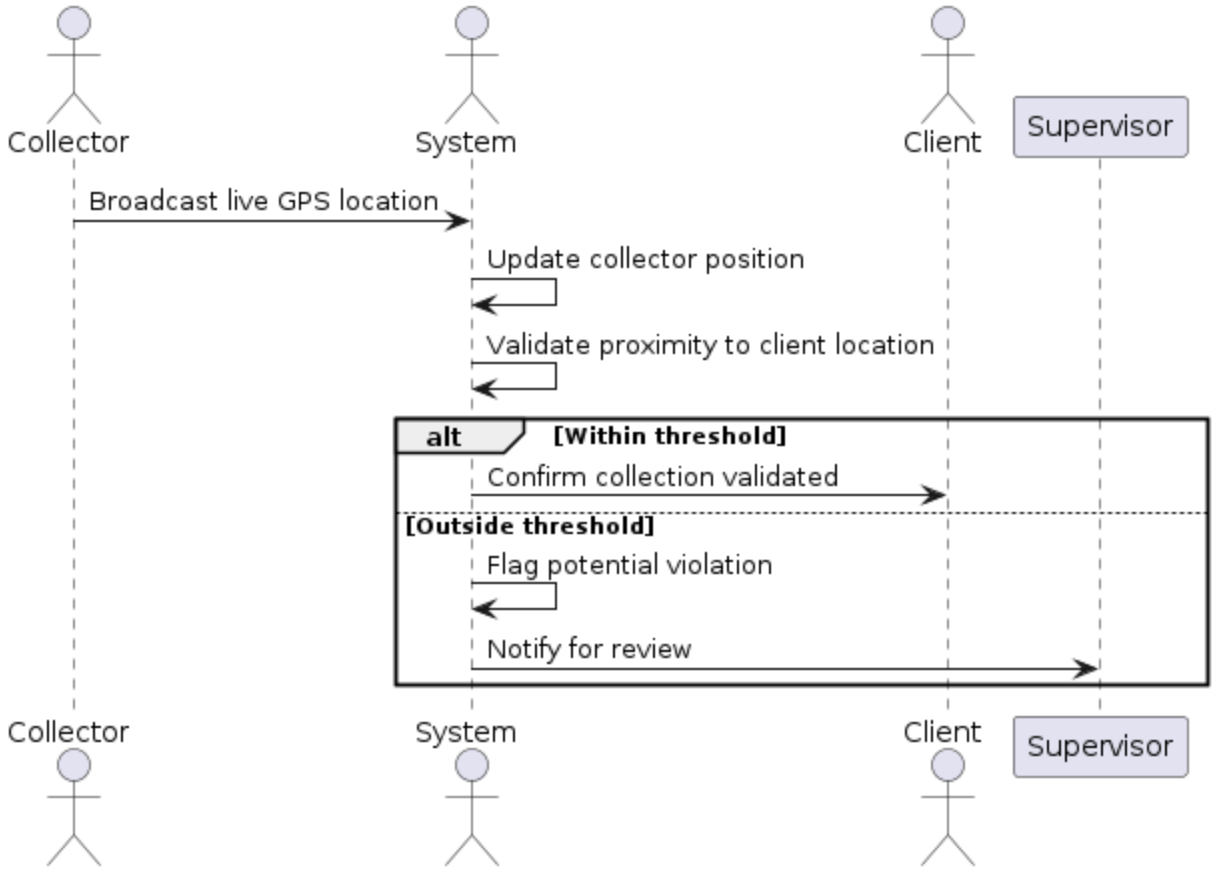
- **Update Collection Status** → Collector marks request as in-progress, completed, skipped, etc.



- **Upload Collection Evidence** → Collector submits before/after photos, notes, payment info.

