Faculty of Engineering

Department of Informatics Engineering

Software and Information System Engineering

الجامعة السورية الخاصة
SYRIAN PRIVATE UNIVERSITY

Route Optimization for Garbage Collection in Damascus City

(Emata)

A senior 1 project report - submitted to complete the requirements for obtaining a bachelor's degree in informatics engineering - Software Engineering and Information Systems

Prepared by

Mohammad Al-Balkhi | Alaa Sheakh Al-shabab

Supervised by

Eng. Anas Abdulaziz

October 2025

SUPERVISION CERTIFICATION

Supervisor Certification

I Certify that the preparation of this project entitled [**Route Optimization for Garbage Collection in Damascus City**]
Prepared by [**Mohammad Al-Balkhi & Alaa Sheakh Al-shabab**] was made under my supervision at Faculty of Informatics Engineering in partial Fulfillment of the Requirements for the Degree of Bachelor of Software and Information system Engineering.

Name: ……………………………………….

Signature: …………………………………..

ABSTRACT

In the realm of modern urban infrastructure, efficient waste management is a critical factor for sustainable city development. Rapid urbanization and population growth have introduced complex logistical challenges, making traditional garbage collection methods increasingly inadequate. One of the primary struggles is the reliance on static schedules and manual route planning, which often results in inefficient paths, excessive fuel consumption, and uneven service coverage.

In response to this challenge, this report presents a specialized Route Optimization System designed for garbage collection in Damascus City. The system leverages advanced Vehicle Routing Problem (VRP) algorithms and geospatial data to automate the generation of optimal collection tours, ensuring that resources are utilized effectively.

The platform integrates comprehensive infrastructure management—covering bins, vehicles, and landfills—and supports scenario-based planning, allowing planners to adapt to specific operational constraints. By delivering precise daily schedules and visualizing performance metrics on interactive maps, the platform significantly reduces operational costs and environmental impact, enhancing decision-making within municipal waste management teams.

ملخص

في مشهد البنية التحتية الحضرية الحديثة، تُعد الإدارة الفعالة للنفايات عاملاً حاسماً لتنمية المدن المستدامة. لقد فرض التوسع العمراني السريع تحديات لوجستية معقدة، مما جعل أساليب جمع القمامة التقليدية غير كافية بشكل متزايد. وتتمثل إحدى العقبات الرئيسية في الاعتماد على الجداول الزمنية الثابتة والتخطيط اليدوي للمسارات، مما يؤدي غالباً إلى مسارات غير فعالة، واستهلاك مفرط للوقود، وتغطية خدمية غير متوازنة.

استجابةً لهذا التحدي، يقدم هذا التقرير نظاماً متخصصاً لتحسين المسارات مصمماً لجمع النفايات في مدينة دمشق. يوظف النظام خوارزميات توجيه المركبات (VRP) المتقدمة والبيانات الجغرافية المكانية لأتمتة إنشاء جولات الجمع المثلى، مما يضمن استخدام الموارد بفعالية.

تدمج المنصة إدارة شاملة للبنية التحتية - تشمل الحاويات، والمركبات، والمكبات - وتدعم التخطيط القائم على السيناريوهات، مما يسمح للمخططين بالتكيف مع القيود التشغيلية المحددة. من خلال تقديم جداول يومية دقيقة وتصور مؤشرات الأداء على خرائط تفاعلية، تقلل المنصة بشكل كبير من التكاليف التشغيلية والأثر البيئي، مما يعزز عملية اتخاذ القرار داخل فرق إدارة النفايات البلدية.

TABLE OF CONTENT

List of tables

List of Figures

List of abbreviations

Abbreviation	Definition
UML	Unified Modeling Language
OTP	One Time Password
SQL	Structured Query Language.
SOW	Statement of Work
RTM	Requirements traceability matrix
JWT	JSON Web Token
DRF	Django REST Framework
ERD	Entity Relationship Diagram
VRP	Vehicle Routing Problem
CVRP	Capacitated Vehicle Routing Problem
OSRM	Open-Source Routing Machine

Table 1 Table of Abbreviations

Chapter 1 Introduction

1. Introduction

In this chapter, we will introduce our project, discussing the main issues and reasons for building this system. We will also explain the objectives and goals we aim to accomplish with this system. Finally, we will provide an overview of the report organization

2. Problem Definition:

Waste management authorities in Damascus currently rely heavily on static scheduling and manual route planning to execute daily garbage collection operations. However, managing this logistical network is increasingly challenging due to rapid urbanization, expanding residential areas, and the dynamic nature of the city's road infrastructure. Manually assigning drivers to districts, estimating fuel requirements, and balancing workload is time-consuming, error-prone, and rigid, leading to an inability to adapt to daily changes such as road closures or vehicle breakdowns.

Additionally, municipalities often struggle with operational inefficiency and a lack of data visibility. Without dynamic routing, collection vehicles frequently traverse overlapping paths or travel longer distances than necessary, resulting in excessive fuel consumption and increased CO2 emissions. Critical information—such as the precise location of bins, their capacities, and the actual routes taken by drivers—remains unrecorded or scattered, making it difficult for planners to track performance, identify service gaps, or justify budget expenditures.

There is a clear need for a computational solution that can automate the planning process using Vehicle Routing Problem (VRP) algorithms and geospatial data. The municipality can benefit from a centralized, intelligent system that digitizes infrastructure management, optimizes collection routes to minimize distance and cost, and ensures that daily schedules are data-driven and environmentally sustainable.

3. Project objectives:

The primary objective of this project is to develop a comprehensive **Route Optimization and Waste Management System** for the Damascus Municipality. The system will not only automate the generation of optimal collection routes using advanced algorithms but also provide end-to-end infrastructure and fleet management features, allowing municipal teams to efficiently track bins, vehicles, and landfills while facilitating daily collection operations.

To achieve this, the system will:

- **Support infrastructure and resource management**, enabling the municipality to manage fleet vehicles, bin locations, disposal sites, and workforce distribution efficiently.
- **Facilitate and optimize daily collection scenarios**, allowing planners to define specific constraints, schedule pickups, and visualize assigned routes on interactive maps seamlessly.
- **Use VRP algorithms and geospatial data** to process location inputs, minimize travel distances, and calculate optimal paths for reduced fuel consumption and operational time.

Ultimately, this system will optimize resource utilization, reduce environmental impact, and empower waste management authorities to make well-informed, data driven decisions by managing their fleets and collection strategies in a structured and intelligent manner.

4. Report organization:

The report will be organized in the following chapters:

Chapter 1: introduction.

Chapter 2: Theoretical Background & Literature

Chapter 3: Project management and initialization

Chapter 4: System Analysis and Requirements.

Chapter 5: System Design

Chapter 6: Implementation

Chapter 7: Testing

Chapter 8: Conclusion.

Chapter 2 Theoretical Background & Literature

1. Introduction

Before delving into the system implementation, it is essential to establish the scientific and mathematical foundations upon which the Route Optimization System is built. This chapter explores the Vehicle Routing Problem (VRP) as a mathematical challenge, explains the complexity of solving it, and details the modern technologies chosen to tackle it—specifically Google OR-Tools and OSRM. Furthermore, it reviews the transition from traditional waste management methods to dynamic, algorithmic approaches.

2. The Vehicle Routing Problem (VRP)

The Vehicle Routing Problem (VRP) is one of the most studied combinatorial optimization problems in operations research. It generalizes the well-known Traveling Salesman Problem (TSP). In a classic VRP, the goal is to design a set of optimal routes for a fleet of vehicles based at a central depot to serve a set of customers (in our case, waste bins).

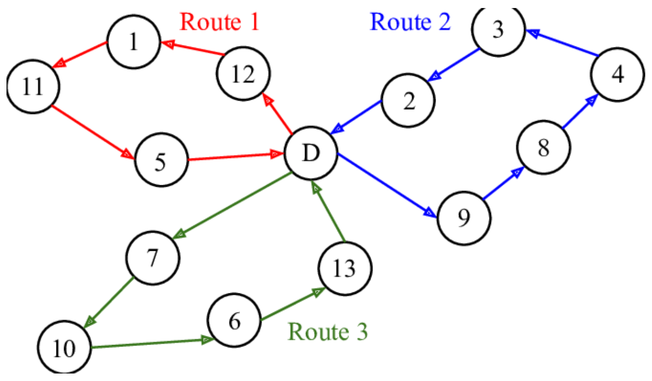


Figure 1 Illustration of the Vehicle Routing Problem (VRP)

2.1 Mathematical Complexity (NP-Hardness)

The VRP is classified as **NP-Hard (Non-deterministic Polynomial-time Hard)**. This implies that as the number of bins increases, the computational time required to find the *perfect* mathematical solution grows exponentially. For example, finding the absolute best route for 10 bins is instant, but for 100 or 1000 bins, checking every possible combination would take years even on supercomputers. Therefore, practical engineering solutions rely on **Heuristics** and **Metaheuristics** to find a "near-optimal" solution in a reasonable time.

2.2 Capacitated VRP (CVRP)

Our project specifically addresses a variant known as the **Capacitated Vehicle Routing Problem (CVRP)**. In CVRP, vehicles have a limited carrying capacity (weight or volume). The constraints are:

1. Every bin must be visited exactly once.
2. All vehicles start and end at the Depot.
3. The Constraint: The total amount of waste collected by a vehicle in one trip must not exceed its maximum capacity.

3. Optimization Technologies

To solve the CVRP efficiently within the system, we integrated two powerful technologies:

3.1 Google OR-Tools

Google OR-Tools is an open-source software suite for combinatorial optimization. Unlike writing a genetic algorithm from scratch (which can be error-prone and slow), OR-Tools uses highly optimized **Constraint Programming (CP)** solvers. In our system, we configure the solver to use **Guided Local Search (GLS)**. GLS is a metaheuristic that helps the solver escape "local optima" (good solutions that are not the best) by penalizing features of the solution that keep the search trapped, allowing it to explore better routing possibilities.

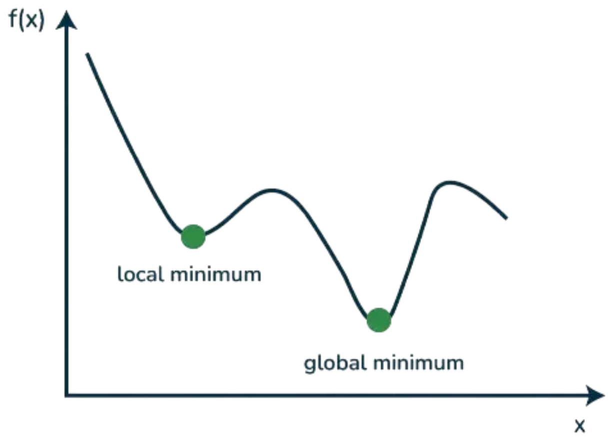


Figure 2 Local Optima vs. Global Optima in Search Algorithms

3.2 Open-Source Routing Machine (OSRM)

A critical flaw in many theoretical VRP studies is the use of **Euclidean Distance** (straight lines) to calculate costs. In a city like Damascus, a straight line does not account for buildings, one-way streets, or U-turns. **OSRM** solves this by providing:

- **Real-world Distance Matrix:** It calculates the travel cost based on the actual road network extracted from OpenStreetMap (OSM).
- **Geometry Decoding:** It generates the exact polyline of the path, ensuring the visual representation on the map matches the actual roads the driver must take.

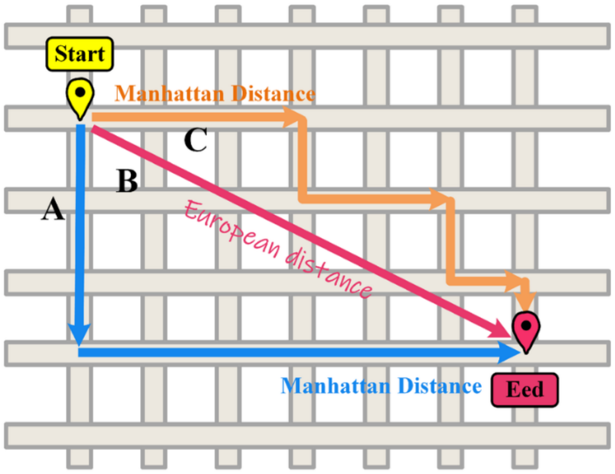


Figure 3 Euclidean Distance (Straight Line) vs. Network Distance (Real Road)

4. Literature Review: Static vs. Dynamic Routing

Waste management has traditionally relied on **Static Routing**, where trucks follow the same path every day regardless of the actual waste levels.

- Drawbacks of Static Routing: High fuel consumption, wear and tear on vehicles, and wasted time visiting empty bins.
- The Shift to Dynamic Routing: Modern research advocates for Dynamic Routing, where routes are generated daily based on demand. Our system implements this by allowing Planners to select only "Active Bins" (bins that need collection) for the optimization process, thereby skipping empty bins and reducing the total distance traveled.

5. Related Work and Comparative Analysis

Recent studies have explored various approaches to solving the Waste Collection Vehicle Routing Problem (WCVRP).

- **IoT-Based Approaches:** A study by *Mishra & Ray (2020)* proposed an IoT-based cyber-physical system utilizing sensors in bins to perform real-time routing. While highly dynamic, this approach requires expensive hardware infrastructure. In contrast, our system focuses on a cost-effective **software-centric solution** using **OSRM** and **Google OR-Tools**, making it more adaptable for cities with limited IoT infrastructure like Damascus.

- **Algorithm Comparison:** Research by *Yildiz et al. (2024)* compared Genetic Algorithms (GA) and Ant Colony Optimization (ACO) for waste routing. They found that while GA offers good solutions, it is computationally expensive. Our project utilizes **Guided Local Search (GLS)** via Google OR-Tools, which is proven to escape local optima more efficiently than standard heuristics for the Capacitated VRP (CVRP), offering a balance between solution quality and execution time.

- **Impact of Optimization:** A systematic review by *MDPI (2024)* on waste collection optimization indicated that route optimization alone can reduce collection distances by approximately **21.5%**. This validates our project's core objective of reducing fuel consumption and operational costs through algorithmic planning rather than static scheduling.

6. Summary

This chapter provided the theoretical framework for the project. We defined the waste collection challenge as a Capacitated Vehicle Routing Problem (CVRP) and justified the use of Google OR-Tools with Guided Local Search to solve this NP-Hard problem. Additionally, we highlighted the importance of integrating OSRM to ensure the generated solutions are geographically accurate and executable in the real world.

Chapter 3 project management and initialization

1. Introduction:

In this chapter, we will explore the project management phase, a crucial element in ensuring the project's success. We will focus on scope determination, initial analysis, and system structuring, laying the foundation for a well-organized and efficient development process.

2. Project management documents:

2.1 Project charter:

A project charter is a formal document that serves as an official authorization for the start of a project. It acts as a reference point throughout the project, providing a clear understanding of the project's purpose and establishing a foundation for decision-making and project governance.

Project title	*Route Optimization for Garbage collection in Damascus City*
Project start date	*October 18, 2025*
Project finish date	
Project manager	*Eng. Anas Abdulaziz*
Project objectives	*Develop an intelligent route optimization system to assist waste management teams in Damascus in planning, organizing, and managing garbage truck routes and schedules efficiently to reduce operational costs and meet service constraints.*
Approach	*1. Precisely define project scope and objectives.* *2. Comprehensively analyze the system's operational and technical requirements.* *3. Design the system architecture and decompose its major components.* *4. Develop the first phase, focusing on core data management such as bin locations, capacities, and depots (starting points).*

5. *Test and validate first-phase functionalities to ensure accurate data entry and map integration.*
6. *Develop the second phase, incorporating route optimization algorithms using tools like Google OR-Tools and integrating them with OpenStreetMap road data.*
7. *Test the second phase to verify calculation accuracy and the efficiency of generated routes.*
8. *Document all development phases and final outcomes, while creating an interactive visualization interface and exportable operational schedules.*

Roles and responsibilities:

Name	*Role*	*Responsibility*
Eng. Anas Abdulaziz	*Supervisor*	*Highly Project management and work monitoring.*
Alaa Sheakh Al-shabab	*Engineer*	*Frontend development - documentation.*
Mohamed al-Balkhi	*Engineer*	*Backend/Frontend development - deployment- documentation.*

2.2 The SOW document:

Statement of Work is a comprehensive document that defines the scope of work for a project. It outlines the specific tasks, deliverables, timeline, and responsibilities. The SOW document provides a clear understanding of what needs to be accomplished, the project's objectives, and the criteria for success.

Project Title:	*Route Optimization for Garbage collection in Damascus City*
Project Description and Objectives:	*The project aims to develop an intelligent software system utilizing Artificial Intelligence (AI) and Operations Research (OR) to assist waste management teams in Damascus. The system optimizes garbage truck routes to minimize travel distance, fuel consumption, and CO_2 emissions while adhering to operational constraints like vehicle capacities and time windows.*
Project Scope:	*This project involves developing a full-stack web application integrating interactive maps with an optimization engine. The system will manage assets (Bins, Vehicles, Landfills), execute VRP/TSP algorithms to generate optimal routes, visualize results on city maps (OpenStreetMap), and provide an interactive dashboard for planners to test scenarios and track key performance indicators (KPIs)*
Project goals:	▪ *Develop a software system that empowers municipalities to make data-driven decisions to reduce operational costs.* ▪ *Utilize Google OR-Tools and OSRM to handle complex routing constraints and generate executable daily schedules.* ▪ *Provide high-level organization features for fleet management and asset tracking via a visual interface.*
Project Deliverables:	○ *Project plan.* ○ *Working software – web app.* ○ *Final project report.*

Technology and Tools:	• *Programming Languages: Python, JavaScript, HTML, CSS.* • *Frameworks: Django (Django REST Framework) for the backend; React (Vite) for the frontend.* • *Database: PostgreSQL.* • *AI/Algorithms & Maps: Google OR-Tools, OSRM, OpenStreetMap, Leaflet/Mapbox.*
Assumptions:	- *Continuous availability of project team members.* - *Regular supervision and feedback from the project manager.* - *Continuous delivery of working software components.*
Project Resources	*Human Resources:* *Eng. Anas Abdulaziz - Project Manager* *Alaa Sheakh Al-shabab - SE Developer* *Mohamed al Balkhi - SE Developer*
Schedules:	*Project Start Date: October 18, 2025* *First project Seminar: November 15, 2025* *second project Seminar: December 21, 2025* *Project Finish Date:*

2.3 Risk Management:

the process of identifying, evaluating, and mitigating potential risks that could impact the success of the project or the team.

Risk_ID	**Rk-01**	**Rk-02**	**Rk-03**	**Rk-04**
Risk title	**Limited Human Resources** (Small Team Size)	**Scope Ambiguity** (Unclear Scope)	**Technical Knowledge Gap** (New Tech Adoption)	**Fragmented Implementation** (Workload Separation)
Risk Description	Relying on only two members creates a critical point of failure; the absence of one member would halt project progress almost entirely.	A vague or inaccurate understanding of requirements at the outset could lead to **Scope Creep** or fundamental structural errors.	The project relies on advanced technologies (AI & VRP Algorithms) that exceed the team's foundational knowledge, posing an implementation challenge.	Working in silos on different system parts simultaneously may lead to conflicts and difficulties during the **Integration Phase**.
State	active	closed	Under Mitigation	Under Mitigation
Impact	high	medium	high	medium
Mitigation Plan	**Mutual support:** Team members commit to helping each other as soon as they finish their own tasks.	**Phase Reviews:** Conduct strict reviews at the end of each cycle to ensure deliverables align with objectives.	**R&D Allocation:** Dedicate specific schedule slots for self-learning and prototyping new technologies.	**Integration Lead:** Assign one member to oversee code unification and ensure component interoperability.

Table 2 Risk Management

3. System initial analysis and design:

This section presents the initial analysis and key features identified during the early stages of project planning and development.

3.1 High-level analysis:

Use Case Diagram: use-case diagrams model the behavior of a system and help to capture the requirements of the system.

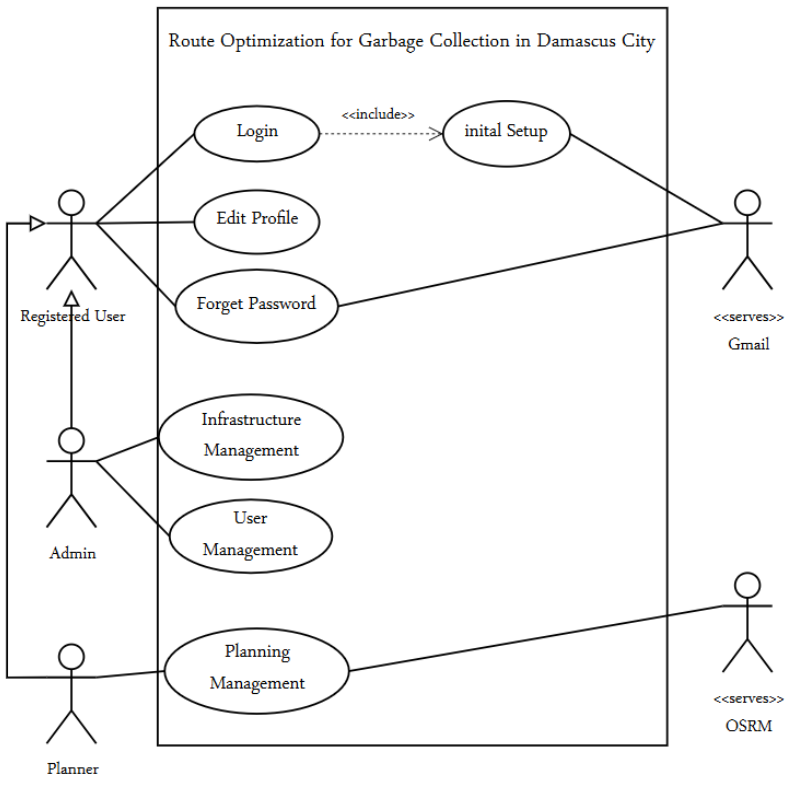


Figure 4 High-Level System Use Case Diagram

High-level Requirements:

Req_ID	Requirement Title	category	description
RE-FR-AM-01	Account & Profile Management	Account Management	Secure login, account setup, password recovery, and profile editing.
RE-FR-AM-02	User Administration	System Administration	Full user lifecycle management with search and filtering.
RE-FR-IM-03	Infrastructure Management	Infrastructure Management	CRUD operations for all assets with activate/deactivate bins only.
RE-FR-PM-04	Scenario Planning	Planning Management	Manage planning scenarios with search and filtering.
RE-FR-05	Routing & Optimization	Optimization Service	Generate collection plans and optimal routes based on scenarios.
RE-FR-06	Visualization & Monitoring	Visualization	Interactive map for assets and routes with filters.
RE-FR-07	Activity Logging	System Security	Log critical user actions for auditing and security.
RE-NFR-08	Role-Based Access Control (RBAC)	Security & Access	Enforce access control based on user roles.
RE-NFR-09	Data Integrity & Encryption	Security	Hash sensitive data and secure APIs using JWT.
RE-NFR-10	High Performance Routing	System Performance	Efficient route generation within acceptable time.
RE-NFR-11	Scalable Architecture	System Architecture	Decoupled frontend and backend architecture for scalability.
RE-NFR-12	Data Handling & Efficiency	Performance	Ensure efficient data loading and display via server-side pagination for all large datasets (Users, Infrastructure, Plans).