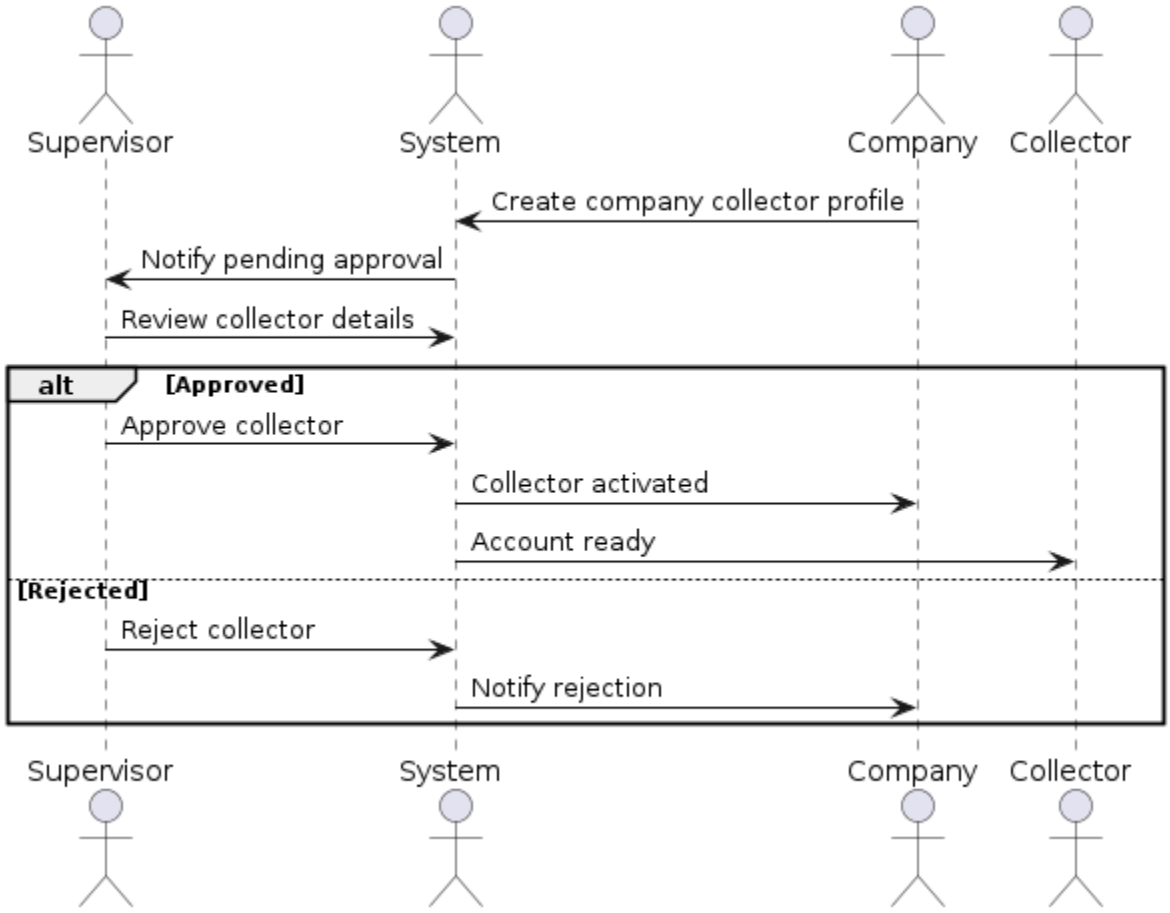


- **GPS Broadcast / Validation** → Collector shares live location; system validates completion proximity.

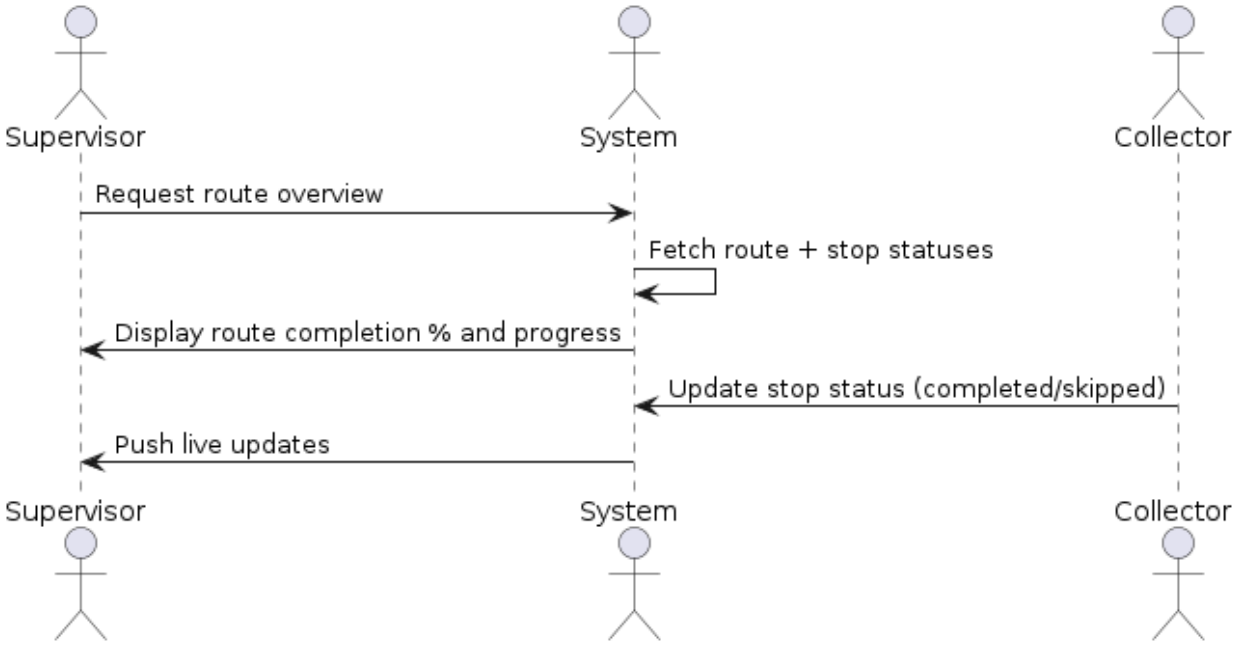

Collector
System
Client
Supervisor
Broadcast live GPS location
Update collector position
Validate proximity to client location
alt
[Within threshold]
Confirm collection validated
[Outside threshold]
Flag potential violation
Notify for review
Collector
System
Client
Supervisor

3. Supervisor Workflows

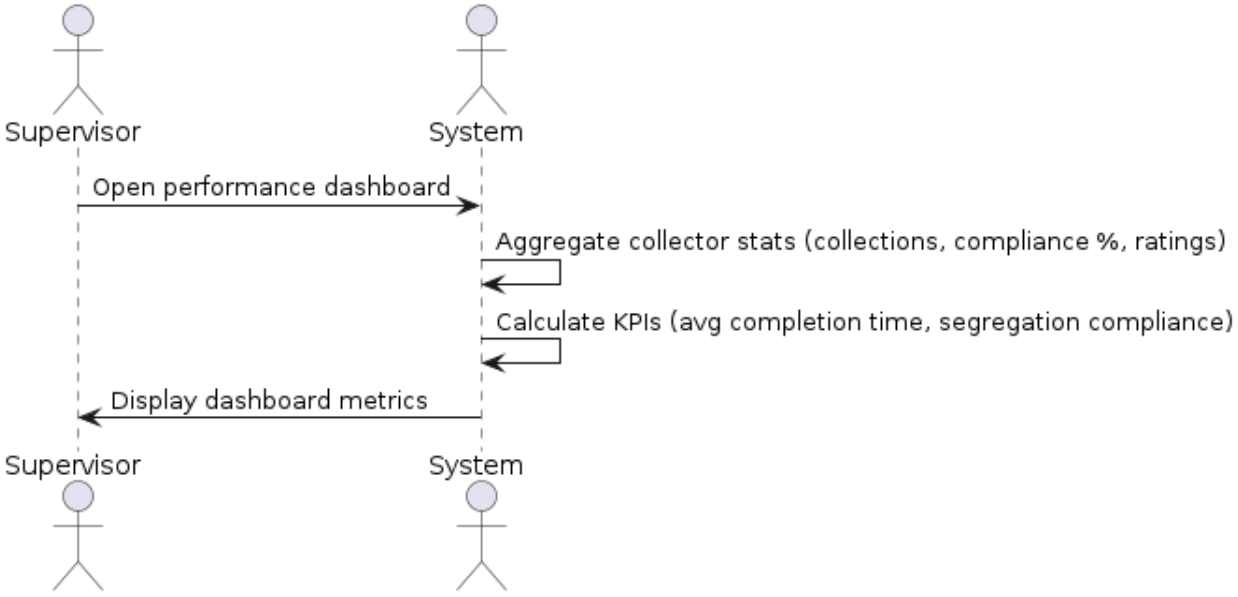
- **Approve Company Collector** → Supervisor validates new company collector accounts.



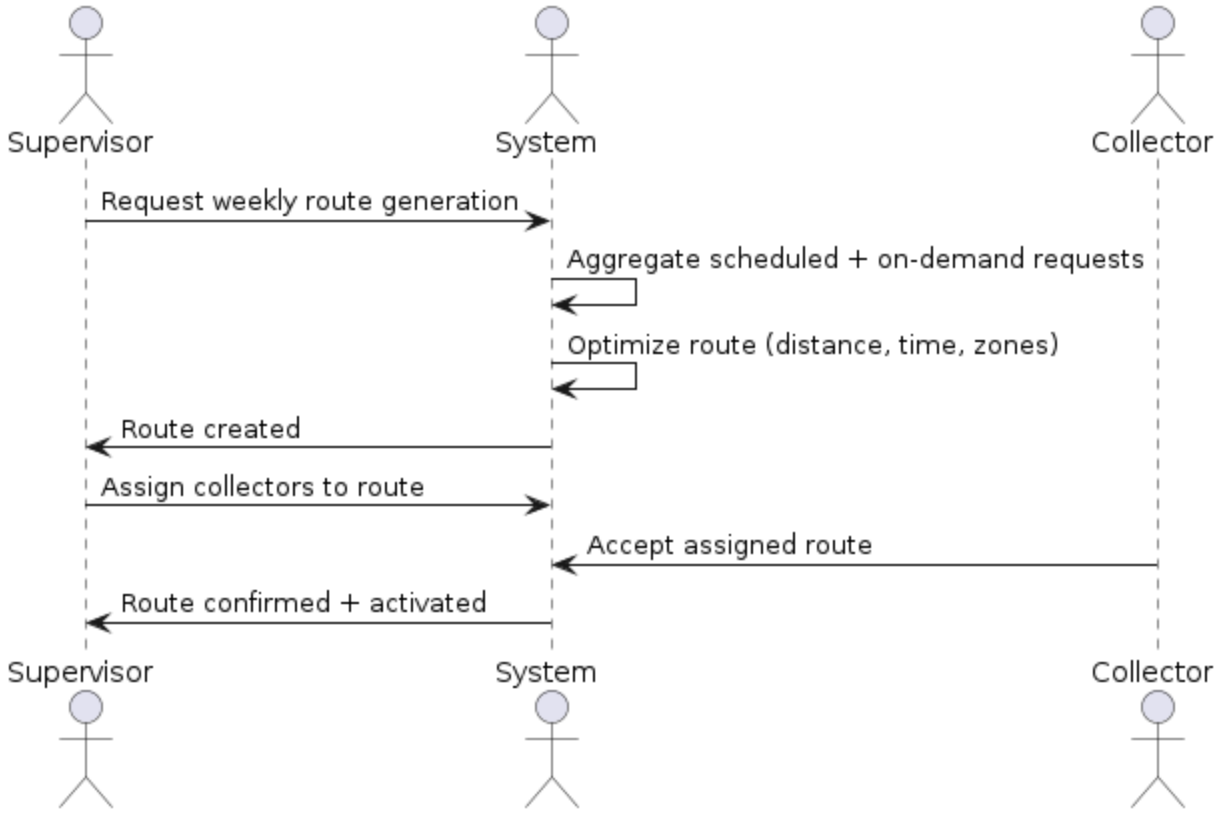
- **Monitor Routes** → Supervisor tracks route progress and completion percentage.



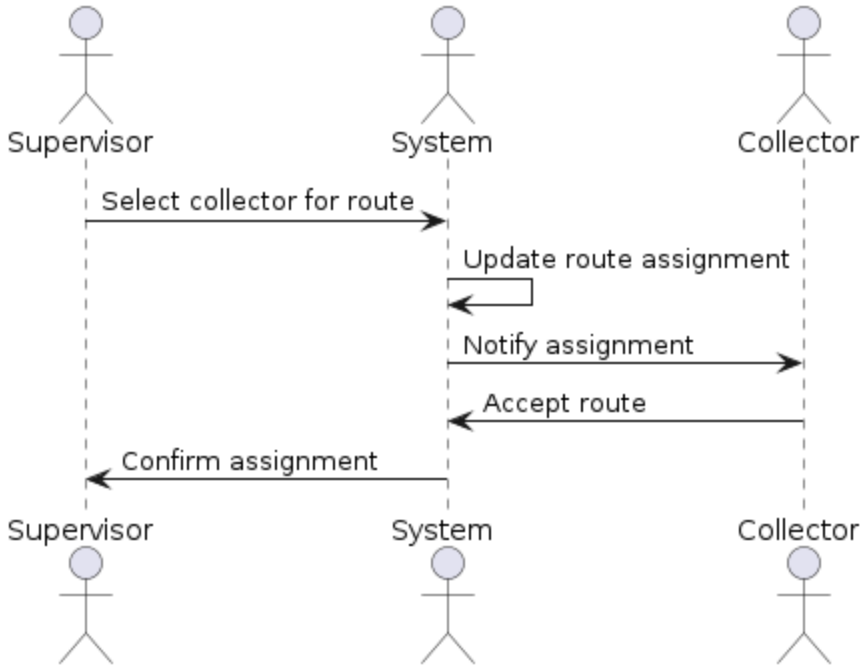
- **View Performance Dashboard** → Supervisor reviews collector stats, compliance, and KPIs.