Table 3 High-Level Project Requirements

3.2 High-level system decomposition:

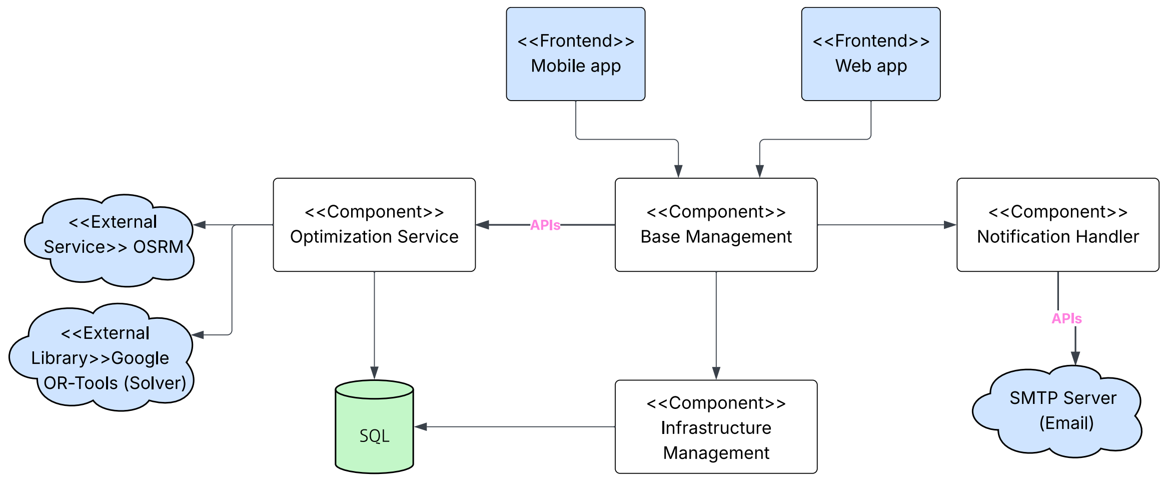

<<Frontend>> Mobile app
<<Frontend>> Web app
<<External Service>> OSRM
<<Component>> Optimization Service
APIs
<<Component>> Base Management
<<Component>> Notification Handler
<<External Library>>Google OR-Tools (Solver)
APIs
SQL
<<Component>> Infrastructure Management
SMTP Server (Email)

Figure 5 Initial System Decomposition

This High-Level System Decomposition diagram provides an overview of the system's architecture – **Modular Architecture** – illustrating its main components and their interactions.

1. Frontend: The system consists of two frontend interfaces:

- **Mobile Interface:** (for drivers accessing daily schedules and route maps via mobile devices).
- **Web Dashboard:** (for System Admins and Planners accessing infrastructure management and scenario planning tools through web browsers).

2. Backend:

- **Base Management Component:** This serves as the core backend module, handling essential functionalities such as authentication, authorization, and user management. It communicates with multiple subsystems, acts as the gateway of the backend system, and manages the primary data storage.

- **Infrastructure Management Component:** Handles the CRUD operations for city assets (municipalities, landfills, vehicles, and bins), called by the base management component via internal service calls.
- **Optimization Service Component:** A specialized component that processes routing scenarios. It calculates distance matrices using OSRM, generates optimal routes using VRP algorithms (Google OR-Tools), and stores the solutions in the database.
- **Notification Handler Component:** Handles sending emails (such as OTPs for verification) and system notifications, called and used by the base management component.

4. Project development life cycle model:

4.1 Determine development model based on system requirements and structure:

After analyzing the system and gathering its requirements, we need to identify the best software development process for our project. Our system combines complex algorithms with interactive user interfaces, so we have selected the **Scrumban** Methodology as the best option.

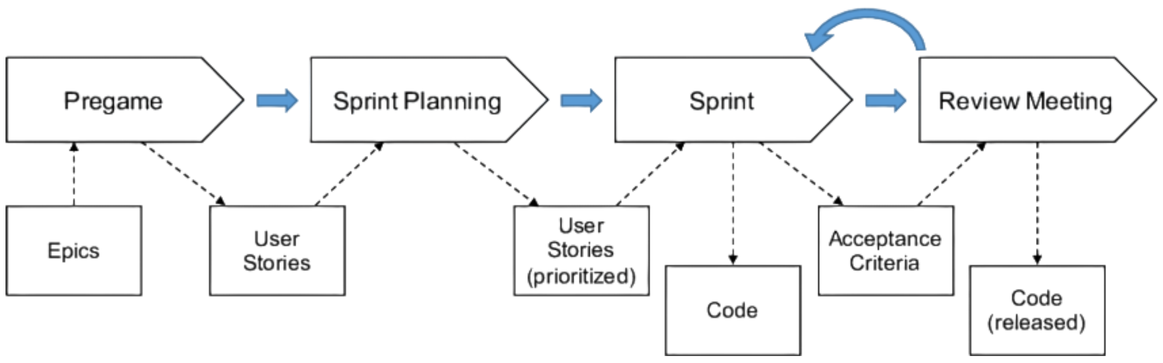


Figure 6 Scrumban Software Development Model

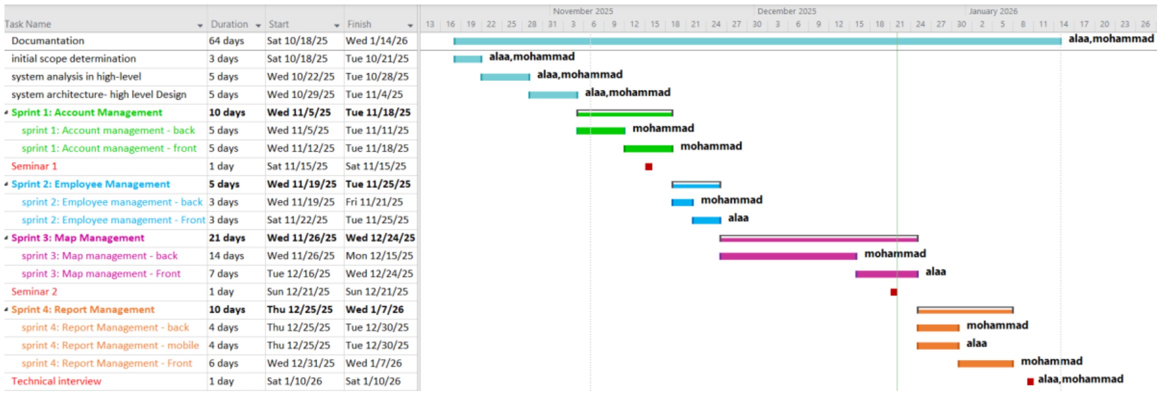
The Scrumban approach fits well with our system's separate architecture. It allows Backend and Frontend development to proceed at the same time while staying in sync. This hybrid model combines the advantages of Scrum, letting us break the project into clear development cycles—like the Infrastructure Foundation cycle followed by the Optimization Engine cycle—with the flexibility of Kanban's pull-based system. This setup ensures a steady workflow, lets us prioritize tasks dynamically, and supports the ongoing improvement needed for the VRP algorithms and map visualizations.

4.2 Scrumban project management:

Unlike traditional models where the whole project is planned in advance, Scrumban project management combines ongoing workflow with on-demand planning. The aim is to meet important business goals and provide value to the customer through a series of small, high-priority tasks instead of strict stages. The Project Charter and Statement of Work (SOW), made at the start of the project, stay flexible and change to fit updates and improvements based on current system needs and development progress.

4.3 Initial Project Plan – Gantt chart:

A document outlining tasks, deadlines, and resources needed to achieve project objectives, serving as a roadmap for successful project execution, and reflecting the system development model



Task Name	Duration	Start	Finish
Documantation	64 days	Sat 10/18/25	Wed 1/14/26
initial scope determination	3 days	Sat 10/18/25	Tue 10/21/25
system analysis in high-level	5 days	Wed 10/22/25	Tue 10/28/25
system architecture- high level Design	5 days	Wed 10/29/25	Tue 11/4/25
Sprint 1: Account Management	**10 days**	**Wed 11/5/25**	**Tue 11/18/25**
sprint 1: Account management - back	5 days	Wed 11/5/25	Tue 11/11/25
sprint 1: Account management - front	5 days	Wed 11/12/25	Tue 11/18/25
Seminar 1	1 day	Sat 11/15/25	Sat 11/15/25
Sprint 2: Employee Management	**5 days**	**Wed 11/19/25**	**Tue 11/25/25**
sprint 2: Employee management - back	3 days	Wed 11/19/25	Fri 11/21/25
sprint 2: Employee management - Front	3 days	Sat 11/22/25	Tue 11/25/25
Sprint 3: Map Management	**21 days**	**Wed 11/26/25**	**Wed 12/24/25**
sprint 3: Map management - back	14 days	Wed 11/26/25	Mon 12/15/25
sprint 3: Map management - Front	7 days	Tue 12/16/25	Wed 12/24/25
Seminar 2	1 day	Sun 12/21/25	Sun 12/21/25
Sprint 4: Report Management	**10 days**	**Thu 12/25/25**	**Wed 1/7/26**
sprint 4: Report Management - back	4 days	Thu 12/25/25	Tue 12/30/25
sprint 4: Report Management - mobile	4 days	Thu 12/25/25	Tue 12/30/25
sprint 4: Report Management - Front	6 days	Wed 12/31/25	Wed 1/7/26
Technical interview	1 day	Sat 1/10/26	Sat 1/10/26

Figure 7 Gantt Chart

5. Summary:

In conclusion, good project management is essential for successful software system development. It provides structure and makes sure projects are completed on time and within scope. Strong project management also improves the delivery of high-quality software systems that meet the project's goals.

Chapter 4 System Analysis and Requirements

1. Introduction

The Core Management Service is the system's backend API gateway, handling all client requests and controlling data flow. It manages users, assets, planning scenarios, and operational limits, forming the backbone of the system. This section documents its analysis, design, development, and overall role.

2. Service Detailed analysis:

2.1 Introduction:

In this section, we will introduce the analytical study of the service using the needed UML diagrams for system requirements modeling.

2.2 Detailed Service Requirements:

Req_ID	Requirement Title	category	description
RE-FR-AM-01	The system should provide its administrator with the ability to create accounts for users using their credentials.	Account Management	The required data is username, email, role (admin, planner, driver)
RE-FR-AM-02	The system should provide its administrator with the ability to update user accounts.	Account Management	
RE-FR-AM-03	The system should provide its administrator with the ability to view user accounts.	Account Management	
RE-FR-AM-04	The system should provide its administrator with the ability to archive user accounts.	Account Management	
RE-FR-AM-05	The system should provide its administrator with the ability	Account Management	

	to restore archived user accounts.		
RE-FR-AM-06	The system should provide its administrator with the ability to search for users by username.	Account Management	
RE-FR-AM-07	The system should provide its administrator with the ability to filter users based on specific data.	Account Management	Filter by: **Role** (admin, planner, driver), **Status** (active, inactive, archived)
RE-FR-AM-08	The system should provide users with the ability to log in using their credentials.	Account Management	The credentials are: **email address** + **password**.
RE-FR-AM-09	The system should provide users with the ability to recover their password if they forget it.	Account Management	Passwords are recovered using: an OTP code, a password, and password confirmation.
RE-FR-AM-10	The system should provide users with the ability to activate their accounts.	Account Management	Activation is done by entering an OTP code and confirming the password.
RE-FR-AM-11	The system must provide users with the ability to update their profile details.	Account Management	Information that users can update: **Username**, **phone number**, and **profile picture**.
RE-FR-AM-12	The system must provide users with the ability to change their account password.	Account Management	Ideally, users must provide the **old password** for verification, followed by the **new password and its confirmation**.
RE-FR-AM-13	The system should provide users with the ability to log out of their accounts.	Account Management	

RE-FR-AM-14	The system should provide its administrator with the ability to view user activity logs.	Account Management	The log includes data such as: time of last login/last password change, reason for password change, user role, and account status.
RE-FR-IM-15	The system should provide its administrator with the ability to **create** a new municipality.	Infrastructure Management	The required data includes the municipality name and Latitude, Longitude
RE-FR-IM-16	The system should provide its administrator with the ability to **update** existing municipality details.	Infrastructure Management	
RE-FR-IM-17	The system should provide its administrator with the ability to **view** a list of municipalities.	Infrastructure Management	
RE-FR-IM-18	The system should provide its administrator with the ability to **delete** a municipality.	Infrastructure Management	
RE-FR-IM-19	The system should provide its administrator with the ability to **register** a new landfill site.	Infrastructure Management	Required data: Name of landfill, exact GPS coordinates (latitude and longitude), and administrative areas to which it belongs.
RE-FR-IM-20	The system should provide its administrator with the ability to **update** landfill information.	Infrastructure Management	
RE-FR-IM-21	The system should provide its administrator with the ability to **view** landfill sites.	Infrastructure Management	

RE-FR-IM-22	The system should provide its administrator with the ability to **delete** a landfill site.	Infrastructure Management	
RE-FR-IM-23	The system should provide its administrator with the ability to **add** a new waste collection vehicle.	Infrastructure Management	Required data: truck name, capacity (kg/m^3), coordinates.
RE-FR-IM-24	The system should provide its administrator with the ability to **update** vehicle details.	Infrastructure Management	
RE-FR-IM-25	The system should provide its administrator with the ability to **view** the fleet of vehicles.	Infrastructure Management	
RE-FR-IM-26	The system should provide its administrator with the ability to **delete** a vehicle.	Infrastructure Management	
RE-FR-IM-27	The system should provide its administrator with the ability to **add** new waste bins.	Infrastructure Management	Required data: bin name, GPS Coordinates (Lat, Lng), Capacity, and the associated Municipality.
RE-FR-IM-28	The system should provide its administrator with the ability to **update** bin details.	Infrastructure Management	
RE-FR-IM-29	The system should provide its administrator with the ability to **view** all waste bins.	Infrastructure Management	
RE-FR-IM-30	The system should provide its administrator with the ability to **delete** a waste bin.	Infrastructure Management	
RE-FR-PM-31	The system should provide the Planner with the ability to	Planning Management	A scenario requires defining: **Scenario Name**, **Municipality**, **Collection**

	create a new collection scenario.		**Date**, **Assigned Vehicle**, and selecting a set of **Target Bins**.
RE-FR-PM-32	The system should provide the Planner with the ability to **view available bins** for a specific plan.	Planning Management	The system must filter and display only **active bins** that are **not currently assigned** to other scenarios on the same date to prevent conflict.
RE-FR-PM-33	The system should provide the Planner with the ability to **update** an existing scenario.	Planning Management	
RE-FR-PM-34	The system should provide the Planner with the ability to **view** the list of collection scenarios.	Planning Management	
RE-FR-PM-35	The system should provide the Planner with the ability to **search** for scenarios.	Planning Management	
RE-FR-PM-36	The system should provide the Planner with the ability to **filter** collection scenarios.	Planning Management	The list can be filtered by: **Status** (Active/Archived), **Municipality**, and specific **Collection Date**
RE-FR-PM-37	The system should provide the Planner with the ability to **delete** a collection scenario.	Planning Management	
RE-FR-PM-38	The system should automatically archive expired scenarios.	Planning Management	Scenarios are automatically classified as **"Archived"** once their scheduled collection date has passed (< Today), removing them from the active list.

RE-FR-PM-39	The system should provide the Planner with the ability to **generate optimal routes** (Solve).	Optimization Service	Triggering the "Solve" action calculates the shortest path visiting all selected bins based on vehicle constraints.
RE-FR-PM-40	The system should provide the Planner with the ability to **view route solutions**.	Optimization Service	
RE-FR-PM-41	The system should provide the Planner with the ability to **view planning statistics**.	Planning Management	
RE-FR-VS-42	The system should provide the Administrator with the ability to **visualize infrastructure assets** on an interactive map.	Visualization & Monitoring	
RE-FR-VS-43	The system should provide the Administrator with the ability to **filter map layers**.	Visualization & Monitoring	The admin can switch the view of assets based on the asset they want (municipality, landfill, truck, bin) to reduce map clutter.
RE-FR-VS-44	The system should provide the Planner with the ability to **visualize route solutions** on the map.	Visualization & Monitoring	The system renders the optimized **path (polyline)** and the **sequence of stops** (numbered markers) for a specific scenario solution.

Table 4 Detailed Service Requirements

2.3 Requirements Modeling:

Use case diagram:

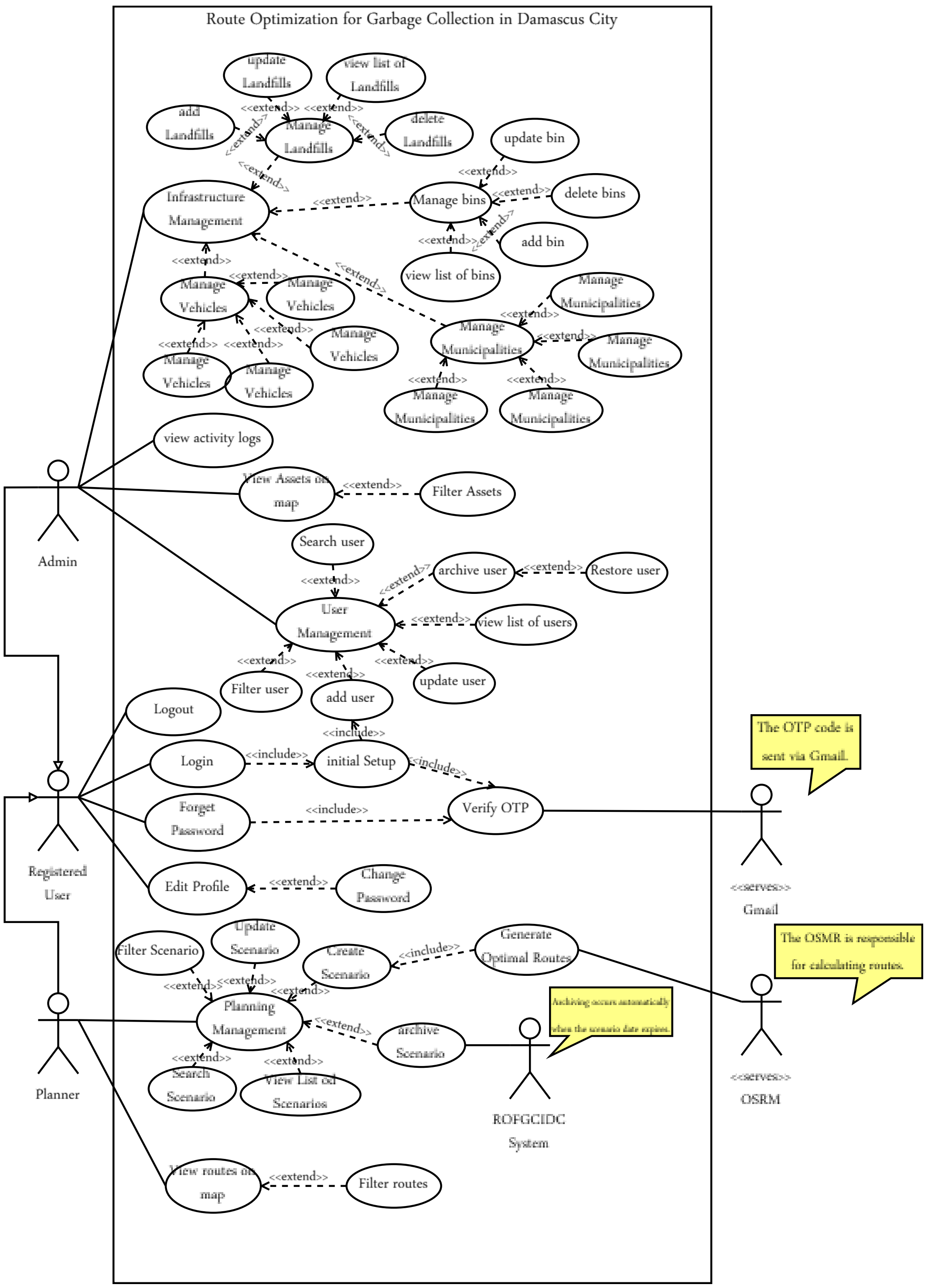


Figure 8 Use Case Diagram

2.3.1 Actor: User (Common)

Activate Account flow UC-01:

Use case name	Activate Account
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user accesses the system website on the account activation page. 2. The system prompts the user to enter the email address previously created by the administrator. 3. The user enters the email address and clicks **"Send Verification Code."** 4. The system sends the verification code (OTP) to the user's email address and displays the account activation page. 5. The user enters the verification code and password, confirms it, and then clicks **"Activate Account."** 6. The system validates the entered data, activates the account, and automatically logs the user in.
Alternative flows	**A1 – Incorrect or Unregistered Email:** • The system displays an error message. • The user re-enters the email address. • The flow returns to Step 3. **A2 – Invalid OTP Code:** • The system displays an error message. • The user requests to resend the verification code. • The flow returns to Step 4. **A3 – Weak or Invalid Password:** • The system displays password requirements. • The user re-enters the password. • The flow returns to Step 5.
Entry condition	The admin should have created an account with the registered email address.
Exit conditions	Account successfully activated.

Table 5 Activate Account - Use Case Diagram

Activity diagram:

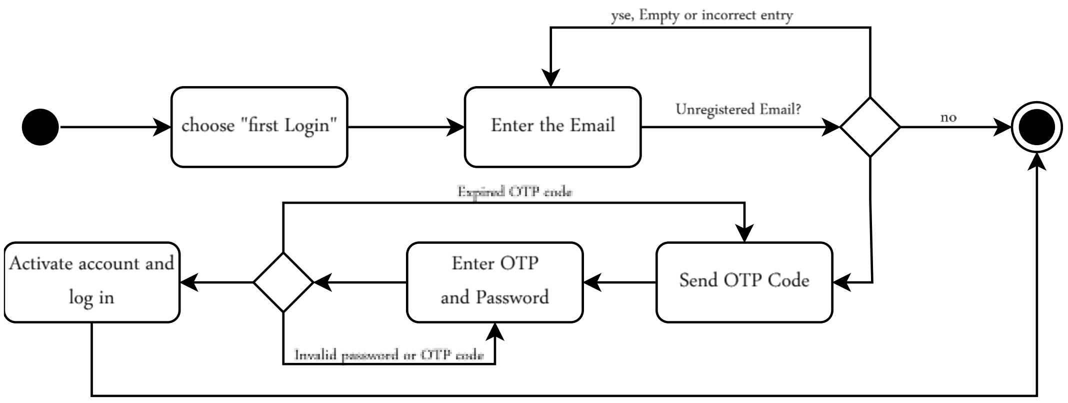


Figure 9 Activate Account - Activity Diagram

Sequence diagram:

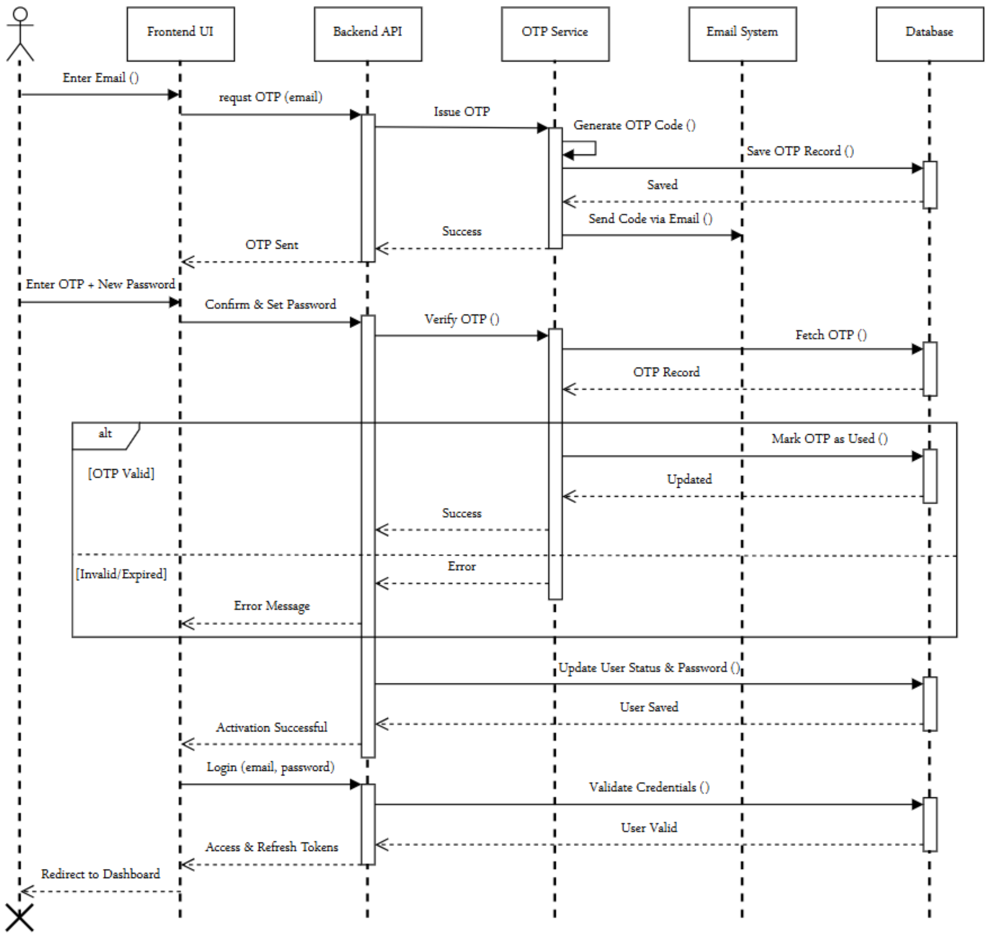


Figure 10 Activate Account - Sequence Diagram

Login flow UC-02:

Use case name	Login
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user accesses the login page. 2. The system displays the email and password fields. 3. The user enters the email address and password and clicks the **"Login"** button. 4. The system validates the entered credentials and redirects the user to the main page.
Alternative flows	**A1 – Incorrect Email or Password:** • The system displays an error message indicating that the email or password is incorrect. • The user re-enters the credentials. • The flow returns to Step 3.
Entry condition	The user has a registered and activated account.
Exit conditions	Successfully logged in to the home page.

Table 6 Login - Use Case Diagram

Activity diagram:

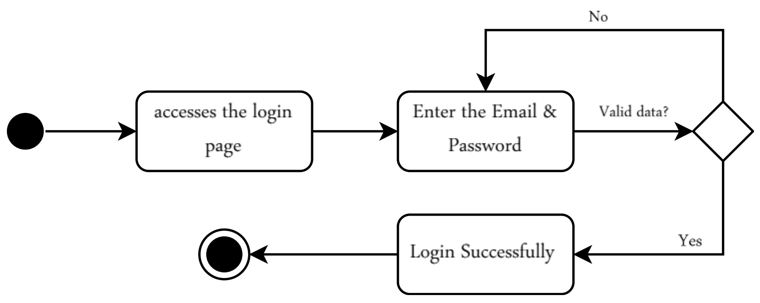


Figure 11 Login - Activity Diagram

Sequence diagram:

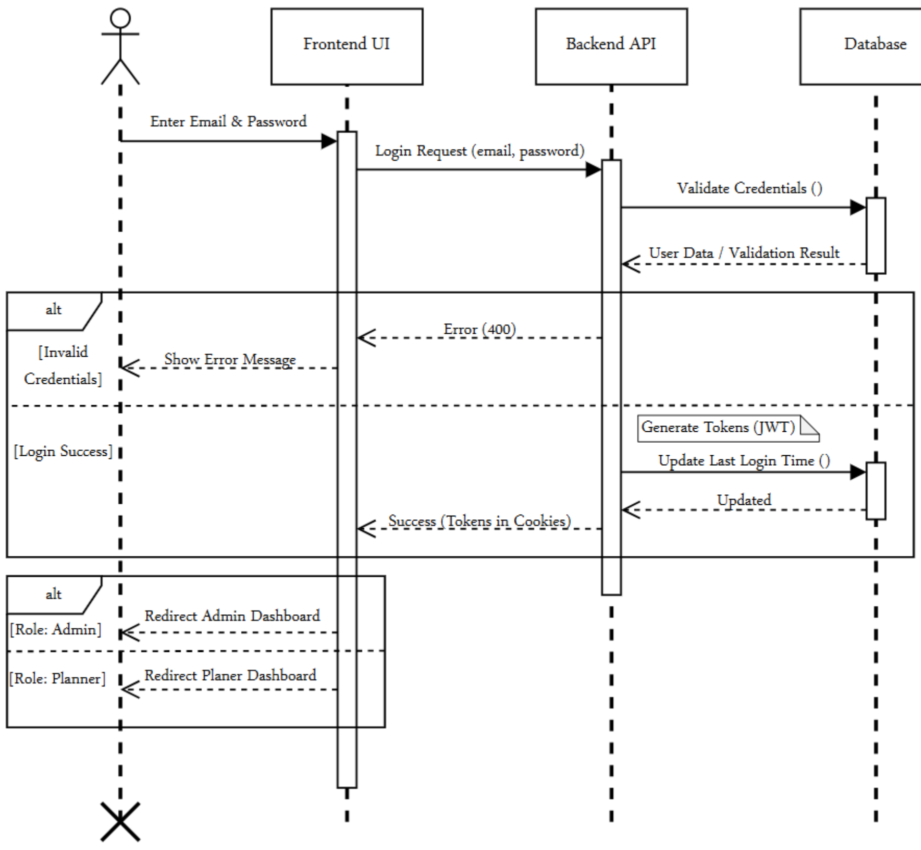


Figure 12 Login - Sequence Diagram

Forget Password flow UC-03:

Use case name	Forget Password
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user navigates to the **Forgot Password** page. 2. The system prompts the user to enter the registered email address. 3. The user enters the email address and clicks **"Send Reset Code."** 4. The system sends a password reset code (OTP) to the user's email address. 5. The user enters the received reset code and a new password, confirms it, and clicks **"Reset Password."** 6. The system validates the entered data and updates the password successfully.

Alternative flows	**A1 – Incorrect or Unregistered Email:** • The system displays an error message indicating that the email is incorrect or not registered. • The user re-enters the email address. • The flow returns to Step 3. **A2 – Invalid Reset Code (OTP):** • The system displays an error message. • The user requests to resend the reset code. • The flow returns to Step 4. **A3 – Weak Password:** • The system displays password requirements. • The user re-enters a stronger password. • The flow returns to Step 5.
Entry condition	The user has a registered account and is not logged in.
Exit conditions	The user successfully resets the password.

Table 7 Forget Password - Use Case Diagram

Activity diagram:

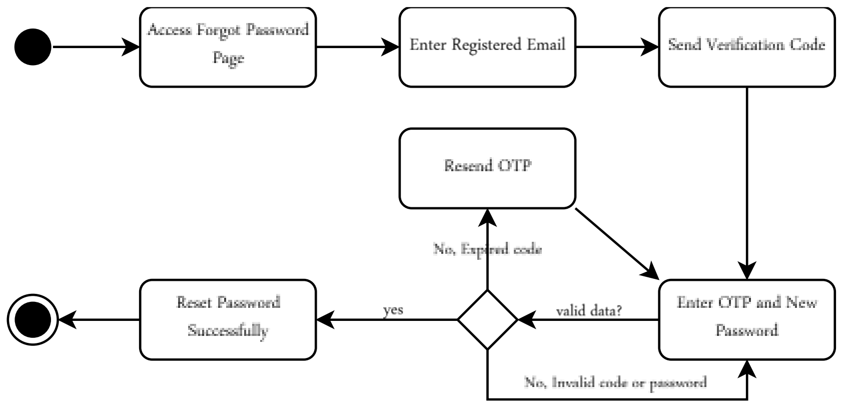

Access Forgot Password Page
Enter Registered Email
Send Verification Code
Resend OTP
No, Expired code
Reset Password Successfully
yes
valid data?
Enter OTP and New Password
No, Invalid code or password

Figure 13 Forgot Password - Activity Diagram

Sequence diagram:

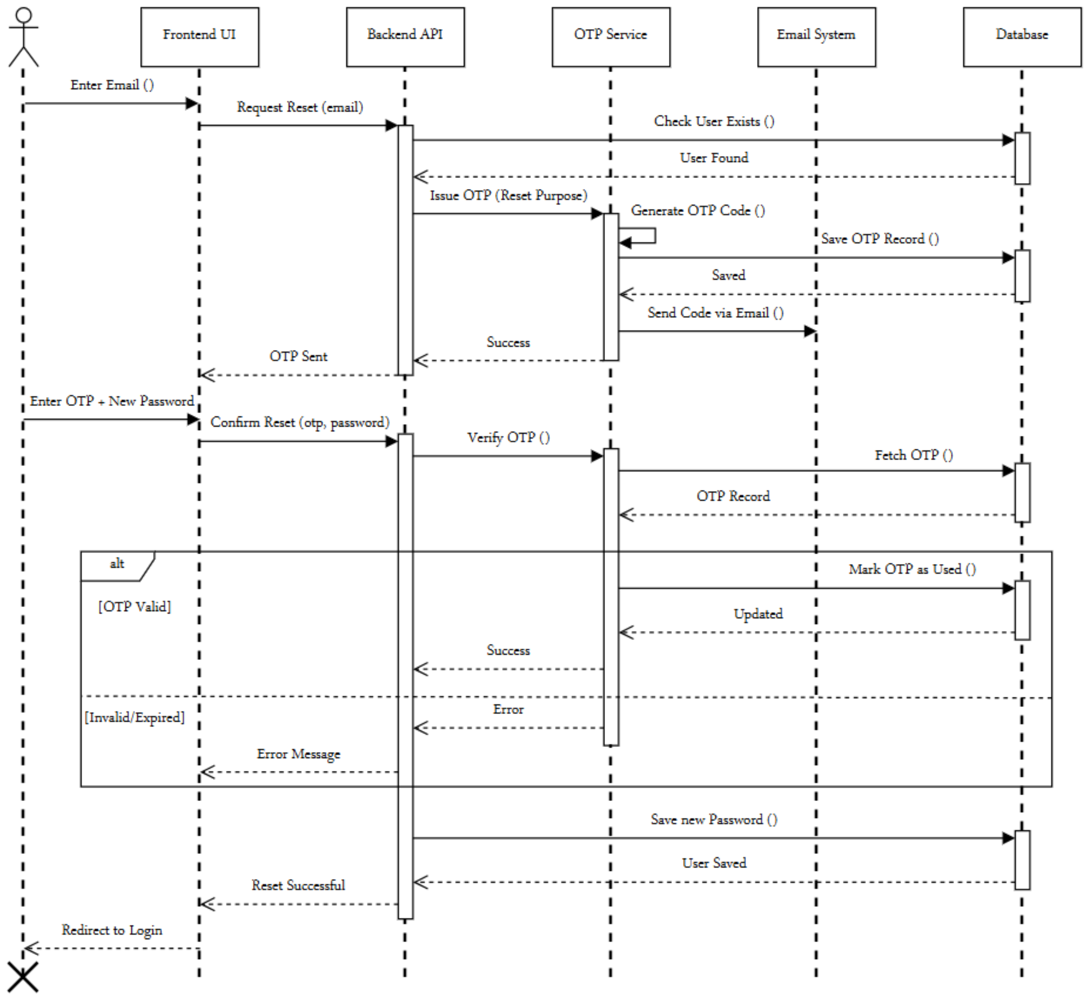


Figure 14 Forget Password - Sequence Diagram

Edit Profile flow UC-04:

Use case name	Edit Profile
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user navigates to the **Edit Profile** page from the home page. 2. The system displays the user's current profile information. 3. The user edits the profile information (profile photo, phone number, username) and clicks **"Save."** 4. The system validates the entered information and saves the changes successfully.

Alternative flows	**A1 – Invalid or Missing Data:** • The system displays an error message indicating invalid or missing fields. • The user corrects the data. • The flow returns to Step 3.
Entry condition	The user is logged in.
Exit conditions	The profile information is updated successfully.

Table 8 Edit Profile - Use Case Diagram

Activity diagram:

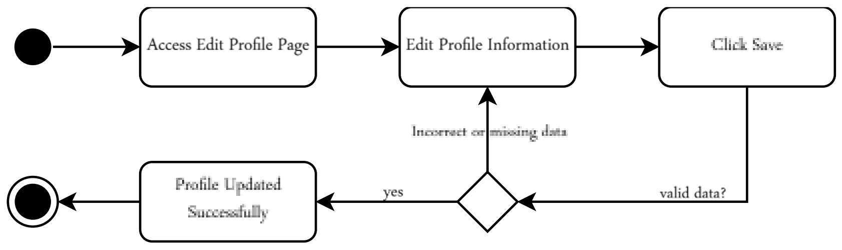


Figure 15 Edit Profile - Activity Diagram

Sequence diagram:

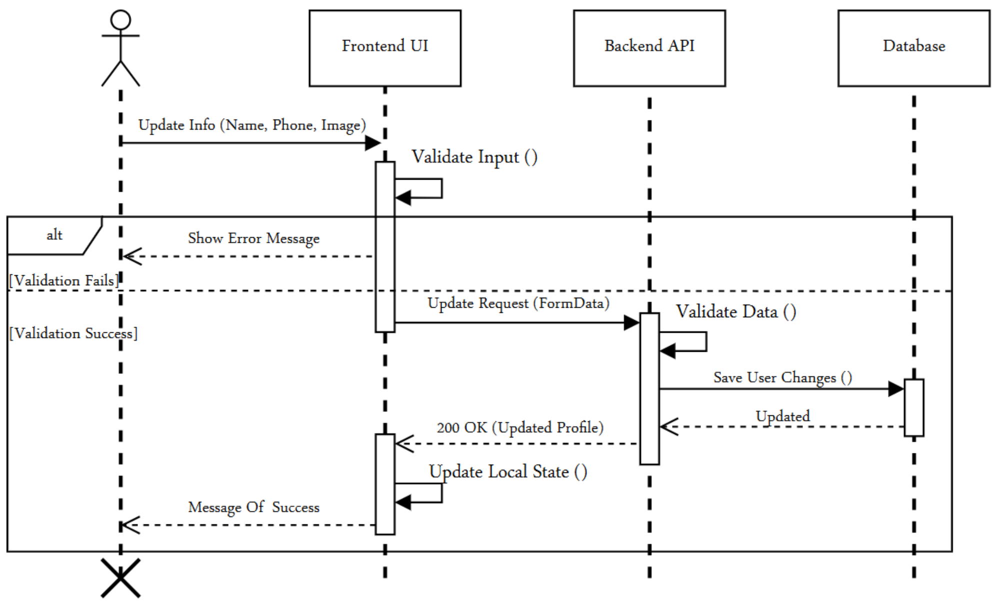


Figure 16 Edit Profile - Sequence Diagram

Change Password flow UC-05:

Use case name	Change Password
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user navigates to the **Change Password** page from account settings. 2. The system displays the current password, new password, and confirmation fields. 3. The user enters the current password, the new password, confirms it, and clicks **"Save."** 4. The system validates the entered data and updates the password successfully.
Alternative flows	**A1 – Incorrect Current Password:** • The system displays an error message. • The user re-enters the current password. • The flow returns to Step 3. **A2 – Weak or Mismatched New Password:** • The system displays password requirements or mismatch error. • The user re-enters the new password. • The flow returns to Step 3.
Entry condition	The user is logged in
Exit conditions	The password is changed successfully

Table 9 Change Password - Use Case Diagram

Activity diagram:

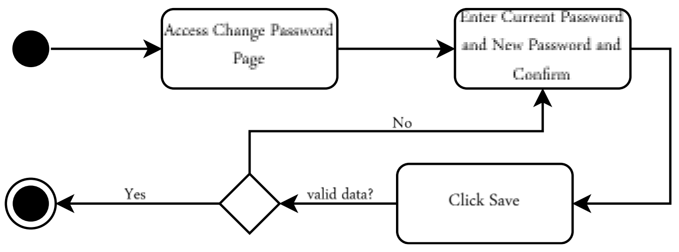

Access Change Password Page
Enter Current Password and New Password and Confirm
No
Yes
valid data?
Click Save

Figure 17 Change Password- Activity Diagram

Sequence diagram:

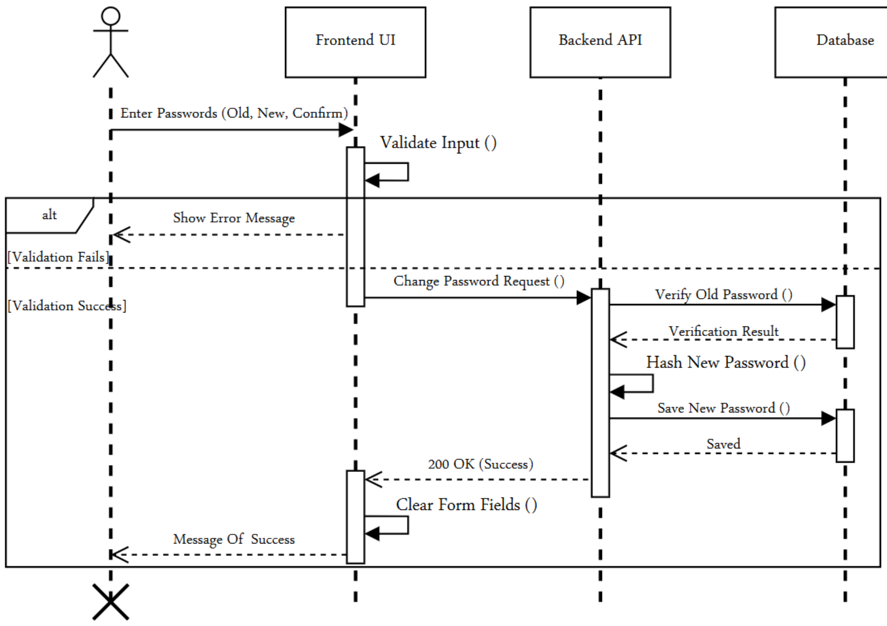


Figure 18 Change Password - Sequence Diagram

Logout flow UC-06:

Use case name	Logout
Participating Actors	Admin, Planner, Driver
Main Flow	1. The user clicks the **"Logout"** option. 2. The system terminates the user session. 3. The system redirects the user to the login page.
Entry condition	The user is logged in
Exit conditions	The user is logged out successfully

Table 10 Logout - Use Case Diagram

Activity diagram:

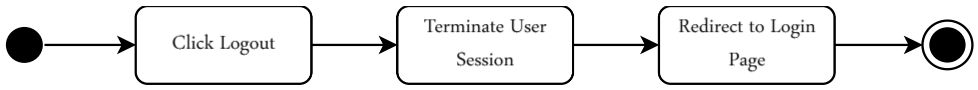


Figure 19 Logout - Activity Diagram

Sequence diagram:

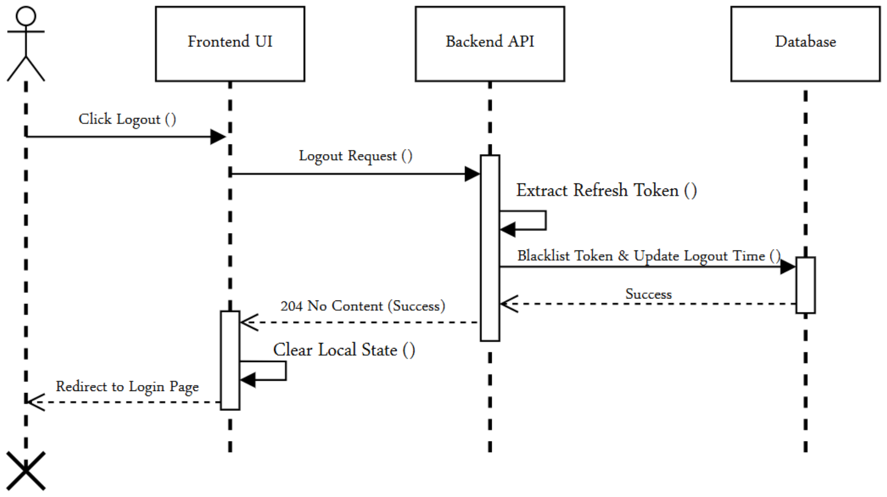


Figure 20 Logout - Sequence Diagram

2.3.2 Actor: Admin

Create User flow UC-07:

Use case name	Create User
Participating Actors	Admin
Main Flow	1. The administrator navigates to the **Users Management** page and selects **“Add New User.”** 2. The system displays a form requesting: username, email address, role, phone number, and profile picture. 3. The administrator enters the required data and clicks **“Create.”** 4. The system validates the entered data and creates a new user record. 5. The system displays a success message and updates the users list.
Alternative flows	**A1 – Validation Error:** • The system detects invalid data (e.g., incorrect email format) or missing required fields. • The system displays descriptive error messages. • The administrator corrects the data and resubmits the form. **A2 – Duplicate Data:** • The system detects that the email address or username already exists. • The system prevents the creation process and notifies the administrator. • The flow returns to Step 3.
Entry condition	The admin is logged in and has the required permissions.
Exit conditions	A new user account is created

Table 11 Create User - Use Case Diagram

Activity diagram:

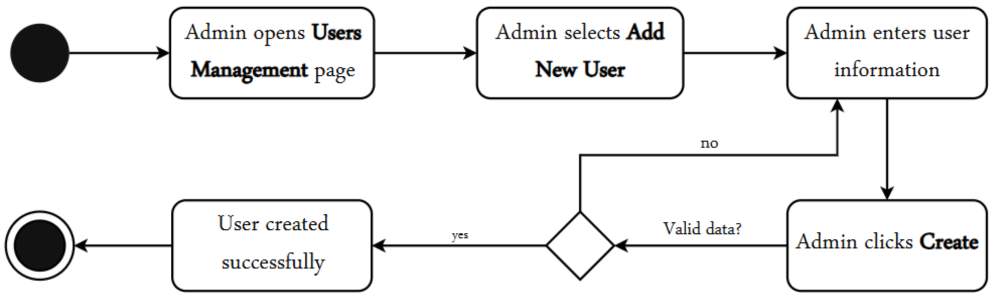

Admin opens **Users Management** page
Admin selects **Add New User**
Admin enters user information
Admin clicks **Create**
Valid data?
no
yes
User created successfully

Figure 21 Create User - Activity Diagram

Sequence diagram:

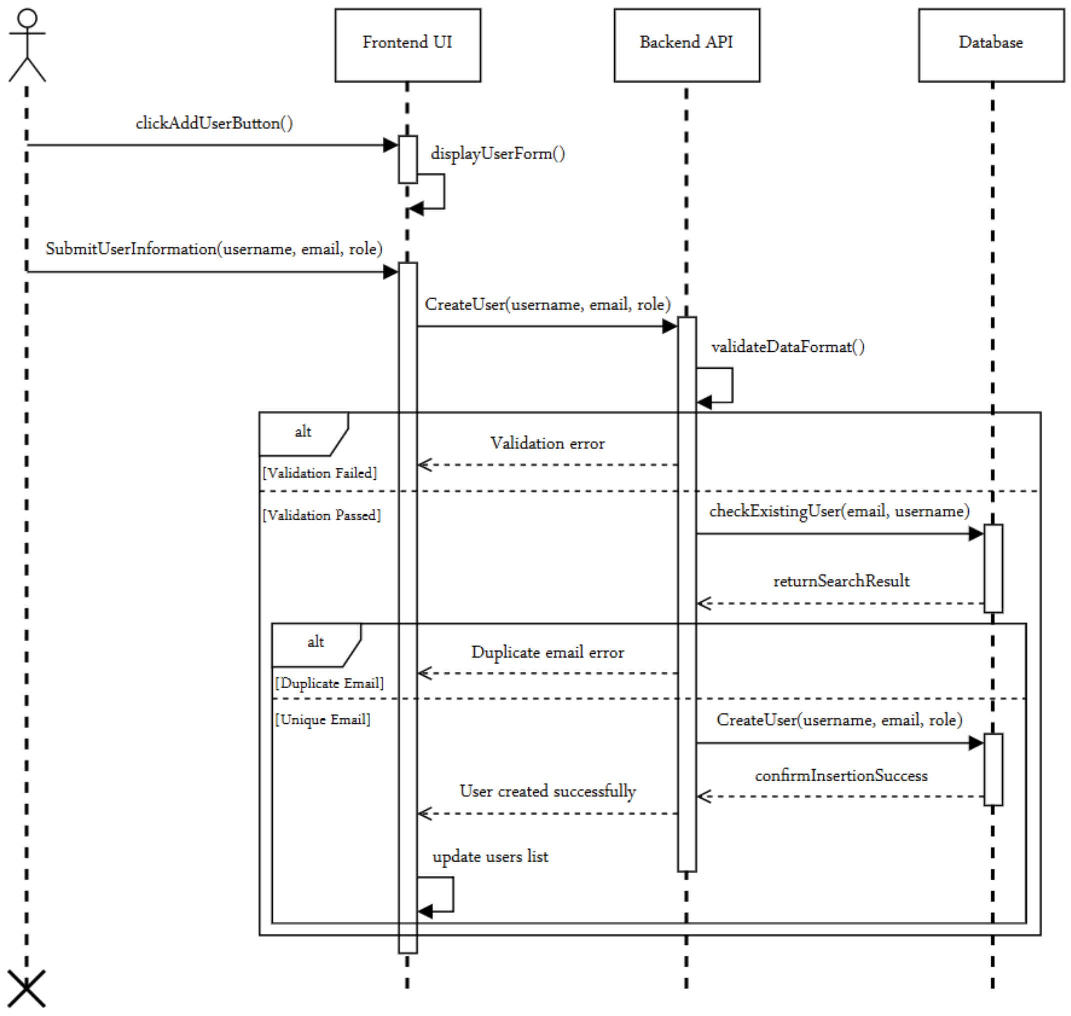


Figure 22 Create User - Sequence Diagram

Edit User flow UC-08:

Use case name	Edit User
Participating Actors	Admin
Main Flow	1. The administrator navigates to the **Users Management** page and selects an existing user. 2. The system displays the user's current information in an editable form. 3. The administrator updates the required fields (e.g., role, phone number, profile picture) and clicks **"Save."** 4. The system validates the updated data and saves the changes successfully. 5. The system displays a success message and updates the users list.
Alternative flows	**A1 – Validation Error:** • The system detects invalid data or missing required fields. • The system displays descriptive error messages. • The administrator corrects the data and resubmits the form. **A2 – Duplicate Data:** • The system detects that the updated email address or username already exists. • The system prevents saving the changes and notifies the administrator. • The flow returns to Step 3.
Entry condition	The user to be modified exists
Exit conditions	Modify user data and update user list successfully

Table 12 Edit User - Use Case Diagram

Activity diagram:

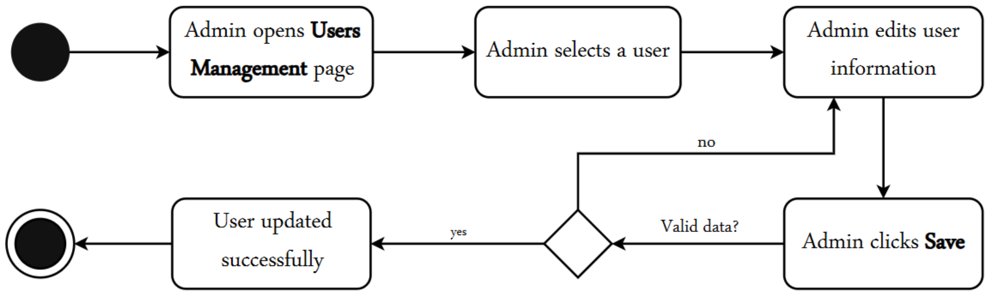

Admin opens Users Management page
Admin selects a user
Admin edits user information
no
Valid data?
Admin clicks Save
yes
User updated successfully

Figure 23 Edit User - Activity Diagram

Sequence diagram:

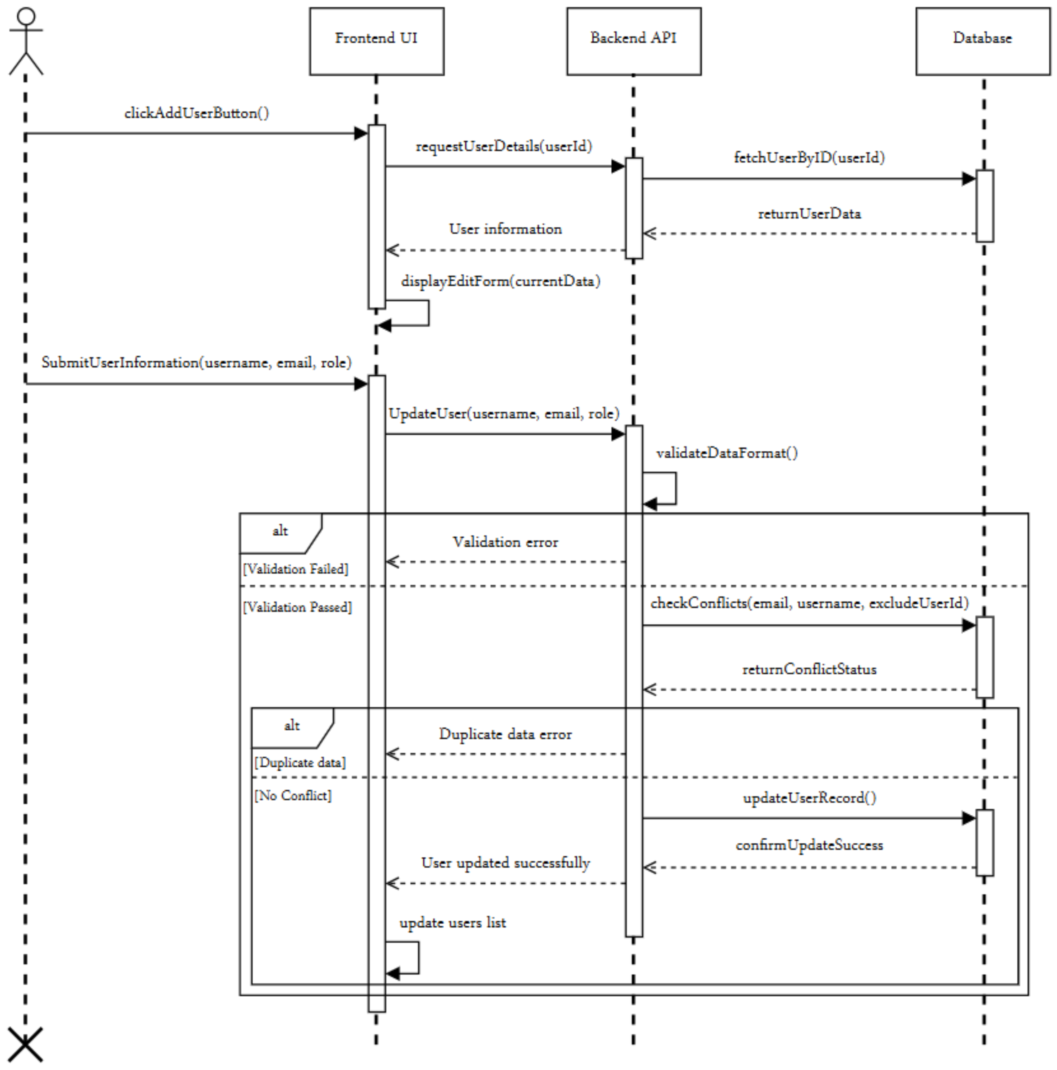


Figure 24 Edit User - Sequence Diagram

Archive User flow UC-09:

Use case name	Archive User
Participating Actors	Admin
Main Flow	1. Admin navigates to the **Users Management** page. 2. The system displays a list of active users. 3. The admin clicks **Archive** for a selected user. 4. The system prompts the admin to confirm the archive action. 5. The admin confirms the action. **6.** The system archives the user and updates the users list with a success message.
Alternative flows	**A1 – Cancel Archive:** • The admin cancels the confirmation. • The system keeps the user active. • The flow ends.
Entry condition	The selected user exists and is active
Exit conditions	Modify user data and update user list successfully

Table 13 Archive User - Use Case Diagram

Activity diagram:

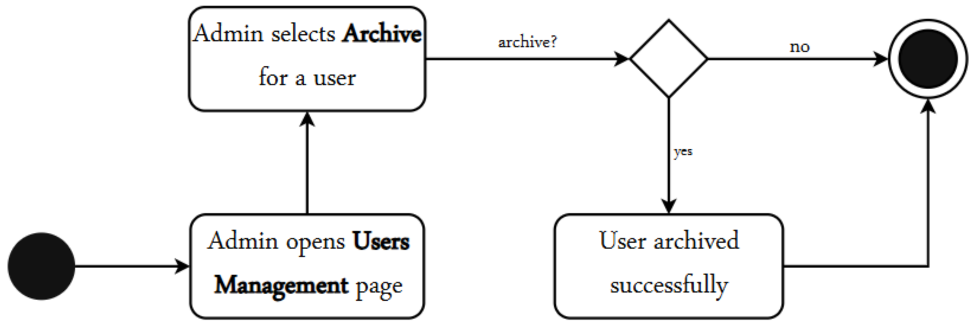

Admin selects **Archive** for a user
archive?
no
yes
Admin opens **Users Management** page
User archived successfully

Figure 25 Archive User - Activity Diagram

Sequence diagram:

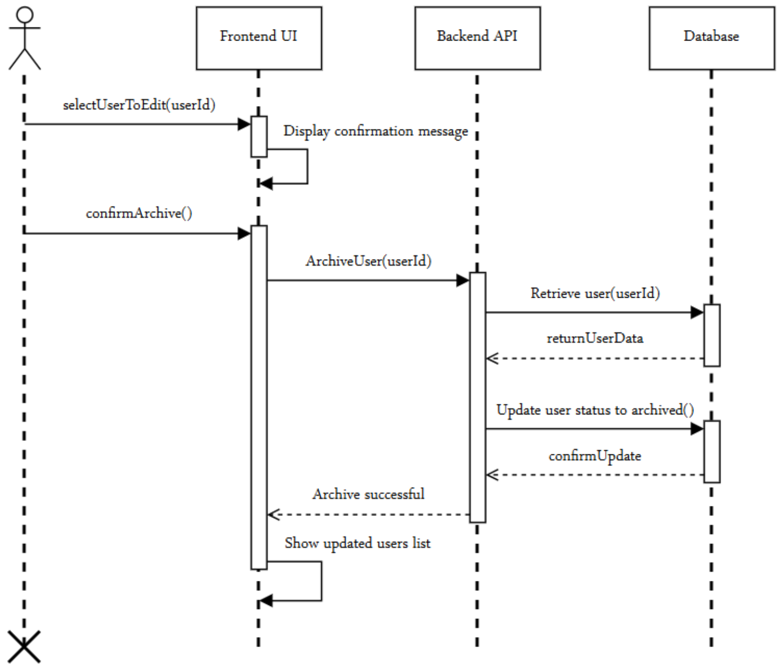


Figure 26 Archive User - Sequence Diagram

Restore User flow UC-10:

Use case name	Restore Archive User
Participating Actors	Admin
Main Flow	1. The administrator navigates to the **Users Management** page. 2. The administrator filters the list to display **Archived Users**. 3. The administrator selects an archived user and chooses the **“Restore”** action. 4. The system prompts the administrator to confirm the restore action. 5. The administrator confirms the action. 6. The system changes the user's status to active and displays a message indicating that the operation was successful.
Alternative flows	**A1 – Cancel Restore:** • The administrator selects **“Cancel.”** • The system cancels the restore operation without making any changes.

Entry condition	The selected user exists and is archived.
Exit conditions	• The user status is changed to **Active**. • appears again in the active users list.

Table 14 Restore Archive User - Use Case Diagram

Activity diagram:

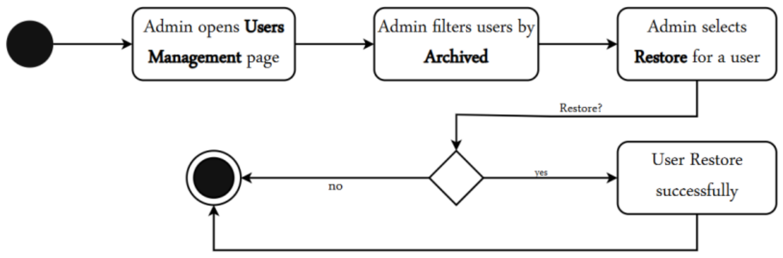

Admin opens **Users Management** page
Admin filters users by **Archived**
Admin selects **Restore** for a user
Restore?
yes
no
User Restore successfully

Figure 27 Restore Archive User - Activity Diagram

Sequence diagram:

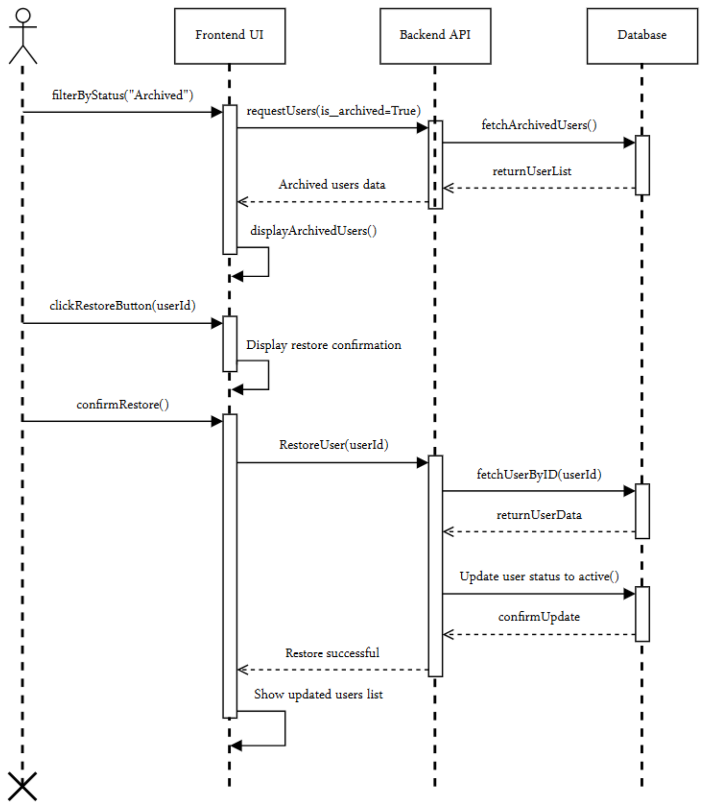


Figure 28 Restore Archive User - Sequence Diagram

View Users List flow UC-11:

Use case name	View Users List
Participating Actors	Admin
Main Flow	1. The administrator navigates to the **Users Management** page. 2. The system displays a paginated list of all users with their basic information (e.g., username, email, role, status).
Alternative flows	**A1 – Search Users:** • The administrator enters a keyword (username) in the search field. • The system filters the users list and displays matching results. **A2 – Filter Users:** • The administrator selects one or more filter criteria (e.g., role, status). • The system updates the list to display users matching the selected criteria.
Entry condition	The admin is logged in
Exit conditions	The user's list is displayed according to the selected criteria.

Table 15 View Users List - Use Case Diagram

Manage Vehicles flow UC-12:

Use case name	Manage Vehicles
Participating Actors	Admin
Description	Allows the admin to view, add, update, and delete vehicles from the system. (Covers requirements: RE-FR-IM-23, 24, 25, and 26).
Main Flow	1. The administrator navigates to the **Vehicle Management** page. 2. The system retrieves and displays the list of registered vehicles. 3. The administrator selects **"Add Vehicle."** 4. The system displays the vehicle creation form. 5. The administrator enters the vehicle data and clicks **"Save."** 6. The system validates the entered data and saves the new vehicle successfully. 7. The system displays a success message and updates the vehicles list.

Alternative flows	**A1 – Edit Vehicle:** 1. The administrator selects an existing vehicle and chooses **"Edit."** 2. The system displays the vehicle data in an editable form. 3. The administrator updates the data and clicks **"Save."** 4. The system validates the data and saves the changes. 5. The system updates the vehicles list. **A2 – Delete Vehicle:** 1. The administrator selects an existing vehicle and chooses **"Delete."** 2. The system prompts for deletion confirmation. 3. The administrator confirms the action. 4. The system deletes the vehicle and updates the vehicles list. **E1 – Validation Error:** 1. The system detects missing or invalid vehicle data. 2. The system displays descriptive error messages. 3. The administrator corrects the data and resubmits the form. 4. The flow returns to Step 5 of the Main Flow.
Entry condition	The administrator is logged in.
Exit conditions	A vehicle is added, edited, or deleted successfully.

Table 16 Manage Vehicles - Use Case Diagram

Activity diagram:

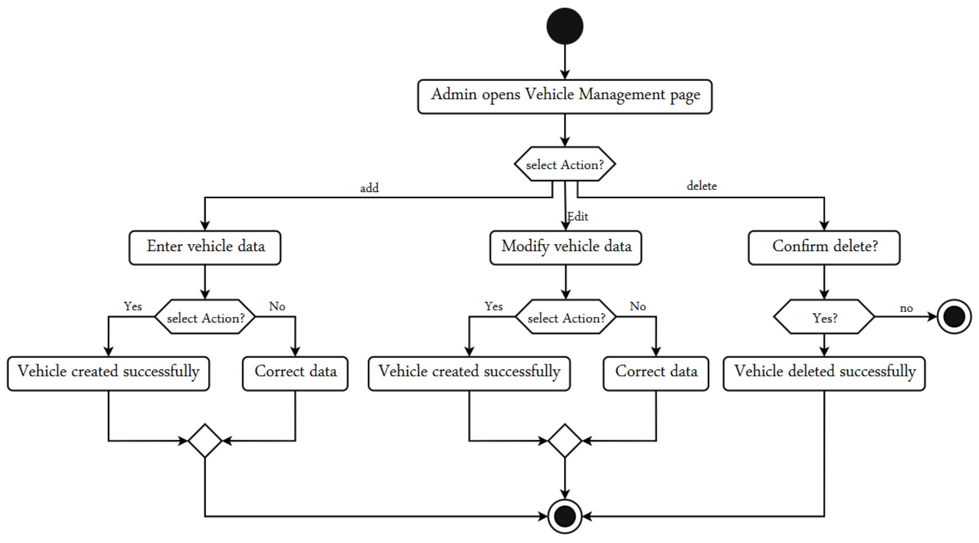


Figure 29 Manage Vehicles - Activity Diagram

Manage Bins flow UC-13:

Use case name	Manage Bins
Participating Actors	Admin
Description	Allows the admin to view, add, update, and delete Bins from the system. (Covers requirements: RE-FR-IM-27 to RE-FR-IM-31).
Main Flow	1. The admin navigates to the **Bins Management** page. 2. The system retrieves and displays the list of waste. 3. The admin selects **"Add Bin."** 4. The system displays a form requesting: bin name, GPS coordinates (latitude, longitude), capacity, and municipality. 5. The admin enters the required data and clicks **"Save."** 6. The system verifies the data and creates a new bin record. 7. The system displays a success message and updates the bins list.
Alternative flows	**A1 – Update Bin Details:** 1. The admin selects a bin from the list and clicks **"Edit."** 2. The system displays the current bin data in an editable form. 3. The admin updates the bin details (e.g., capacity) and clicks **"Save."** 4. The system updates the bin record successfully. **A2 – Delete Bin:** 1. The admin selects a bin and clicks **"Delete."** 2. The system displays a confirmation message. 3. The admin confirms the deletion. 4. The system permanently deletes the bin and updates the bins list.
Exception Flows	**E1 – Validation Error:** 1. The system detects invalid input 2. The system displays an error message. 3. The admin corrects the data and resubmits the form.
Entry condition	The administrator is logged in.
Exit conditions	Bin data is created, updated, deleted, or its status changed successfully.