- **Generate Weekly Routes** → Company generates routes with stops linked to requests.



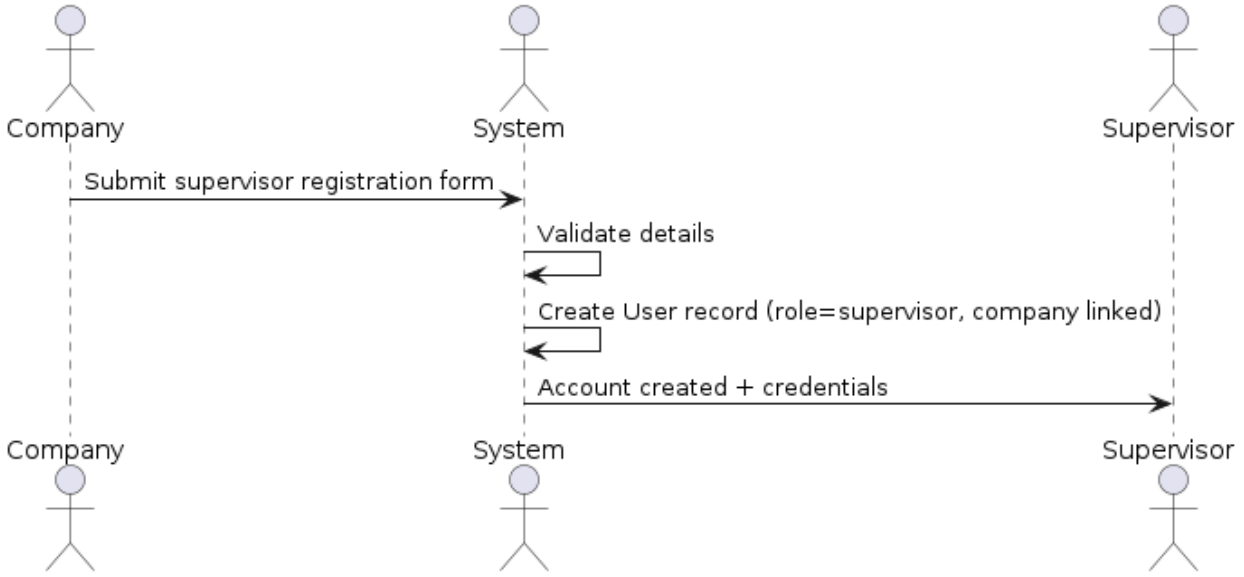
- **Assign Collectors to Routes** → Company assigns collectors to specific routes.



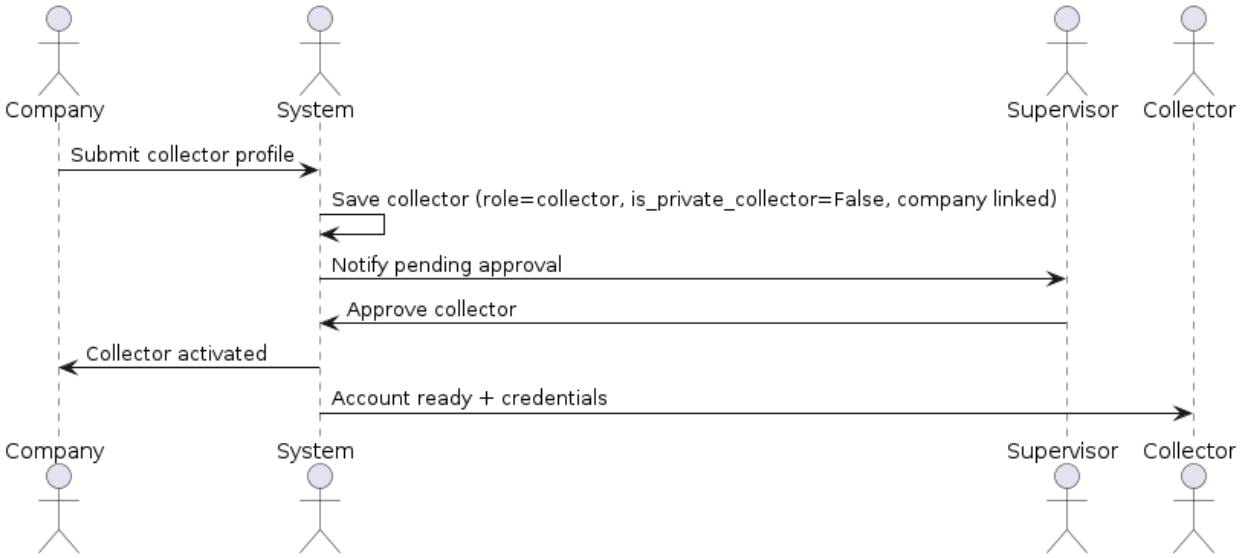
- **Handle Escalations** → Supervisor assigns requests manually if auto-assignment fails.