Table 17 Manage Bins - Use Case Diagram

Activity diagram:

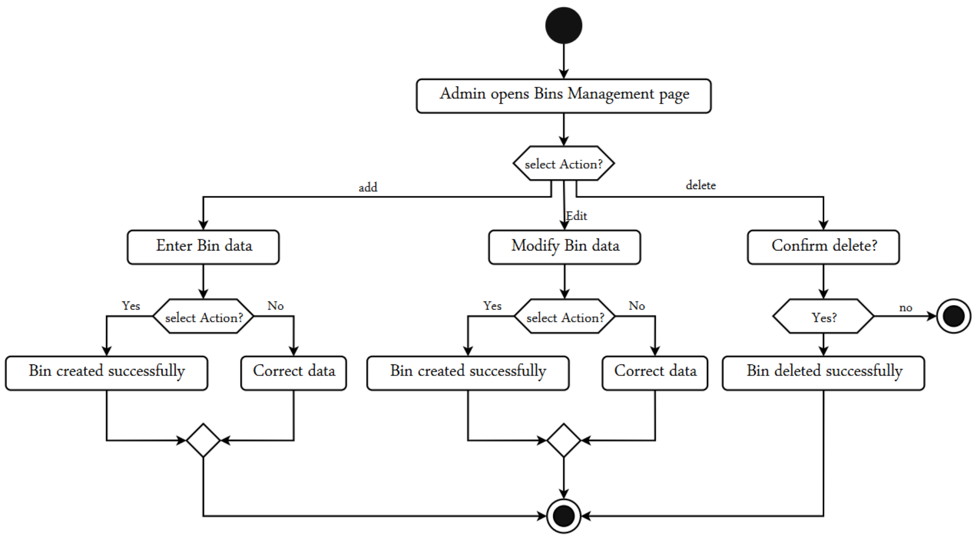


Figure 30 Manage Bins - Activity Diagram

Manage Municipalities flow UC-14:

Use case name	Manage Municipalities
Participating Actors	Admin
Description	Allows the admin to view, add, update, and delete Municipalities from the system. (Covers requirements: RE-FR-IM-15 to RE-FR-IM-18).
Main Flow	1. The admin navigates to the **Municipalities Management** page. 2. The system retrieves and displays the list of registered municipalities. 3. The admin selects **"Add Municipality."** 4. The system displays a form requesting: name and center coordinates. 5. The admin enters the details and clicks **"Save."** 6. The system verifies the data and creates a new municipal record. 7. The system displays a success message and updates the list of municipalities.
Alternative flows	**A1 – Update Municipality Details:** 1. The admin selects a municipality from the list and clicks **"Edit."** 2. The system loads the current municipality data. 3. The admin updates the name or coordinates and clicks **"Save."** 4. The system updates the municipality record successfully.

	A2 – Delete Municipality: 1. The admin selects a municipality and clicks **"Delete."** 2. The system checks for linked assets (bins/vehicles). 3. If no active dependencies exist, the system displays a confirmation message. 4. The admin confirms the deletion. 5. The system deletes the municipality and updates the list.
Exception Flows	**E2 – Validation Error:** 1. The system detects invalid input (e.g., duplicate municipality name or invalid coordinates). 2. The system displays an error message. **3.** The admin corrects the data and resubmits the form.
Entry condition	The administrator is logged in.
Exit conditions	Municipality data is created, updated, deleted, or its status changed successfully.

Table 18 Manage Municipalities - Use Case Diagram

Activity diagram:

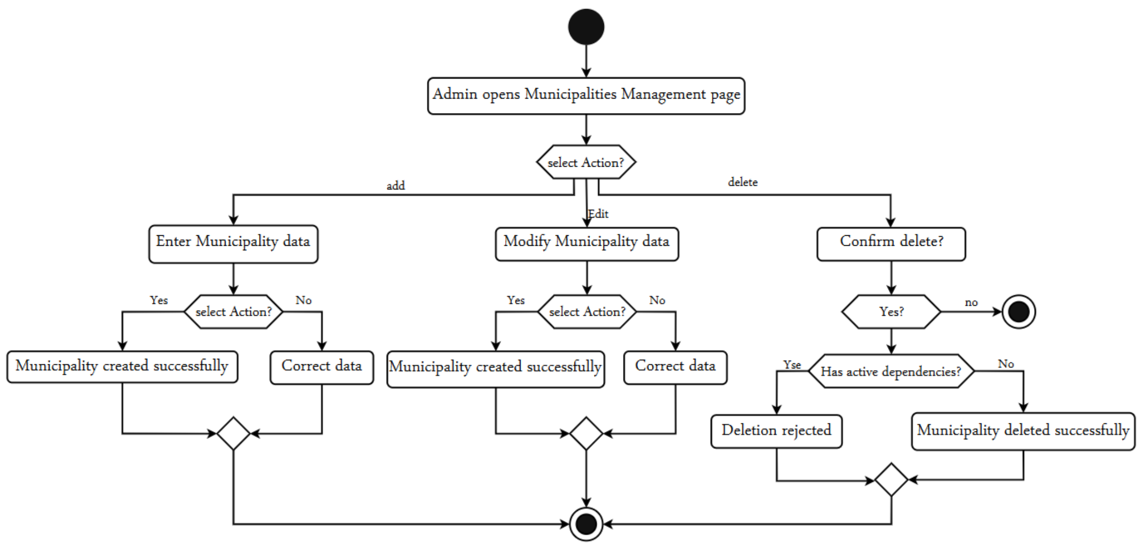


Figure 31 Manage Municipalities - Activity Diagram

Manage Landfills flow UC-15:

Use case name	Manage Landfills
Participating Actors	Admin
Description	Allows the admin to view, add, update, and delete Landfills from the system. (Covers requirements: RE-FR-IM-19 to RE-FR-IM-22).
Main Flow	1. The administrator navigates to the landfill management page. 2. The system retrieves and displays a list of registered landfills. 3. The administrator selects "Add Landfill." 4. The system displays a form requesting: the name of the landfill, GPS coordinates (latitude and longitude), and associated municipalities. 5. The administrator enters the details, selects one or more municipalities served by the landfill, and clicks "Save." 6. The system verifies the validity of the entries and creates a new landfill record. 7. The system displays a success message and updates the list of landfills.
Alternative flows	**A1 – Update Landfill Details:** 1. The admin selects a landfill from the list and clicks **"Edit."** 2. The system loads the current landfill data. 3. The admin modifies the details and clicks **"Save."** 4. The system updates the landfill record successfully. **A2 – Delete Landfill:** 1. The admin selects a landfill and clicks **"Delete."** 2. The system displays a confirmation message. 3. The admin confirms the deletion. 4. The system deletes the landfill record and updates the list.
Entry condition	The administrator is logged in.
Exit conditions	landfill data is created, updated, deleted, or its status changed successfully.

Table 19 Manage Landfills - Use Case Diagram

Activity diagram:

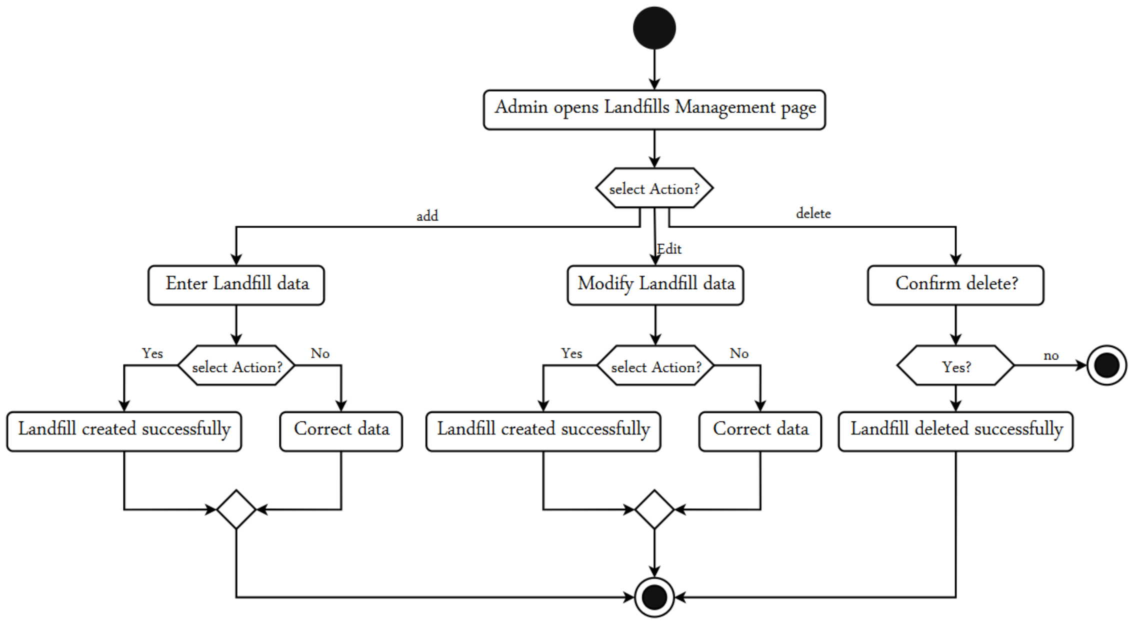


Figure 32 Manage Landfills - Activity Diagram

2.3.3 Actor: Planner

Create Collection Scenario flow UC-16:

Use case name	Create Collection Scenario
Participating Actors	Planner
Main Flow	1. The planner goes to the “Collection Plans” page and clicks on “Add Plan.” 2. The system displays a form requesting: municipality, starting point, collection date, vehicle, and target Bins. 3. The planner selects the municipality and collection date. 4. The system filters the available vehicles (excluding those that are busy on the same date). 5. The planner selects a suitable vehicle. 6. The system displays the available Bins (active Bins that are not associated with other scenarios on the same date and belong to the selected municipality). 7. The planner selects the target Bins and clicks “Save.” 8. The system generates an automatic name for the scenario (if not entered) and saves the scenario as unsolved.
Exception flows	**E1 – Validation Error (Resource Busy):** 1. The Planner attempts to select a vehicle or bin that is already assigned on the selected date. 2. The system prevents saving and displays an error message: *“Resource is already in use on this date.”*
Entry condition	The Planner is logged in.
Exit conditions	A new collection scenario is created.

Table 20 Create Collection Scenario - Use Case Diagram

Activity diagram:

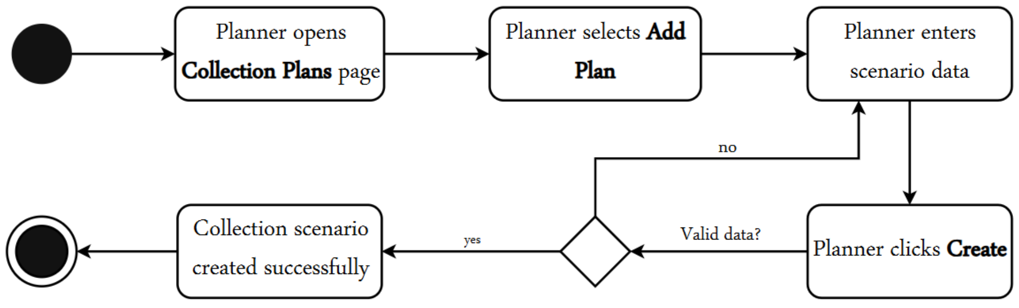

Planner opens **Collection Plans** page
Planner selects **Add Plan**
Planner enters scenario data
no
Planner clicks **Create**
Valid data?
yes
Collection scenario created successfully

Figure 33 Create Collection Scenario - Activity Diagram

Sequence diagram:

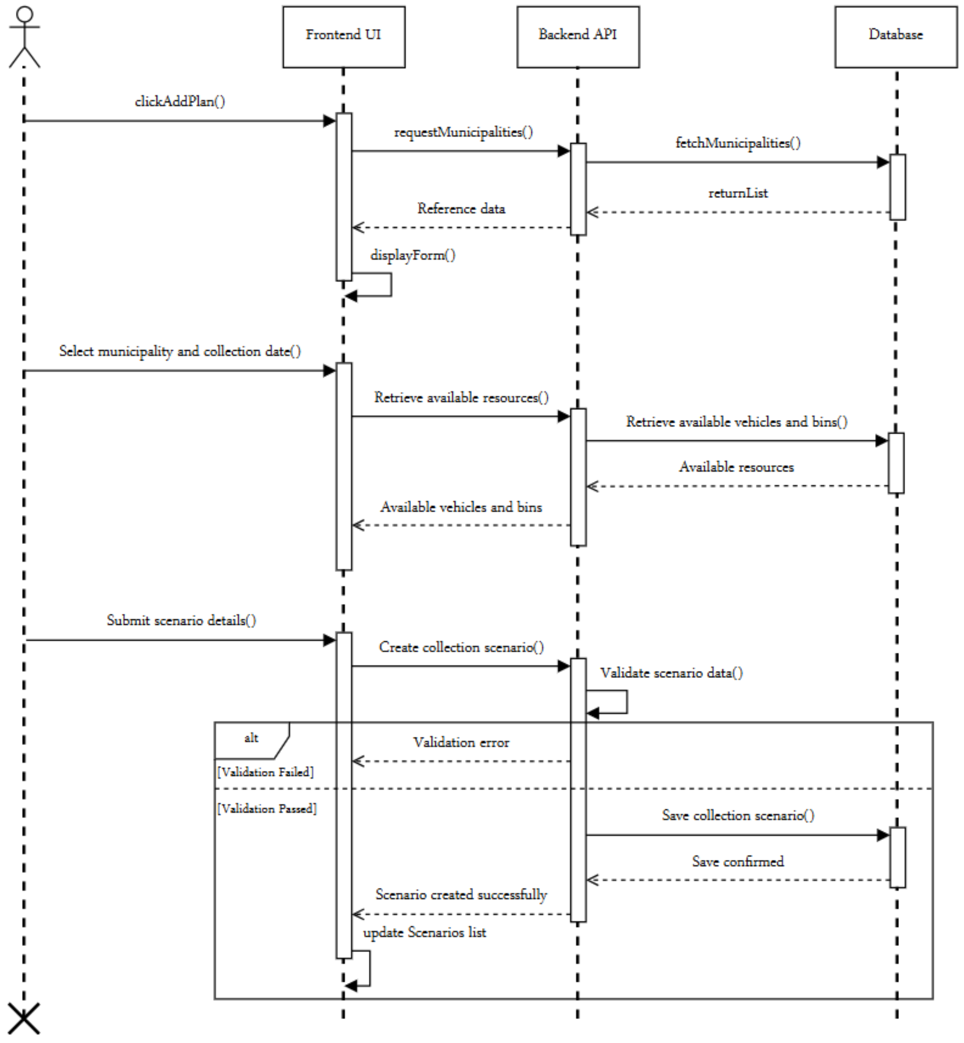


Figure 34 Create Collection Scenario - Sequence Diagram

View & Filter Collection Plans flow UC-17:

Use case name	View & Filter Collection Plans
Participating Actors	Planner
Main Flow	1. The Planner navigates to the **Scenarios** management page. 2. The system retrieves the list of scenarios created by the Planner. 3. The system applies the default filter (**Current & Future Plans**). 4. The system displays a paginated list showing: scenario name, municipality, collection date, and optimization status. 5. The Planner browses the list using pagination controls.
Alternative flows	**A1 – Search by Keyword:** 1. The Planner enters a keyword (e.g., "Midan Zone") in the search field. 2. The system filters the list and displays scenarios with matching names. **A2 – Filter by Date / Status (Archived):** 1. The Planner selects **"Archived"** from the status filter. 2. The system retrieves and displays scenarios with past collection dates. **A4 – Empty Results:** 1. The applied search or filter criteria return no matching scenarios. 2. The system displays a **"No plans found"** message. 3. The Planner clears the filters to reset the view.
Entry condition	The Planner is logged in.
Exit conditions	The desired list of collection plans is displayed according to the selected criteria.

Table 21 View & Filter Collection Plans - Use Case Diagram

Update Scenario flow UC-18:

Use case name	Update Scenario
Participating Actors	Planner
Main Flow	1. The planner selects "Edit" for a specific scenario. 2. The system loads the current scenario data. 3. The planner modifies the scenario inputs (e.g., adds boxes or changes the assigned vehicle). 4. The planner clicks "Save." 5. The system checks whether important data (bins or vehicle) has changed. 6. If important changes are detected, the system deletes any previous solutions and creates new ones. 7. The system saves the updated scenario details.
Exception flows	**E1 – Permission Denied:** 1. The Planner attempts to edit a scenario they did not create. 2. The system blocks the request and returns an access denied response (HTTP 403).
Entry condition	The Planner is logged in.
Exit conditions	The scenario is updated successfully.

Table 22 Update Scenario - Use Case Diagram

Activity diagram:

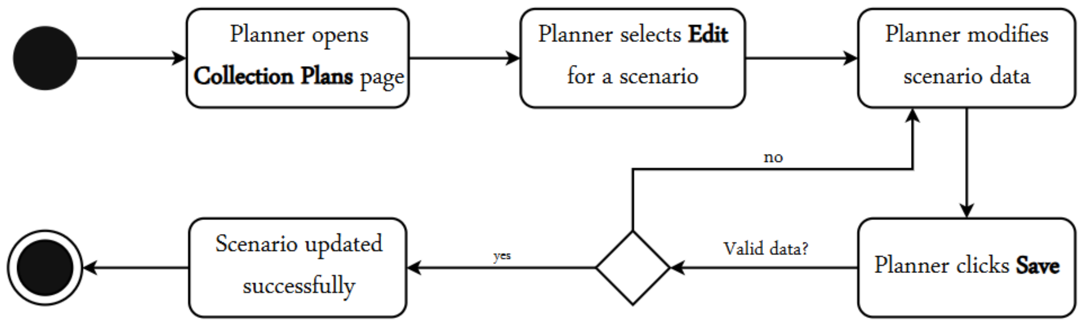

Planner opens **Collection Plans** page
Planner selects **Edit** for a scenario
Planner modifies scenario data
no
Planner clicks **Save**
Valid data?
yes
Scenario updated successfully

Figure 35 Update Scenario - Activity Diagram

Sequence diagram:

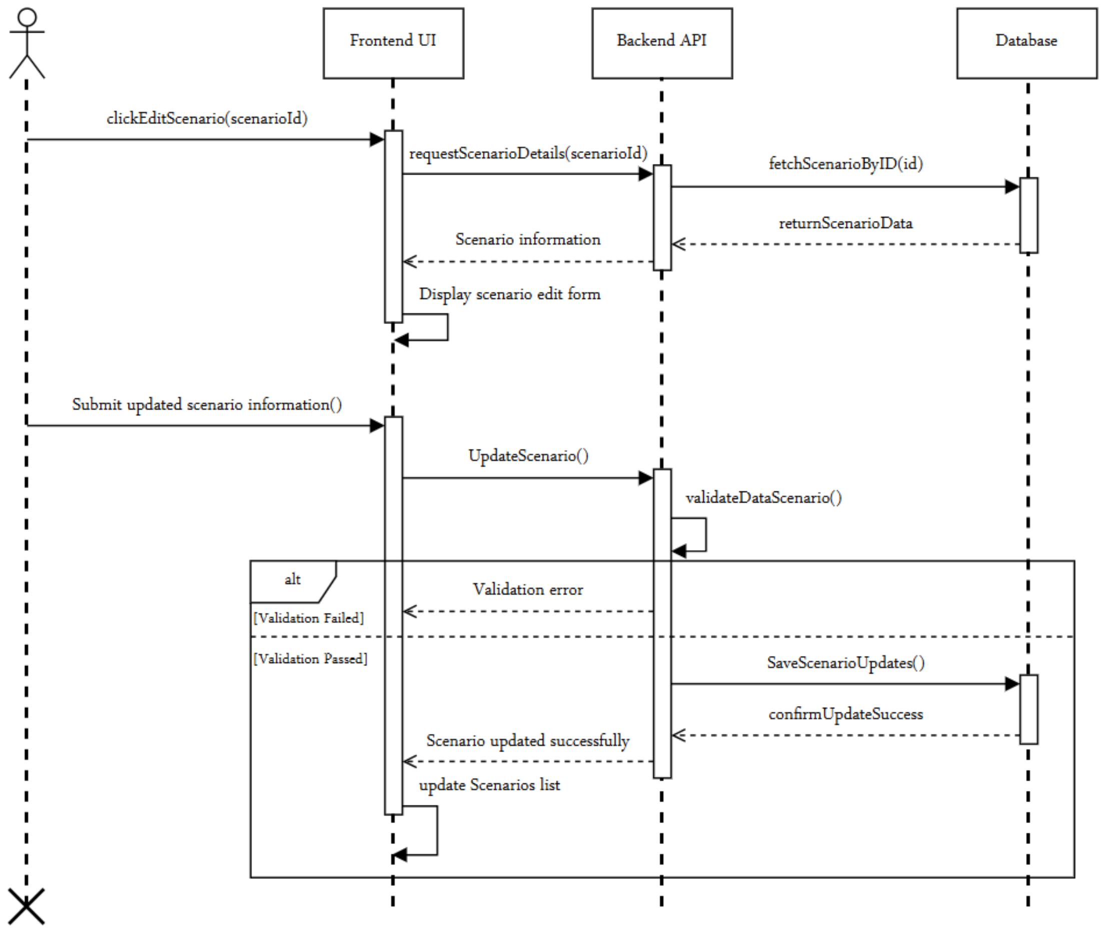


Figure 36 Update Scenario - Sequence Diagram

Delete Scenario flow UC-19:

Use case name	Delete Scenario
Participating Actors	Planner
Main Flow	1. The Planner selects **"Delete"** for a specific scenario. 2. The system displays a confirmation warning: *"This will delete the plan and any generated routes."* 3. The Planner confirms the deletion 4. The system permanently deletes the scenario from the database. 5. The system automatically removes any related **RouteSolution** records (cascade delete).
Exception flows	**E1 – Permission Denied:** 1. The Planner attempts to delete a scenario created by another user. 2. The system blocks the action and denies access.
Entry condition	The Planner is logged in.
Exit conditions	The scenario is deleted successfully.