4. Company Workflows

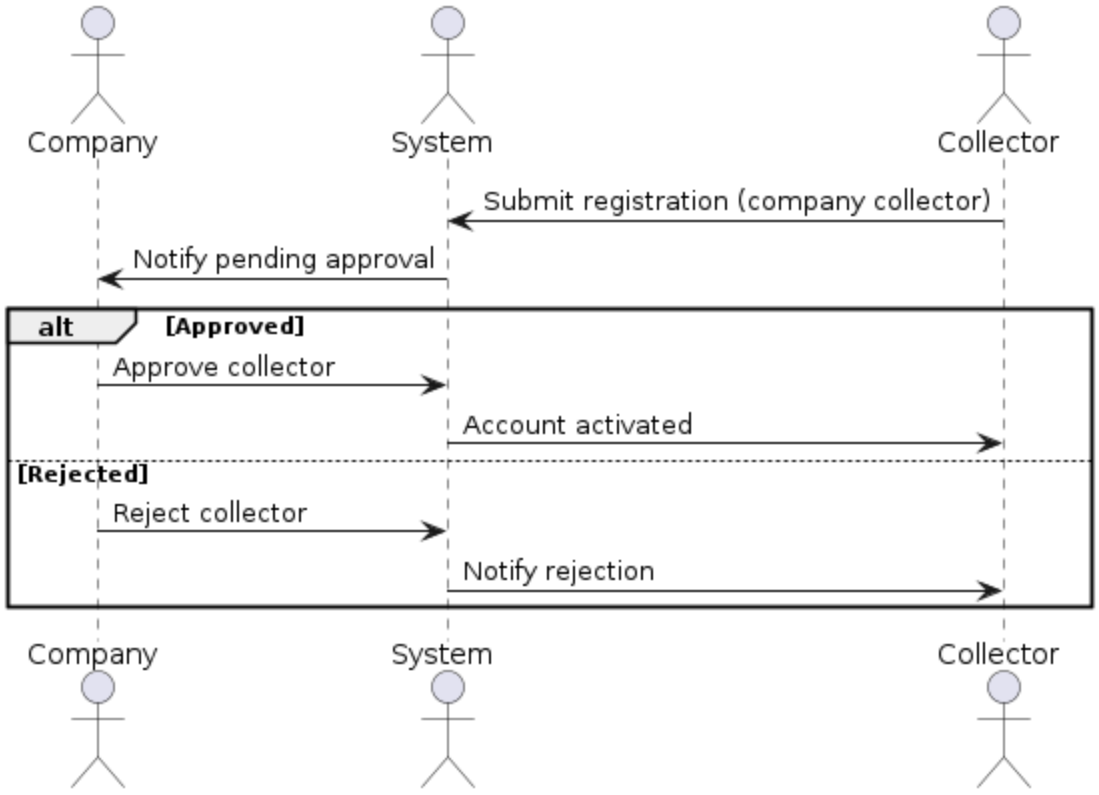
- **Create Supervisor** → Company registers supervisors.



- **Create Company Collector** → Company creates collector accounts.

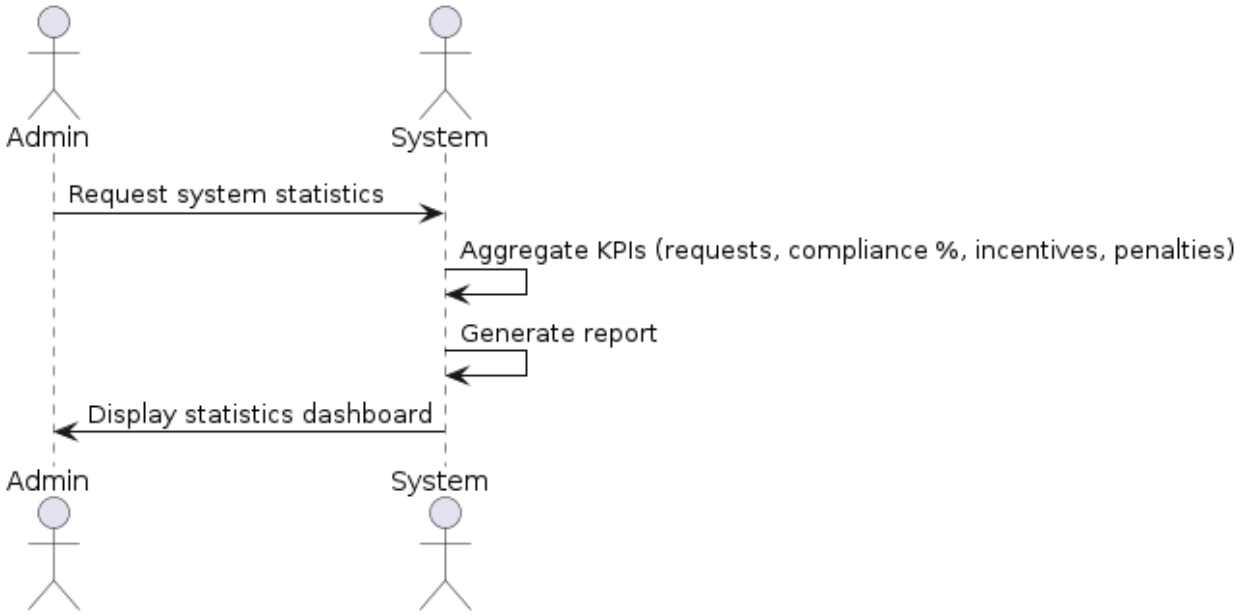


- **Approve Collector Registration** → Company validates pending collector accounts.