Table 23 Delete Scenario - Use Case Diagram

Activity diagram:

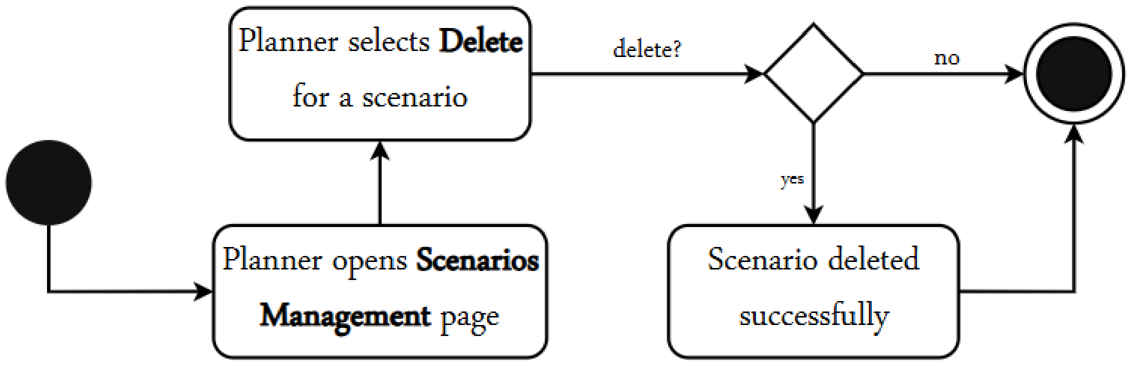

Planner selects **Delete** for a scenario
delete?
no
yes
Planner opens **Scenarios Management** page
Scenario deleted successfully

Figure 37 Delete Scenario - Activity Diagram

Sequence diagram:

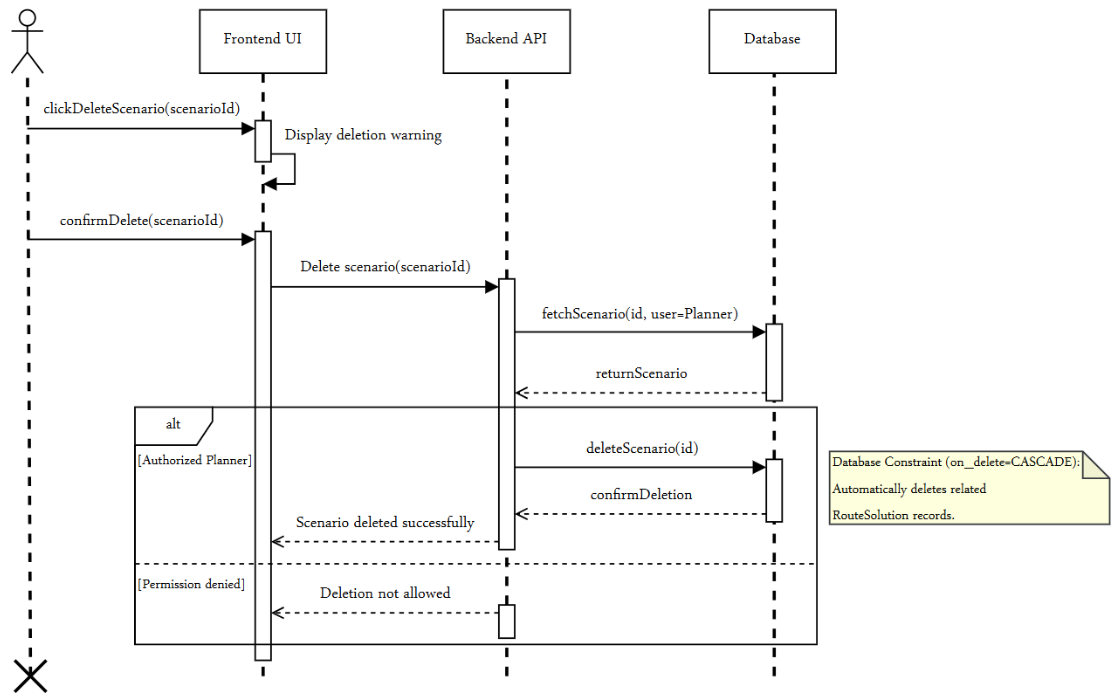

Frontend UI
Backend API
Database
clickDeleteScenario(scenarioId)
Display deletion warning
confirmDelete(scenarioId)
Delete scenario(scenarioId)
fetchScenario(id, user=Planner)
returnScenario
alt
[Authorized Planner]
deleteScenario(id)
confirmDeletion
Scenario deleted successfully
[Permission denied]
Deletion not allowed
Database Constraint (on_delete=CASCADE): Automatically deletes related RouteSolution records.

Figure 38 Delete Scenario - Sequence Diagram

3. Summary

In this chapter, we have conducted a comprehensive analysis of the system's functional and non-functional requirements. We detailed the core services, including Account Management, Infrastructure Management, and Planning Scenarios, supported by UML diagrams (Use Case, Activity, and Sequence) to illustrate the system's behavior.

These requirements establish a solid foundation for the next phase. In **Chapter 4**, we will translate these analytical models into concrete system designs, detailing the database structure and software architecture required to build the solution.

Chapter 5 System Design

1. Introduction

The purpose of this chapter is to present the comprehensive design of the Route Optimization for Garbage Collection system. Following the detailed requirements analysis in Chapter 3, this chapter bridges the gap between the problem domain and the solution domain. It details the architectural choices, database schema, and structural components that enable the system to function efficiently, serving as the blueprint for the implementation phase.

2. System Architecture

The system is designed using a modern Three-Tier Architecture (Client-Server model) with a fully decoupled Frontend and Backend. This approach ensures scalability, maintainability, and allows for independent development of the user interface and the core logic.

The architecture consists of the following layers:

1. **Presentation Layer (Frontend):**
   - **Technology:** Built using **React.js** powered by **Vite** for high performance.
   - **Responsibility:** Handles all user interactions, renders the dynamic dashboard, and visualizes geospatial data (maps, routes, and bins) using **Leaflet**. It communicates with the backend via RESTful APIs using **Axios**.

2. **Application Layer (Backend):**
   - **Technology:** Developed using Python and the Django REST Framework **(DRF).**
   - **Responsibility:** Serves as the system's brain. It manages authentication (JWT), validates business rules, and hosts the **Optimization Engine** which integrates **Google OR-Tools** and **OSRM** to calculate the optimal collection routes.

3. **Data Layer (Database):**
   - **Technology:** PostgreSQL.
   - **Responsibility:** Stores structured relational data including user profiles, infrastructure assets (bins, vehicles, landfills), and the generated planning scenarios.

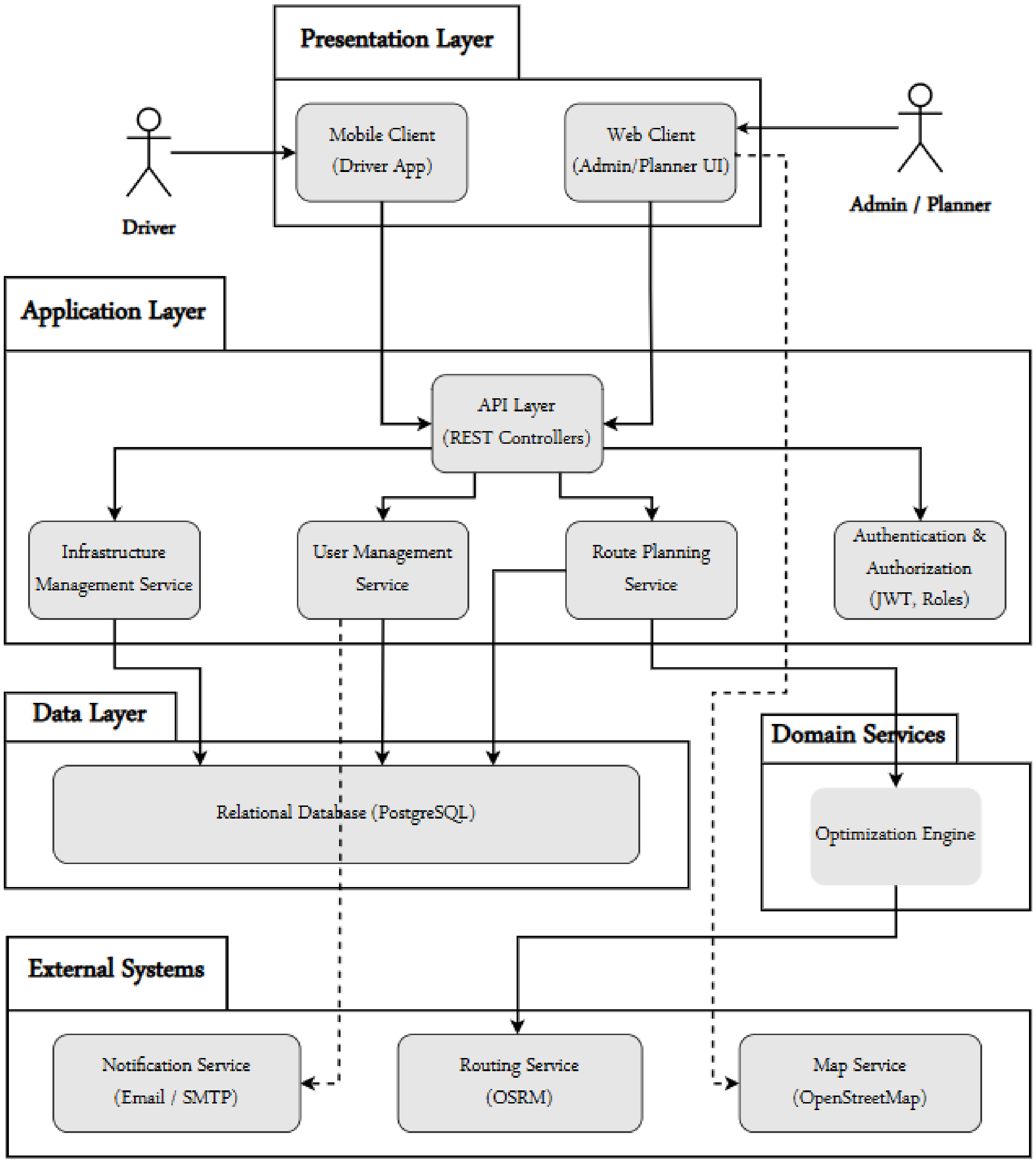

Presentation Layer
Mobile Client (Driver App)
Web Client (Admin/Planner UI)
Driver
Admin / Planner
Application Layer
API Layer (REST Controllers)
Infrastructure Management Service
User Management Service
Route Planning Service
Authentication & Authorization (JWT, Roles)
Data Layer
Relational Database (PostgreSQL)
Domain Services
Optimization Engine
External Systems
Notification Service (Email / SMTP)
Routing Service (OSRM)
Map Service (OpenStreetMap)

Figure 39 System Architecture

3. Database Design

The database is the backbone of the application, designed to ensure data integrity and efficient retrieval. We utilized a relational database model implemented in PostgreSQL to manage the complex relationships between municipalities, assets, and plans.

3.1 Entity Relationship Diagram (ERD)

The following Entity Relationship Diagram (ERD) illustrates the logical structure of the database, showing the entities, their attributes, and the relationships between them.

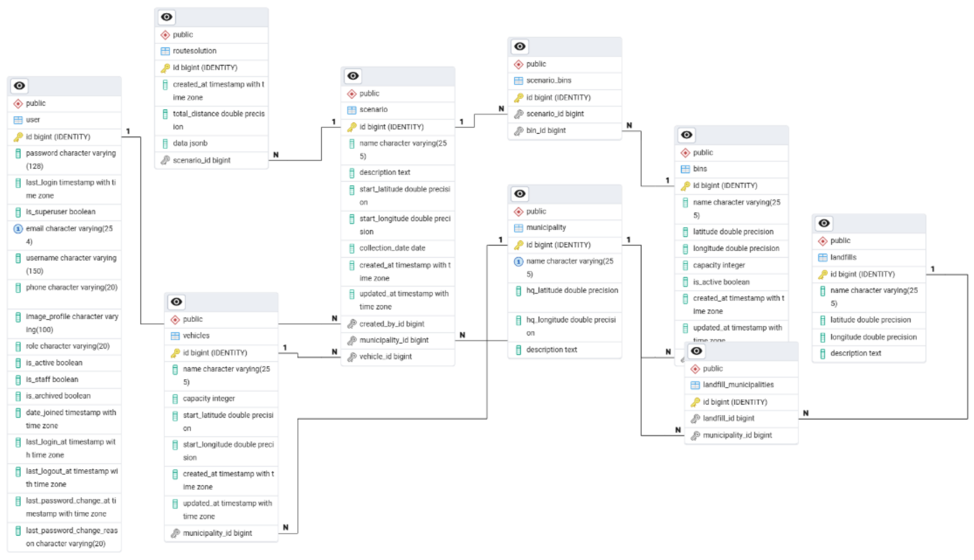

public
user
id bigint (IDENTITY)
password character varying(128)
last_login timestamp with time zone
is_superuser boolean
email character varying(254)
username character varying(150)
phone character varying(20)
image_profile character varying(100)
role character varying(20)
is_active boolean
is_staff boolean
is_archived boolean
date_joined timestamp with time zone
last_login_at timestamp with time zone
last_logout_at timestamp with time zone
last_password_change_at timestamp with time zone
last_password_change_reason character varying(20)
public
routesolution
id bigint (IDENTITY)
created_at timestamp with time zone
total_distance double precision
data jsonb
scenario_id bigint
public
vehicles
id bigint (IDENTITY)
name character varying(255)
capacity integer
start_latitude double precision
start_longitude double precision
created_at timestamp with time zone
updated_at timestamp with time zone
municipality_id bigint
public
scenario
id bigint (IDENTITY)
name character varying(255)
description text
start_latitude double precision
start_longitude double precision
collection_date date
created_at timestamp with time zone
updated_at timestamp with time zone
created_by_id bigint
municipality_id bigint
vehicle_id bigint
public
scenario_bins
id bigint (IDENTITY)
scenario_id bigint
bin_id bigint
public
municipality
id bigint (IDENTITY)
name character varying(255)
hq_latitude double precision
hq_longitude double precision
description text
public
bins
id bigint (IDENTITY)
name character varying(255)
latitude double precision
longitude double precision
capacity integer
is_active boolean
created_at timestamp with time zone
updated_at timestamp with time zone
public
landfill_municipalities
id bigint (IDENTITY)
landfill_id bigint
municipality_id bigint
public
landfills
id bigint (IDENTITY)
name character varying(255)
latitude double precision
longitude double precision
description text
1
N

Figure 40 Database Entity Relationship Diagram (ERD)

4. Structural Design

To provide a detailed view of the system's internal structure and the relationships between its components, we utilize the Class Diagram. This diagram represents the static view of the application, focusing on the backend models and their interactions within the Django framework.

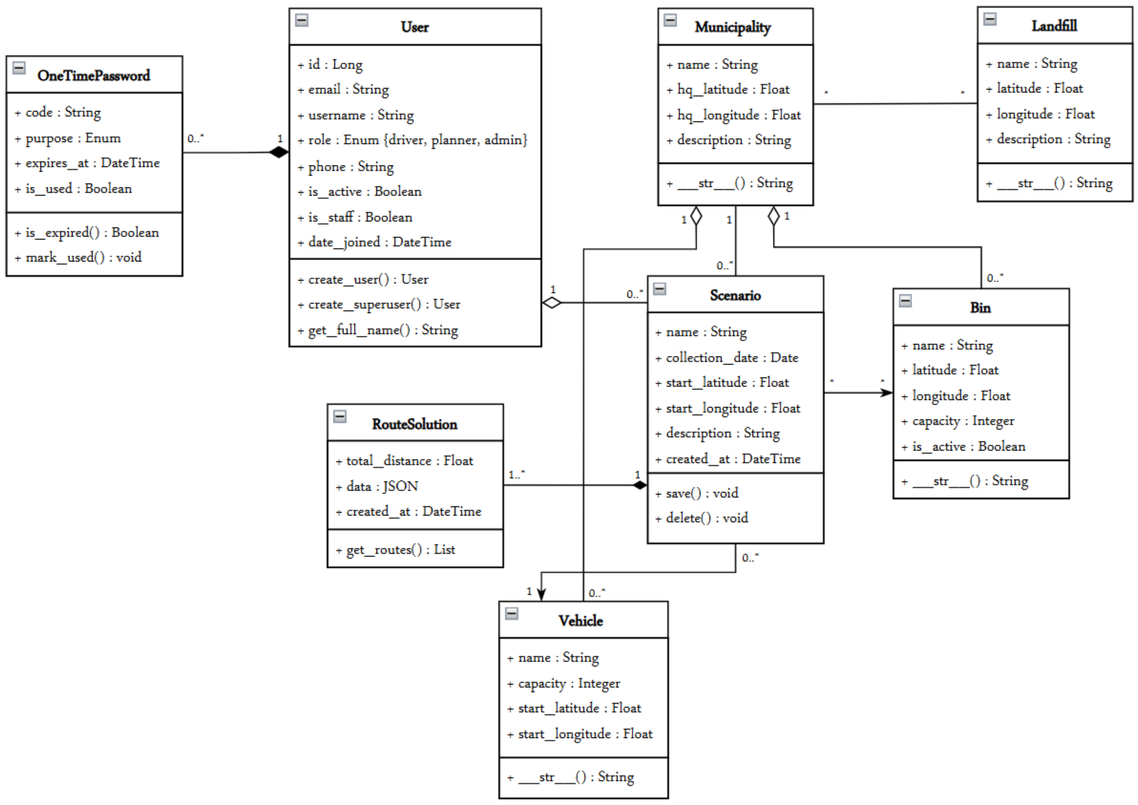


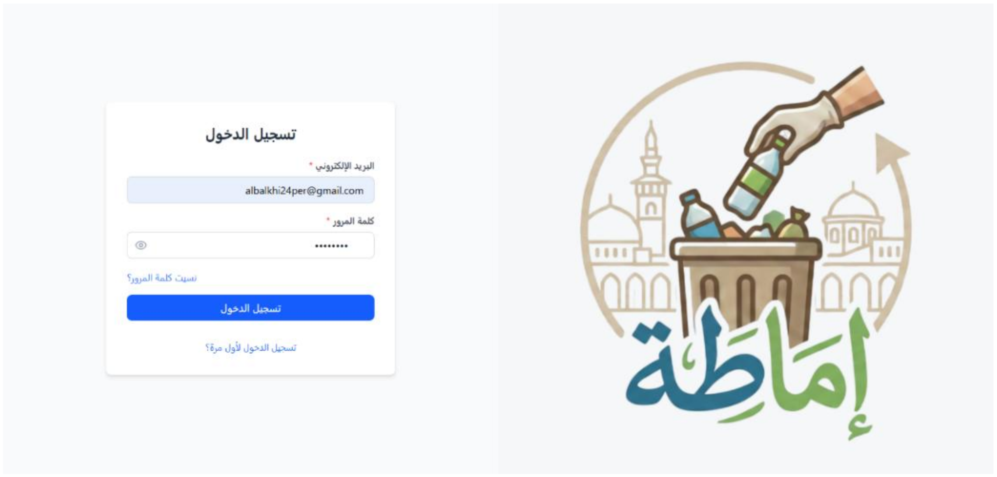
Figure 41 Class Diagram

5. User Interface Design

The User Interface (UI) is designed to be intuitive and responsive, ensuring that non-technical users (like municipal planners) can easily interact with complex optimization tools. The frontend is built using React and Material UI (or Tailwind CSS) principles.

Below are screenshots of the key system interfaces:

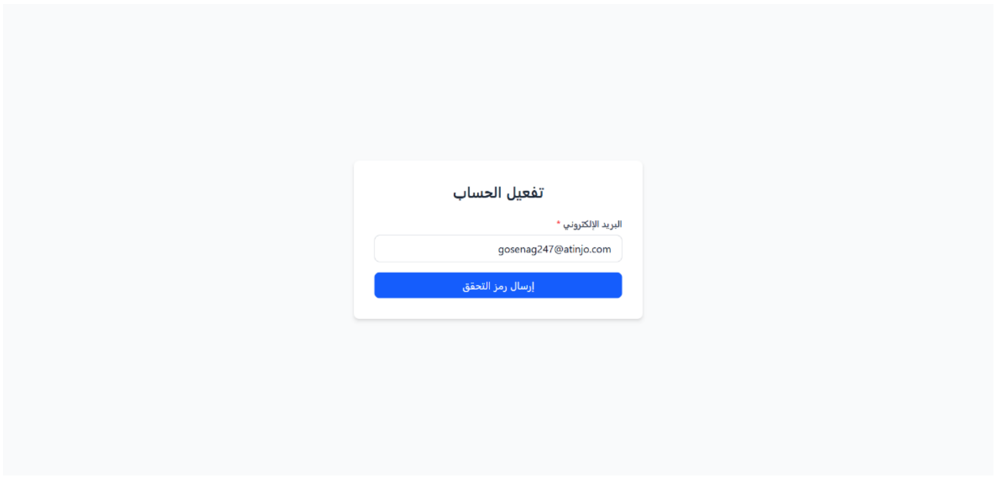
5.1 Common User Interface



البريد الإلكتروني *: albalkhi24per@gmail.com

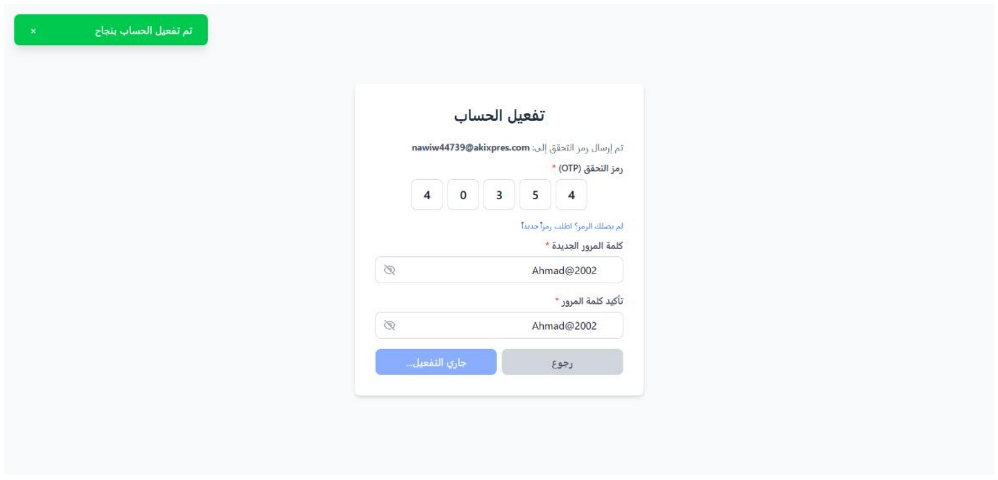
كلمة المرور *: ••••••••

Figure 42 Login - Common User Interface



البريد الإلكتروني *: gosenag247@atinjo.com

Figure 43 Request OTP to Activate Account Interface


× تم تفعيل الحساب بنجاح

تفعيل الحساب

تم إرسال رمز التحقق إلى: **nawiw44739@akixpres.com**

رمز التحقق (OTP) *

4 0 3 5 4

لم يصلك الرمز؟ اطلب رمزاً جديداً

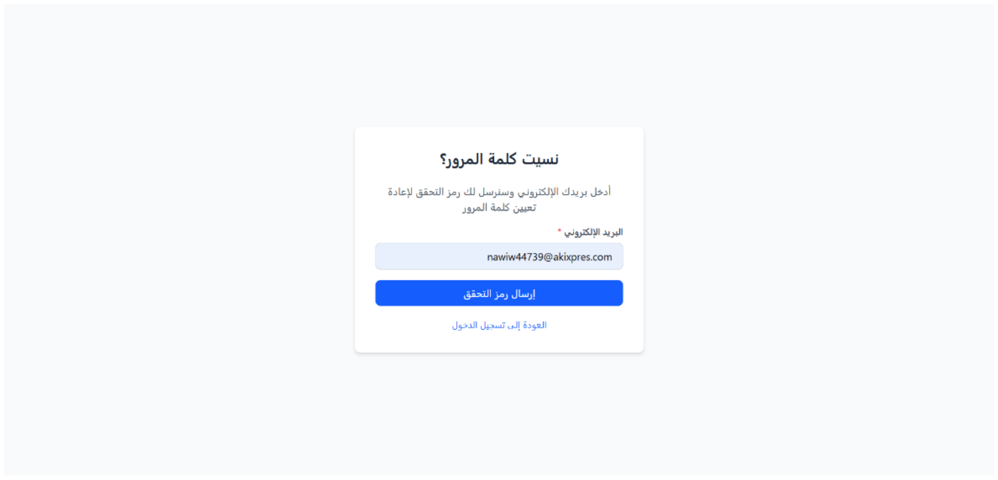
كلمة المرور الجديدة *

Ahmad@2002

تأكيد كلمة المرور *

Ahmad@2002

Figure 44 Activate Account Interface

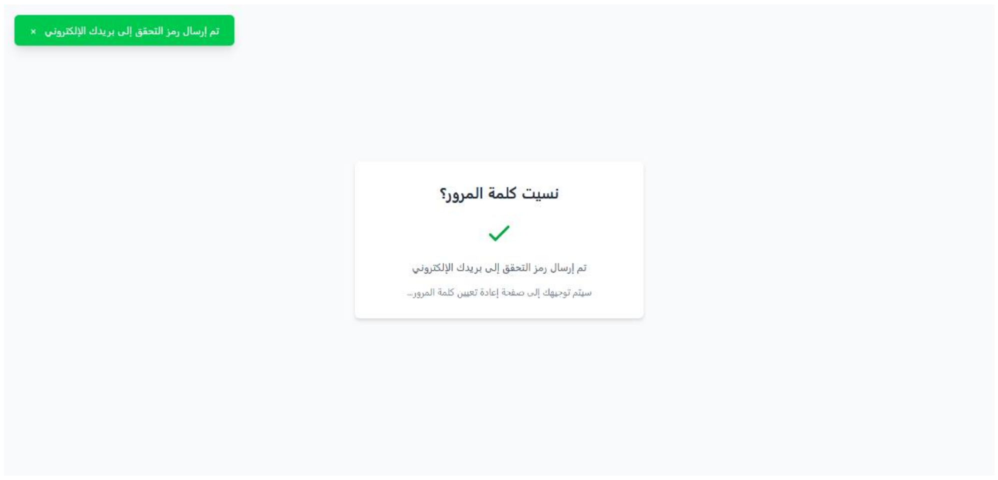

نسيت كلمة المرور؟

أدخل بريدك الإلكتروني وسنرسل لك رمز التحقق لإعادة تعيين كلمة المرور

البريد الإلكتروني *

nawiw44739@akixpres.com

Figure 45 Request OTP to Reset Password Interface


× تم إرسال رمز التحقق إلى بريدك الإلكتروني

نسيت كلمة المرور؟

✓

تم إرسال رمز التحقق إلى بريدك الإلكتروني

سيتم توجيهك إلى صفحة إعادة تعيين كلمة المرور...

Figure 46 Verification Code Sent Confirmation Interface

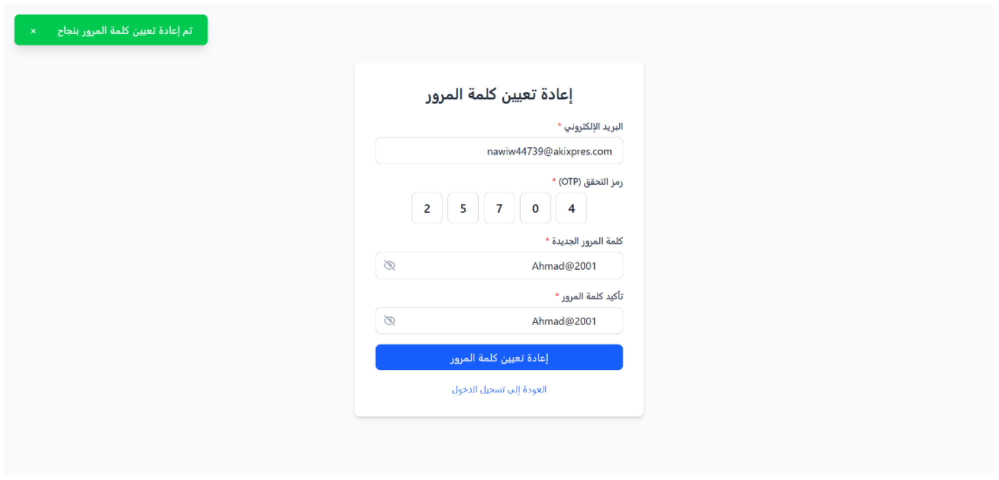


Figure 47 Reset Password Interface

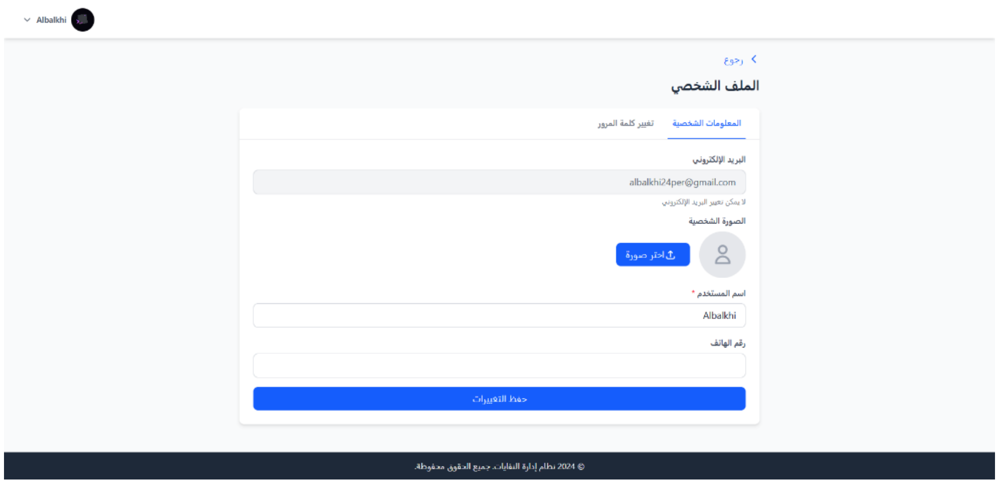


Figure 48 Edit Profile Interface

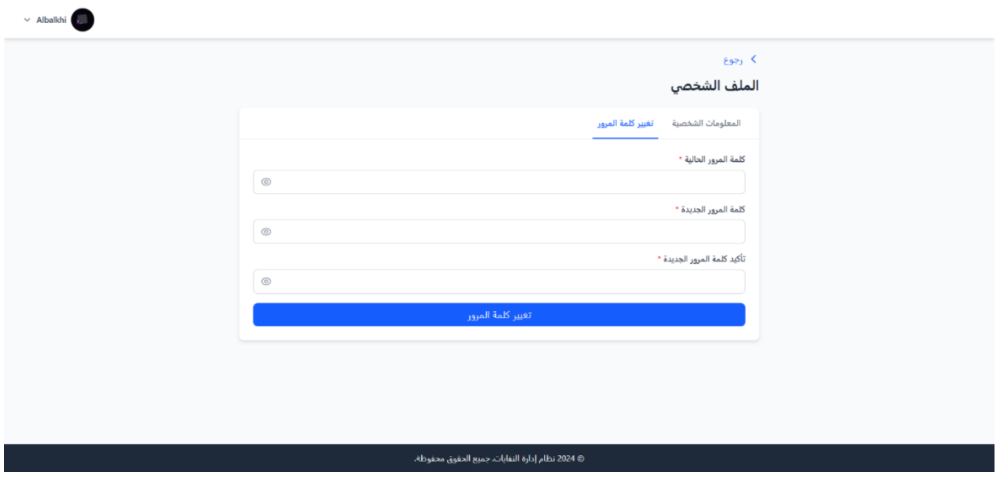


Figure 49 Change Password Interface

5.2 Admin User Interface

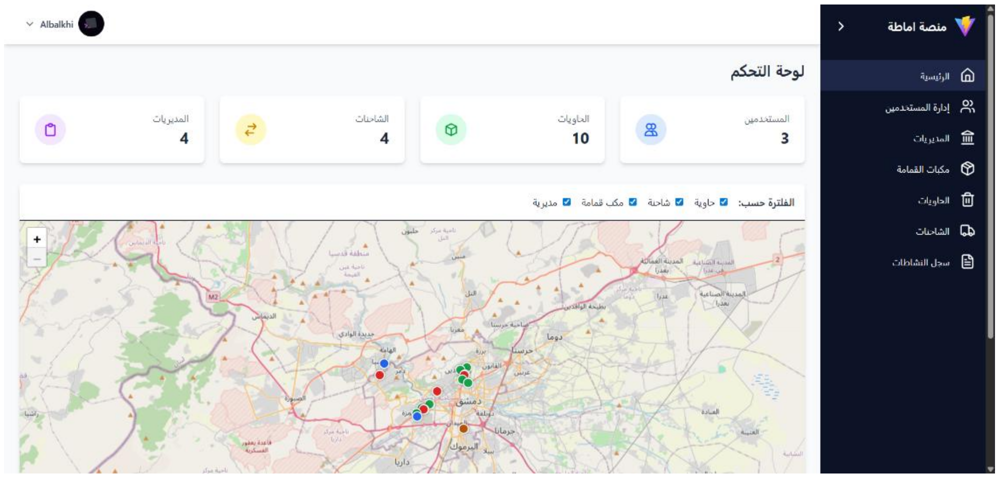
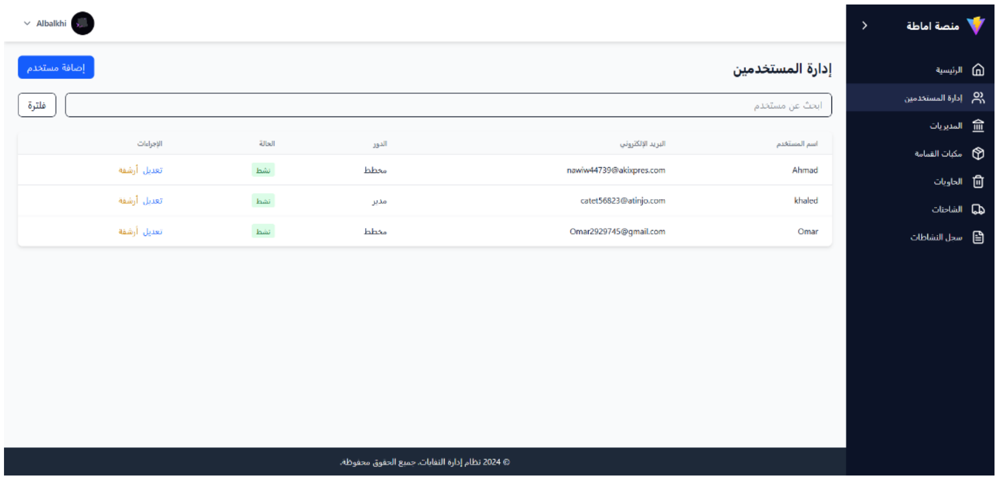


Figure 50 Admin Dashboard Interface

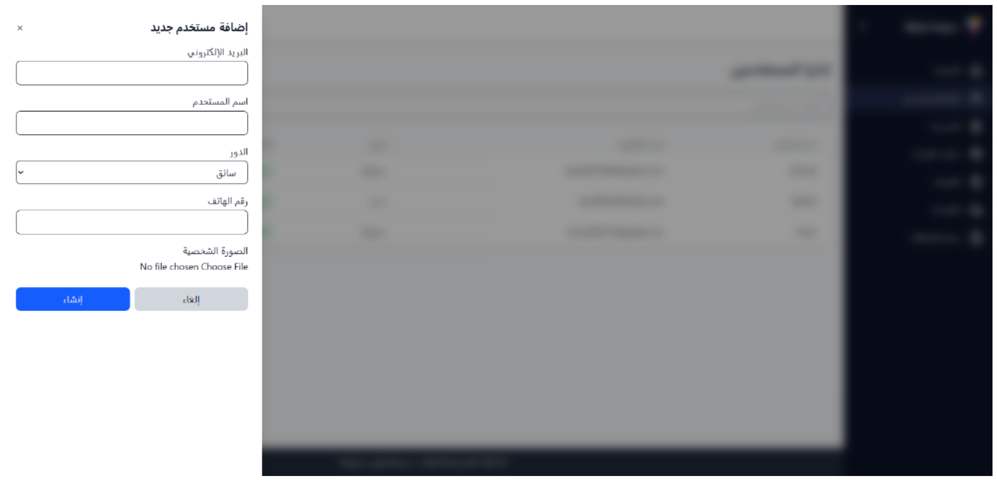


إدارة المستخدمين

اسم المستخدم	البريد الإلكتروني	الدور	الحالة	الإجراءات
Ahmad	nawiw44739@akixpres.com	مخطط	نشط	تعديل أرشفة
khaled	catet56823@atinjo.com	مدير	نشط	تعديل أرشفة
Omar	Omar2929745@gmail.com	مخطط	نشط	تعديل أرشفة

© 2024 نظام إدارة النفايات. جميع الحقوق محفوظة.

Figure 51 User Managment - Main Page



إضافة مستخدم جديد ×

البريد الإلكتروني

اسم المستخدم

الدور

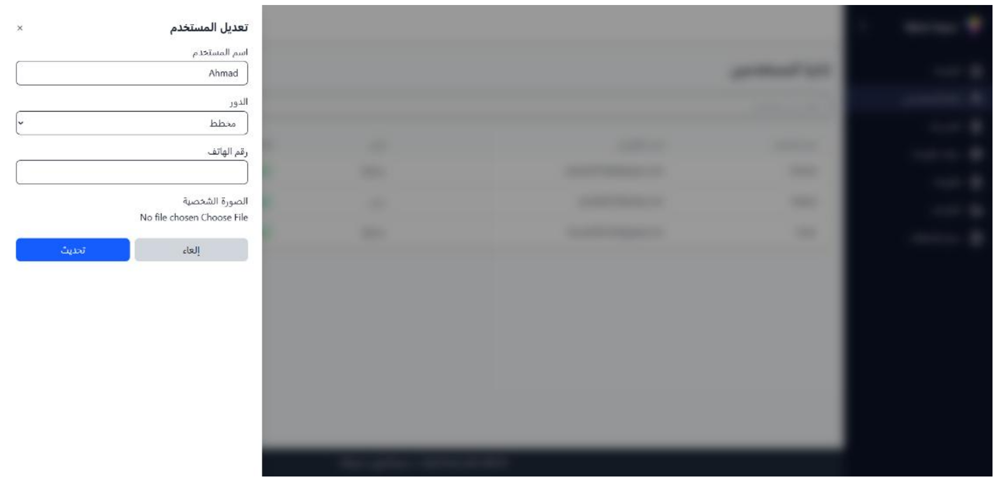
سائق

رقم الهاتف

الصورة الشخصية

No file chosen Choose File

Figure 52 User Management - Create User



تعديل المستخدم ×

اسم المستخدم

Ahmad

الدور

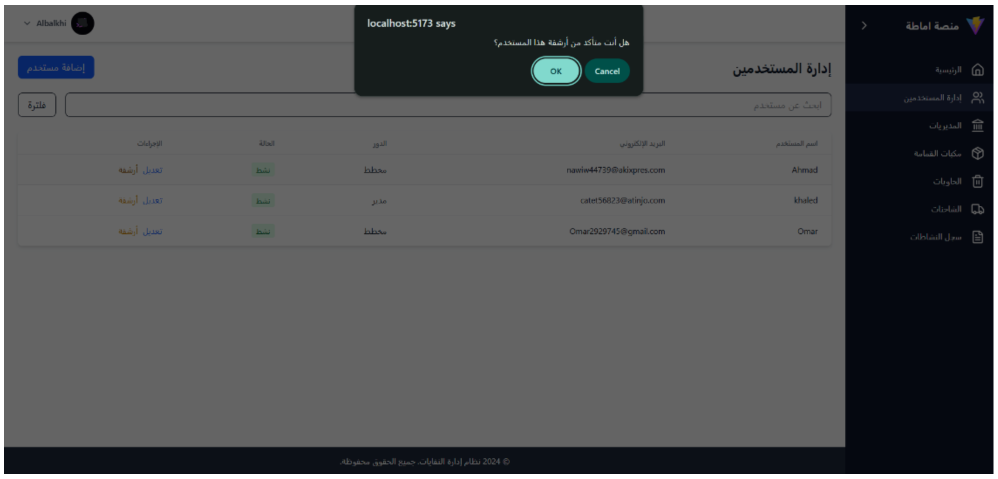
محطط

رقم الهاتف

الصورة الشخصية

No file chosen Choose File

Figure 53 User Management - Update User



localhost:5173 says

هل أنت متأكد من أرشفة هذا المستخدم؟

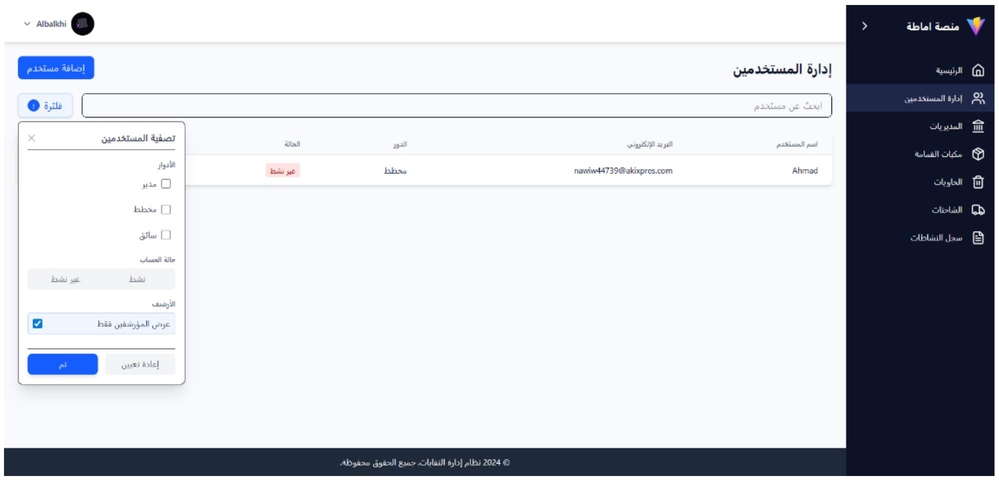
إدارة المستخدمين

ابحث عن مستخدم

اسم المستخدم	البريد الإلكتروني	الدور	الحالة	الإجراءات
Ahmad	nawiw44739@akixpres.com	محطط	نشط	تعديل أرشفة
khaled	catet56823@atinjo.com	مدير	نشط	تعديل أرشفة
Omar	Omar2929745@gmail.com	محطط	نشط	تعديل أرشفة

© 2024 نظام إدارة النفايات. جميع الحقوق محفوظة.

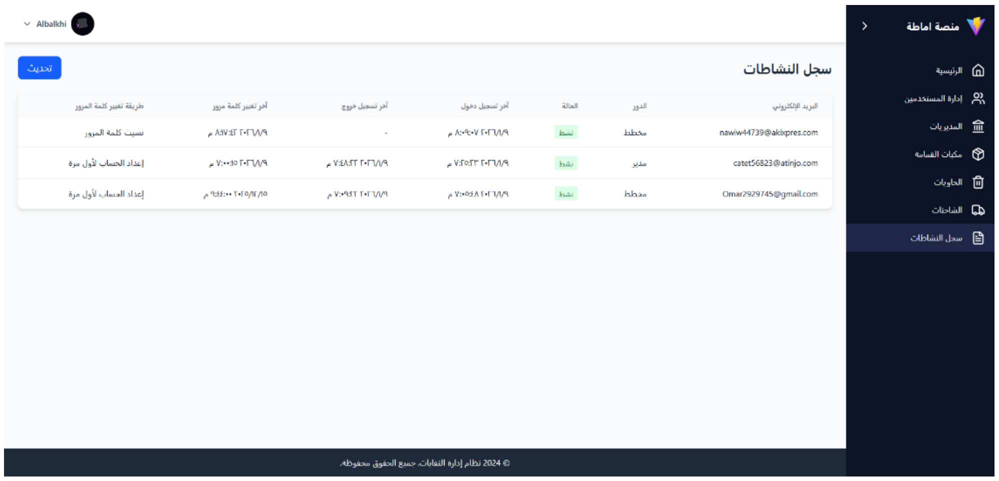
Figure 54 User Management - User Archiving



إدارة المستخدمين

اسم المستخدم	البريد الإلكتروني	الدور	الحالة
Ahmad	nawiw44739@akixpres.com	مخطط	غير نشط

Figure 55 User Managment - Apply Filtering



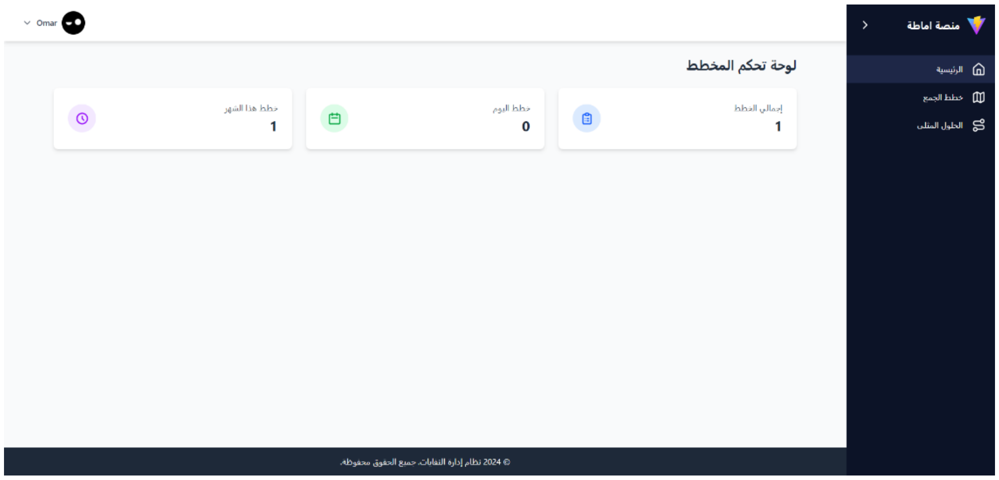
سجل النشاطات

البريد الإلكتروني	الدور	الحالة	آخر تسجيل دخول	آخر تسجيل خروج	آخر تغيير كلمة مرور	طريقة تغيير كلمة المرور
nawiw44739@akixpres.com	مخطط	نشط	٢٠٢٦/١/٩ ٨:٠٩:٠٧ م	-	٢٠٢٦/١/٩ ٨:١٧:٤٢ م	نسيت كلمة المرور
catet56823@atinjo.com	مدير	نشط	٢٠٢٦/١/٩ ٧:٢٥:٢٣ م	٢٠٢٦/١/٩ ٧:٤٨:٢٢ م	٢٠٢٦/١/٩ ٧:٠٠:١٥ م	إعداد الحساب لأول مرة
Omar2929745@gmail.com	مخطط	نشط	٢٠٢٦/١/٩ ٧:٠٥:٤٨ م	٢٠٢٦/١/٩ ٧:٠٩:٢٢ م	٢٠٢٥/١٢/١٥ ٩:٤٤:٠٠ م	إعداد الحساب لأول مرة

Figure 56 Activity Log Interface

Note: Infrastructure management interfaces follow the same CRUD patterns as User Management shown in Figure (50 – 51 – 52 – 53). However, features like **User Archiving** and **Advanced Search** are specific to the Accounts module and do not apply to physical assets.