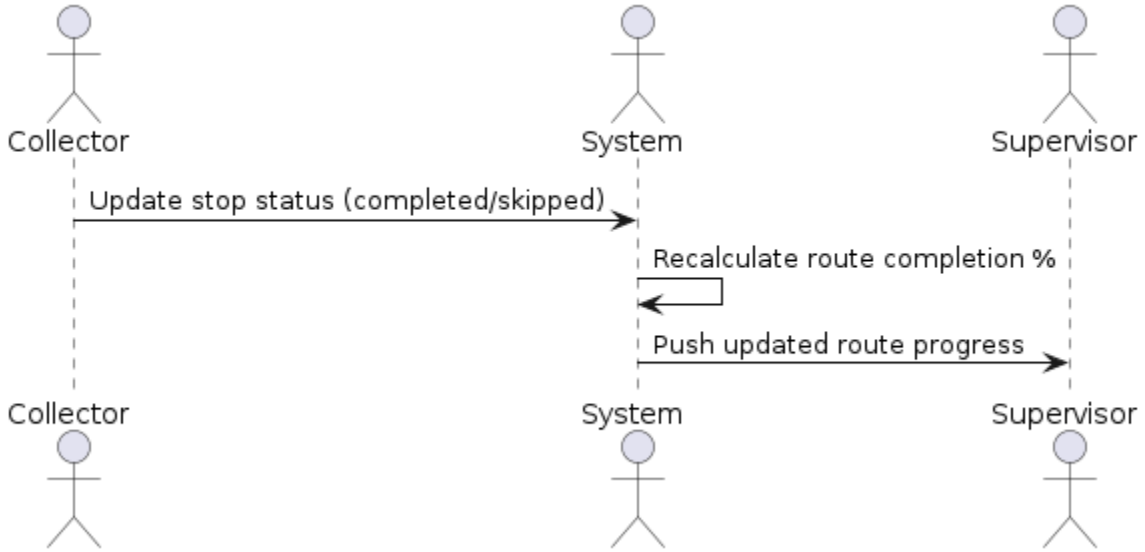
5. Admin Workflows

- **View System Statistics** → Admin monitors system-wide KPIs and operational metrics.

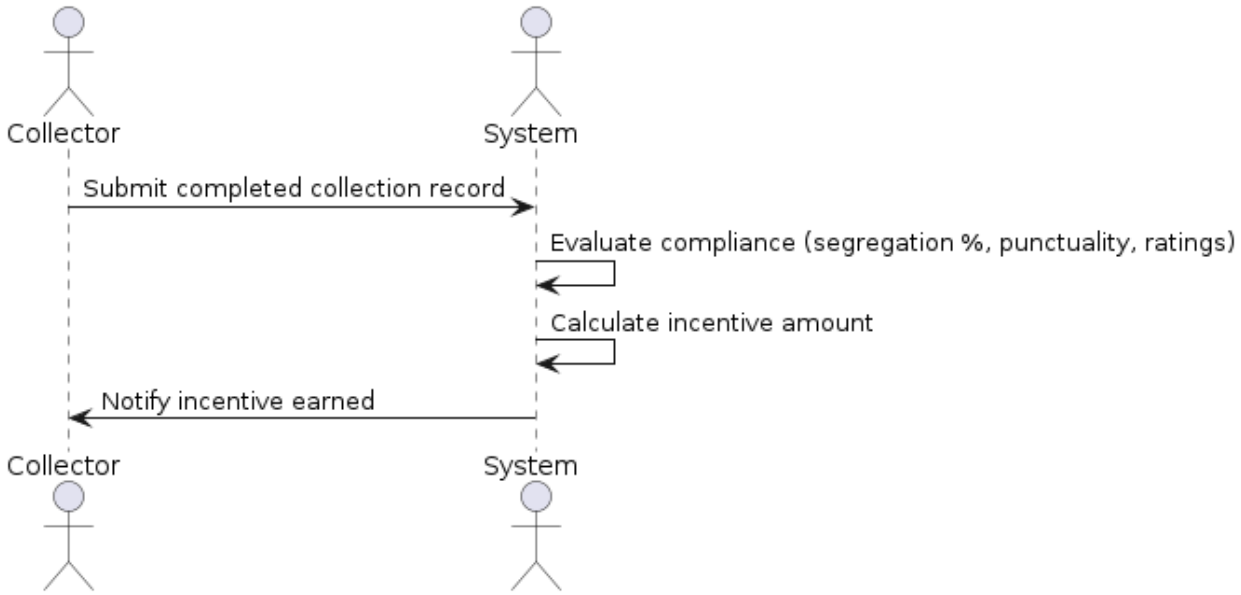


6. **System Workflows (Automated)**

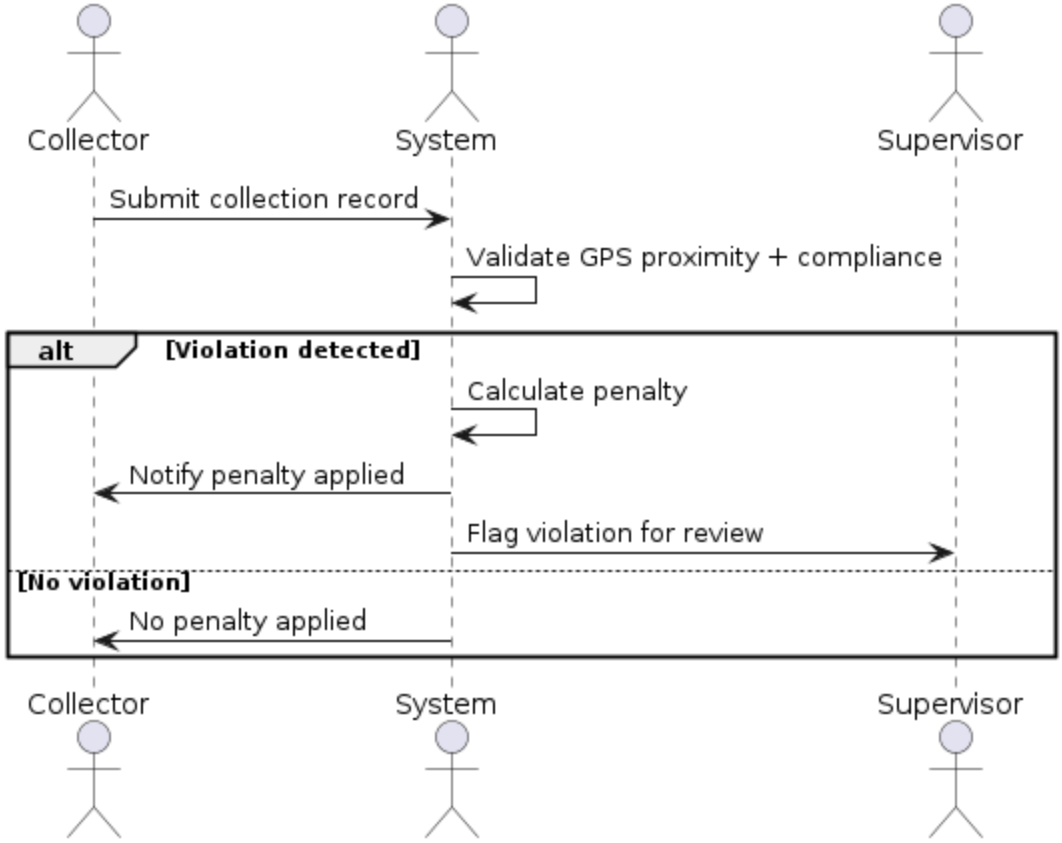
- **Route Completion Update** → System recalculates completion % based on stop statuses.



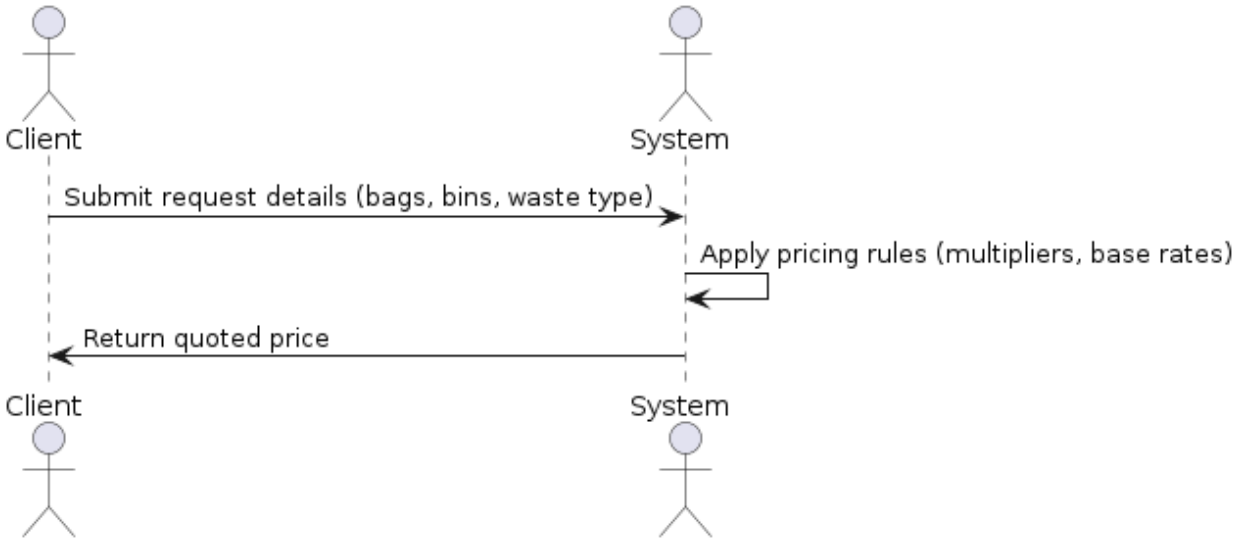
- **Incentive Calculation** → System rewards collectors based on compliance, punctuality, and ratings.



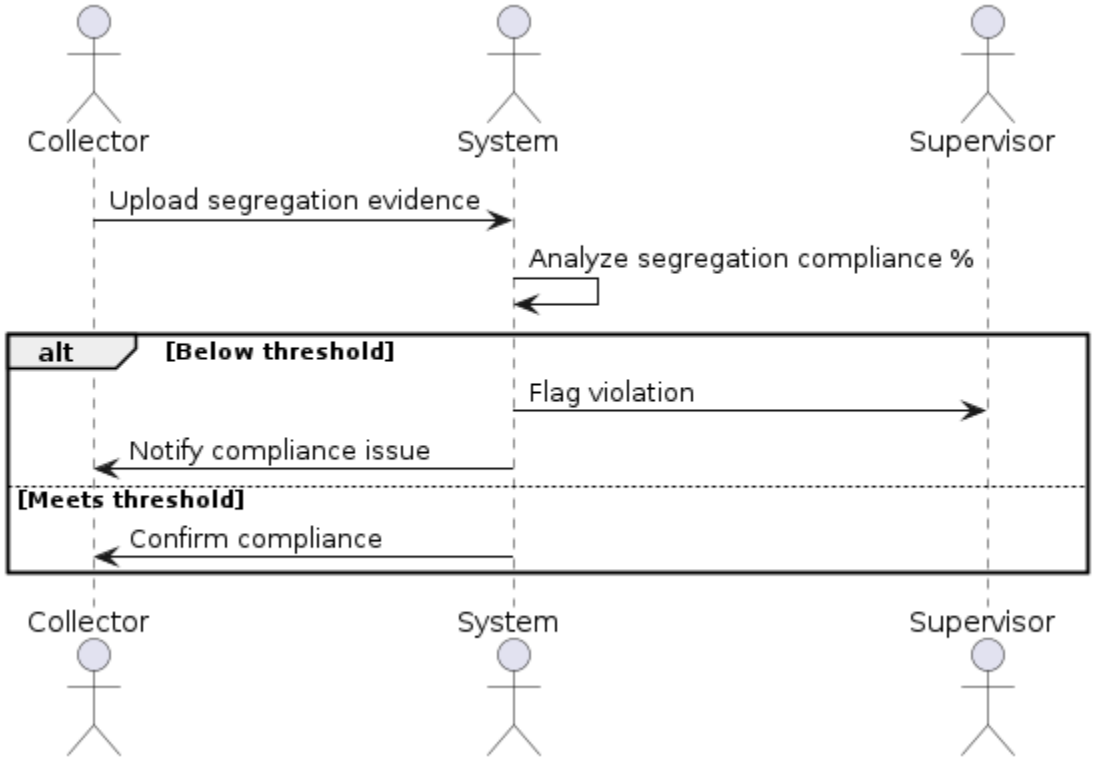
- **Penalty Calculation** → System penalizes collectors for missed stops, low compliance, or GPS mismatch.



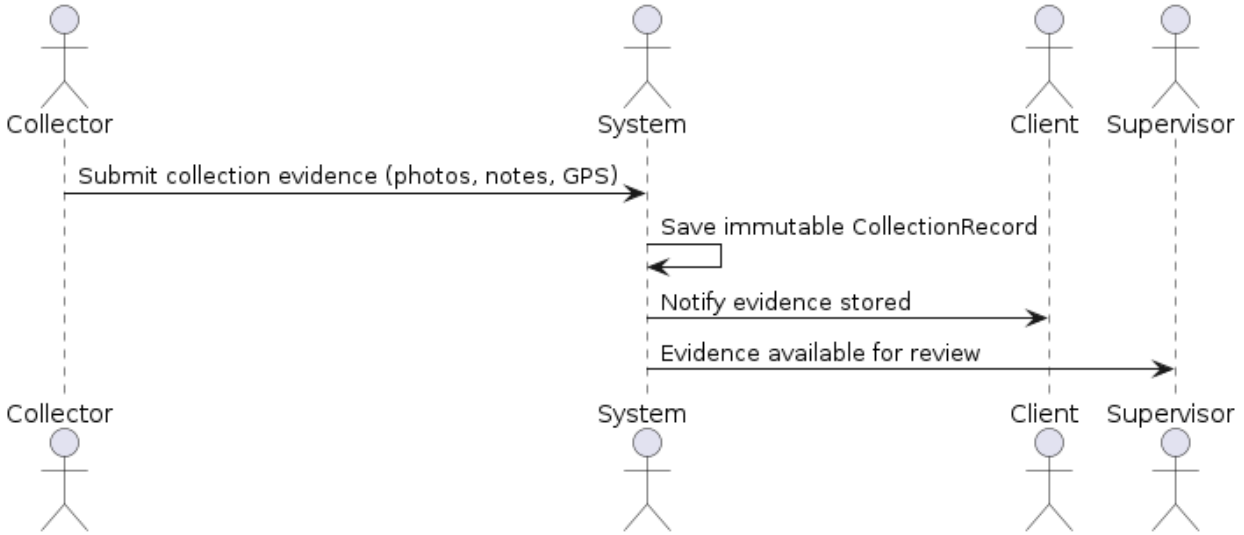
- **Pricing Calculation** → System computes quoted price based on bag/bin size and waste type multipliers.



- **Compliance Validation** → System checks segregation compliance and flags violations.



- **Audit Logging** → System records immutable collection evidence in CollectionRecord.



Future Work / Not Yet Achieved

This section outlines features and improvements that are **planned but not yet implemented**. These items represent potential extensions of the system beyond the current project scope and are intended to guide future development.

Features Pending Implementation

- **Dynamic Pricing:** The current system applies fixed pricing multipliers. A future enhancement would introduce demand-based pricing that adjusts costs dynamically based on factors such as time, location, and request volume.
- **Real-Time Incentive Calculation:** Collector incentives are currently calculated in batch processes. A future goal is to compute incentives immediately after each completed collection.
- **Advanced Route Optimization:** Route planning is currently zone-based. Future improvements may include AI-driven optimization across multiple zones or cities to improve efficiency.
- **Collector Ratings and Feedback Integration:** Collector ratings are stored but not yet used in assignment logic. Future versions could incorporate ratings into automated assignment decisions.

- **Segregation Compliance Automation:** Waste segregation evidence is manually reviewed. A future enhancement would involve automated image analysis to assist with compliance scoring.

Technical Enhancements

- **Scalability Improvements:** The current architecture is designed to support approximately 10,000 requests per day per zone. Future iterations may target significantly higher volumes using distributed or microservice-based architectures.
- **Hybrid Technology Stack:** While Django currently handles all business logic, future versions may introduce Go-based microservices for high-frequency operations such as real-time GPS tracking.
- **Offline Support:** Collectors currently require continuous connectivity. A future improvement would allow offline data capture with synchronization once connectivity is restored.

Operational Improvements

- **Supervisor Dashboards:** Current dashboards provide basic performance indicators. Future enhancements may include predictive analytics for missed stops, compliance trends, and workload forecasting.
- **Company Insights:** Companies can currently view collector lists. Future versions may provide aggregated company-wide analytics and performance reports.
- **Client Experience Enhancements:** While a wallet system exists, future improvements may include loyalty programs, subscription tiers, and push notifications.