5.3 Planner User Interface

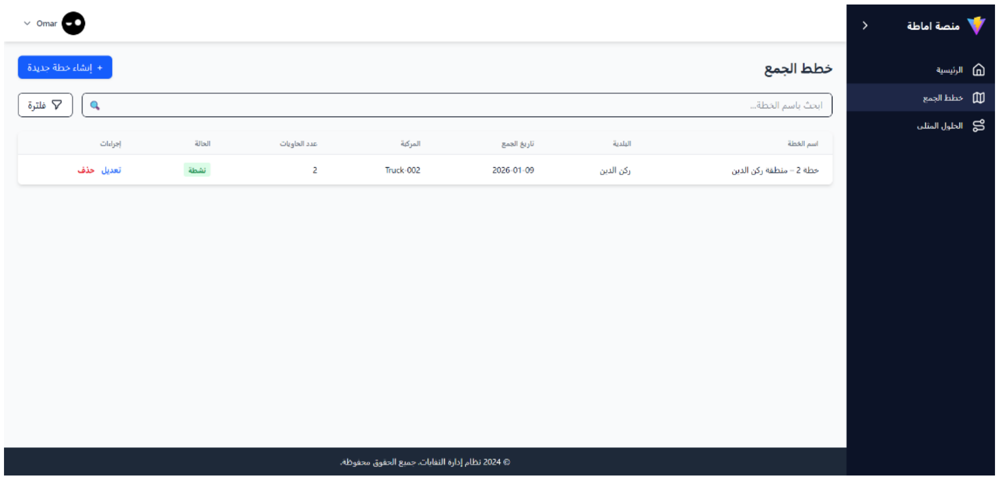


لوحة تحكم المخطط

إجمالي الخطط	خطط اليوم	خطط هذا الشهر
1	0	1

© 2024 نظام إدارة النفايات. جميع الحقوق محفوظة.

Figure 57 Planner Dashboard Interface

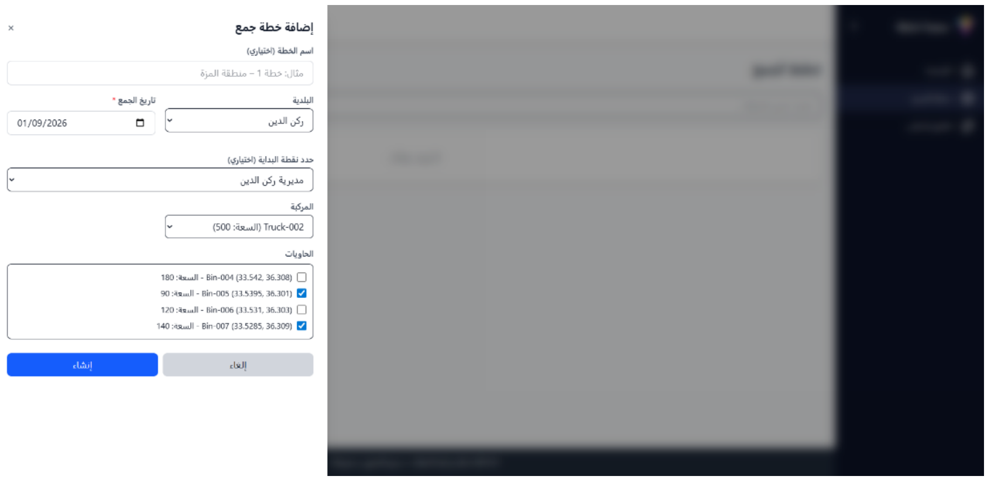


خطط الجمع

اسم الخطة	البلدية	تاريخ الجمع	المركبة	عدد الحاويات	الحالة	إجراءات
خطة 2 – منطقة ركن الدين	ركن الدين	2026-01-09	Truck-002	2	نشطة	تعديل حذف

© 2024 نظام إدارة النفايات. جميع الحقوق محفوظة.

Figure 58 Plans Management - Main Page



إضافة خطة جمع ×

اسم الخطة (اختياري): مثال: خطة 1 – منطقة المزة

البلدية: ركن الدين

تاريخ الجمع *: 01/09/2026

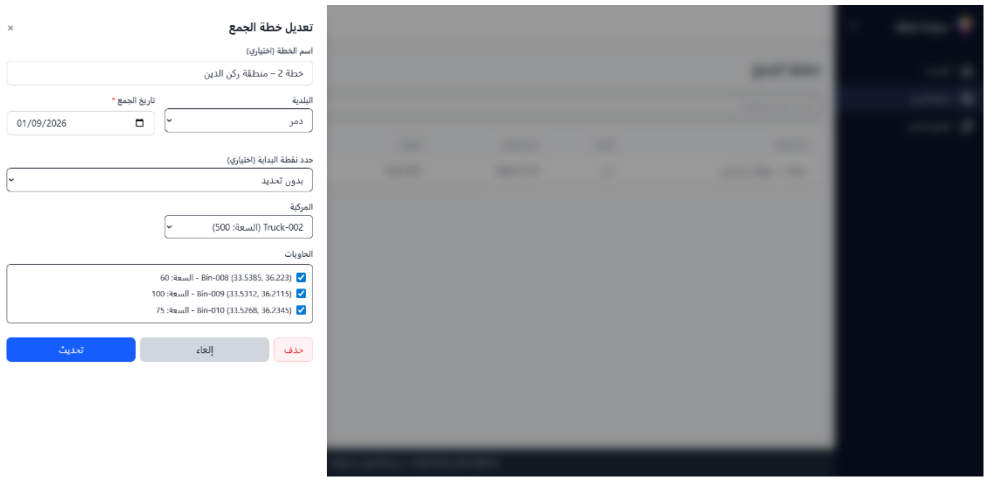
حدد نقطة البداية (اختياري): مديرية ركن الدين

المركبة: Truck-002 (السعة: 500)

الحاويات:

- [ ] Bin-004 (33.542, 36.308) - السعة: 180
- [x] Bin-005 (33.5395, 36.301) - السعة: 90
- [ ] Bin-006 (33.531, 36.303) - السعة: 120
- [x] Bin-007 (33.5285, 36.309) - السعة: 140

Figure 59 Plans Management - Craete Plan



تعديل خطة الجمع ×

اسم الخطة (اختياري): خطة 2 – منطقة ركن الدين

البلدية: دمر

تاريخ الجمع *: 01/09/2026

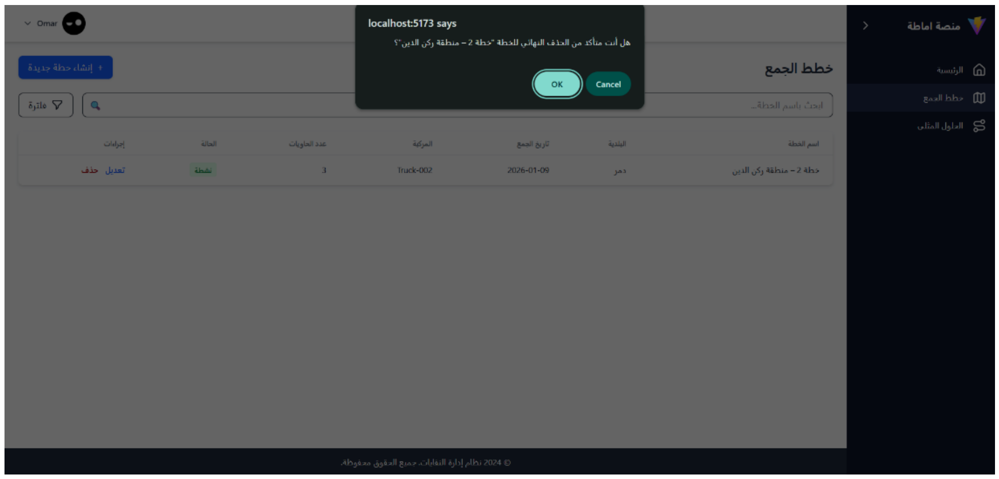
حدد نقطة البداية (اختياري): بدون تحديد

المركبة: Truck-002 (السعة: 500)

الحاويات:

- [x] Bin-008 (33.5385, 36.223) - السعة: 60
- [x] Bin-009 (33.5312, 36.2115) - السعة: 100
- [x] Bin-010 (33.5268, 36.2345) - السعة: 75

Figure 60 Plans Management - Update Plan



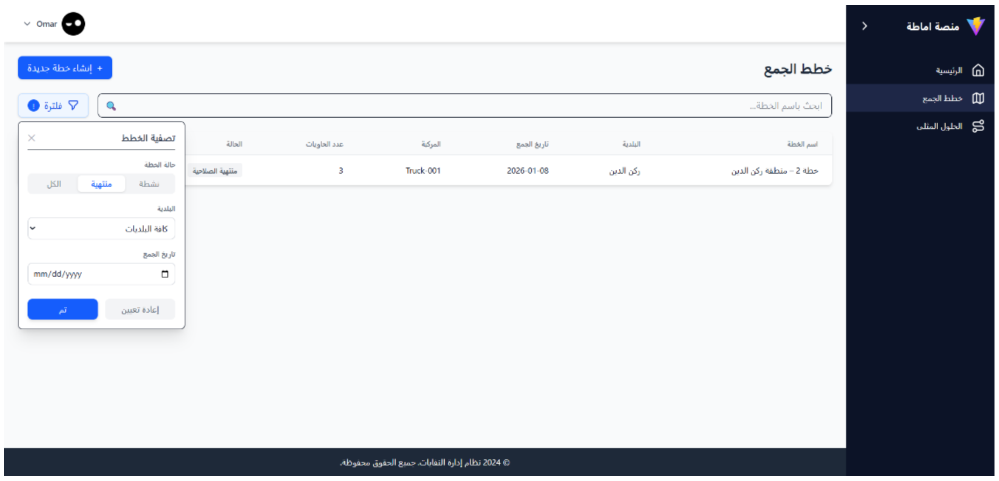
localhost:5173 says

هل أنت متأكد من الحذف النهائي للخطة "خطة 2 – منطقة ركن الدين"؟

خطط الجمع

اسم الخطة	البلدية	تاريخ الجمع	المركبة	عدد الحاويات	الحالة	إجراءات
خطة 2 – منطقة ركن الدين	دمر	2026-01-09	Truck-002	3	نشطة	تعديل حذف

Figure 61 Plans Management - Delete Plan



خطط الجمع

اسم الخطة	البلدية	تاريخ الجمع	المركبة	عدد الحاويات	الحالة
خطة 2 – منطقة ركن الدين	ركن الدين	2026-01-08	Truck-001	3	منتهية الصلاحية

Figure 62 Plans Management - Apply Filtering

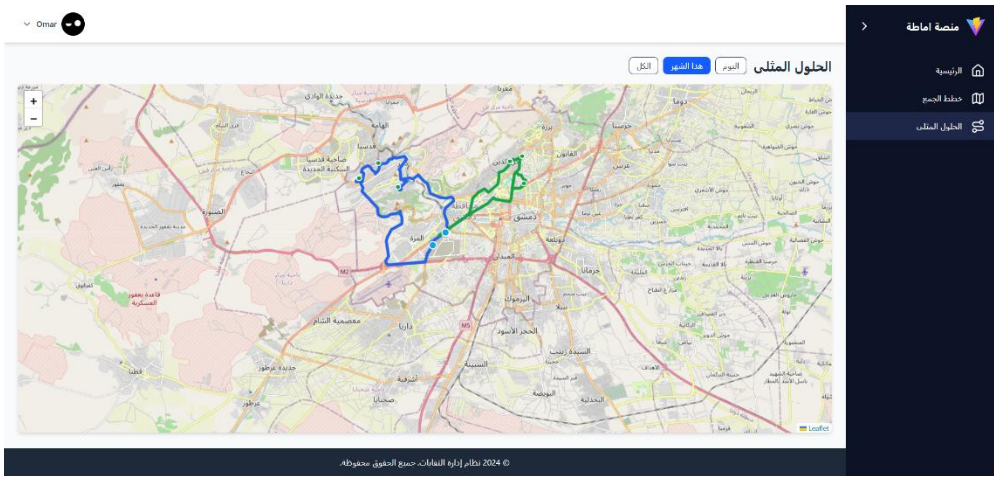


Figure 63 Plans Management - Optimal Solution

6. Summary

This chapter presented the architectural and structural design of the Route Optimization System, including the three-tier architecture, database schema, and user interface design.

These specifications form the foundation for the implementation phase, which will be covered in Chapter 5.

Chapter 6 Implementation

1. Introduction

This chapter documents the actual construction and realization of the Route Optimization System defined in the previous design chapters. It bridges the gap between the theoretical design and the functional software product. The implementation process focuses on three main pillars: establishing the development environment, coding the core **Optimization Engine** using the Vehicle Routing Problem (VRP) algorithms, and building the **User Interface** that allows planners to interact with these complex algorithms intuitively. Furthermore, this chapter discusses the integration challenges, specifically connecting the Django backend with the OSRM routing service and the React frontend, ensuring a seamless data flow across the system's three-tier architecture.

2. Development Environment & Tools

The system implementation relies on a set of modern tools and frameworks selected for their performance, community support, and compatibility with the project's architectural requirements. The key technologies include:

- **Backend Development**:
  - **Python** (3.10+): Selected as the core programming language for its rich ecosystem of data science and optimization libraries.
  - **Django & Django REST Framework**: Used to build the secure API structure, managing database interactions and request handling.
  - **SimpleJWT**: Implemented to secure API endpoints using stateless JSON Web Token authentication.
- **Frontend Development:**
  - **React.js:** Used to build the Single Page Application (SPA), providing a reactive and component-based user interface.

  - **Vite:** Utilized as the build tool for faster development cycles and optimized production bundling.
  - **Leaflet:** Integrated for rendering interactive maps and visualizing geospatial data (routes and markers) efficiently.

- **Optimization & Geospatial Engines:**
  - **Google OR-Tools:** The core engine used to model and solve the Vehicle Routing Problem (VRP).
  - **OSRM** (Open-Source Routing Machine): A high-performance routing engine used to calculate real-world travel distances and fetch route geometries (polylines).
  - **PostgreSQL:** The relational database system chosen for its robustness and ability to handle complex relational queries.

3. Core Algorithms Implementation

The heart of the system is the Optimization Engine, which translates business requirements (bins, vehicles, locations) into mathematical models. This section details the implementation of the "VRPSolver" class and its integration with OSRM.

3.1 Distance Matrix Calculation (OSRM Integration)

The VRP algorithm requires a distance matrix representing the travel cost between every pair of points (Depot and Bins). We implemented the "OSRMService" class to fetch this data dynamically.

Instead of using Euclidean (straight-line) distance, which is inaccurate for city navigation, we query the OSRM API. The "get_distance_matrix" method constructs a query string with all location coordinates and parses the returned

JSON matrix. This ensures the optimization creates routes based on actual road networks.

3.2 Solving the Vehicle Routing Problem (Google OR-Tools)

The VRPSolver class orchestrates the optimization process. The implementation follows a structured pipeline:

1. **Data Preparation:** The `_prepare_locations` method aggregates the vehicle's starting point (Depot) and all active bins into a single location list.
2. **Model Setup:** We utilize the "`pywrapcp.RoutingModel`". A crucial step is defining the **Capacity Constraint** using the "`AddDimension`" method. This ensures that the total waste collected by the vehicle does not exceed its maximum capacity.
3. **Search Strategy:** To find a high-quality solution within a reasonable time, we configured the solver to use **Guided Local Search (GLS)** as the metaheuristic strategy. This allows the solver to escape local optima and explore the solution space effectively.

3.3 Route Geometry & Visualization

A key feature of our system is visualizing the actual path the truck should take, not just the sequence of stops. We extended the "OSRMService" with a "`get_route_geometry`" method. Once the VRP solver determines the optimal sequence of bins, this method sends the ordered coordinates to OSRM with the parameter geometries=polyline. The returned encoded string represents the exact path along the road network, which is then decoded by the frontend (Leaflet) to draw the route on the map.

4. Backend Services Implementation

Beyond the core optimization logic, the backend acts as the secure gateway and data manager for the entire system. This section details the implementation of security protocols and the structural design of the RESTful API.

4.1 User Authentication & Security

To ensure system integrity and restrict access to authorized personnel (Admins and Planners), we implemented a robust authentication mechanism combining **JSON Web Tokens (JWT)** and **One-Time Passwords (OTP)**.

- **JWT Implementation:** We utilized the SimpleJWT library to handle stateless authentication. Upon successful login, the server issues an "`access_token`" (short-lived) for API requests and a "`refresh_token`" to maintain the session without storing sensitive data on the client side.
- **OTP Service:** As an added security layer for critical actions (like account activation or password resets), we implemented a custom "`OTPService`" in accounts/services.py. This service generates a cryptographically secure 6-digit code and dispatches it via email using the "`EmailService`".
- **Role-Based Access Control (RBAC):** We extended Django's permission classes to create custom decorators such as "`IsAdmin`" and "`IsPlanner`". This ensures, for example, that only Admins can archive users, while Planners are restricted to scenario management.

4.2 API Structure & Data Management

To maintain code scalability and adhere to the **DRY (Don't Repeat Yourself)** principle, the API architecture follows a strict separation of concerns using Django REST Framework's **`ViewSets`**.

- **`ModelViewSets`:** Instead of writing separate functions for each HTTP method (GET, POST, PUT, DELETE), we utilized `ModelViewSet` for standard CRUD operations. For instance, the `UserViewSet` automatically handles listing, creating, and updating user records, significantly reducing boilerplate code.
- **Service Layer Pattern:** To keep the "Views" clean (Thin Views), complex business logic was moved to a dedicated service layer. For example, the logic for User Archiving is encapsulated in `UserService.archive_user()` rather than embedding it directly in the API endpoint.
- **Serialization & Validation:** We implemented strict Serializers (e.g., `UserSerializer`, `ScenarioSerializer`) to validate incoming data before it touches the database. This prevents SQL injection attacks and ensures data consistency (e.g., verifying that a user's email format is correct before creation).

5. Frontend Implementation & Integration

The user interface is constructed as a Single Page Application (SPA) using React.js powered by Vite. This modern stack ensures a responsive experience, allowing planners to interact with complex maps and large datasets without page reloads. The frontend integration focuses on two main aspects: secure data consumption and geospatial visualization.

5.1 API Consumption & State Management

To facilitate seamless communication between the React client and the Django backend, we established a centralized networking layer using the Axios library.

- **Interceptors**: As implemented in "`src/services/api.js`", we configured request interceptors to automatically attach the Authorization: Bearer <token> header to every outgoing request. This design abstracts the authentication logic, ensuring that all protected endpoints (like creating scenarios) are accessed securely without repeating code in every component.
- **Global State**: To manage the user's session and application-wide data, we utilized a global store (implemented in "`src/store/authStore.js`"). This ensures that the user's login state and permissions persist across different routes and page refreshes.

5.2 Interactive Map Visualization (Leaflet)

The core feature for the planner is the ability to visualize routes on a map. This was implemented in "`src/components/MapView.jsx`" using the **React-Leaflet** library.

- **Dynamic Layers:** The map component dynamically renders different data layers based on the current context:
  - **Infrastructure Layer:** Bins and Depots are rendered as Marker components with custom icons.
  - **Route Layer:** The optimized route geometry (polyline) fetched from the backend is rendered as a Polyline component. This allows the planner to see the exact path the truck will take along the road network, rather than just straight lines between points.
- **Interactivity:** The map supports interactive features such as zooming, panning, and popups that display detailed information (e.g., bin fill levels) when an asset is clicked.

6. Summary

This chapter detailed the technical realization of the Route Optimization System. We successfully established a robust development environment, implemented the core VRP algorithms using Google OR-Tools and OSRM, and built a secure, service-oriented backend architecture. Finally, these components were integrated into a responsive React frontend with advanced mapping capabilities. The resulting implementation effectively transforms raw geospatial data into actionable, optimized waste collection plans.

Chapter 7 Testing and Validation

1. Introduction

Testing is a fundamental phase in the software development lifecycle, aiming to verify that the system complies with the specified requirements and to validate that it meets the business needs. In this project, we adopted a structured testing approach that begins with defining test cases, tracing them back to the functional requirements (RTM), and finally executing them to record the results.

2. Test cases definition:

In this section, we define the detailed test scenarios designed to verify the system's compliance with the functional requirements

2.1 Test cases definition – Common User

Test Case Scenario:		**Sce-01: Check Activate Account Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-01	Activate account with valid data	1. Access the system website on the account activation page. 2. Enter the email address previously created by the administrator. 3. Click "Send Verification Code." 4. Enter the received verification code (OTP) and a valid password. 5. Confirm the password and click "Activate Account."	Account is activated and user is logged in
TC-02	Activate account with unregistered email	1. Access the system website on the account activation page. 2. Enter an incorrect or unregistered email address 3. Click "Send Verification Code."	Error message is displayed

TC-03	Activate account with invalid OTP	1. Perform the activation flow up to receiving the OTP. 2. Enter an invalid verification code. 3. Click "Activate Account."	Error message is displayed
TC-04	Activate account with weak password	1. Perform the activation flow up to the password entry. 2. Enter a password that does not meet system requirements. 3. Click "Activate Account."	Password requirements are displayed

Test Case Scenario:		**Sce-02: Check login Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-05	Login with valid credentials	1. Access the login page. 2. Enter a valid email address and password. 3. Click the "Login" button.	User is logged in and redirected to home page
TC-06	Login with invalid credentials	1. Access the login page. 2. Enter an incorrect email or password. 3. Click the "Login" button.	Error message is displayed

Test Case Scenario:		**Sce-03: Check Forget Password Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-07	Reset password with valid data	1. Navigate to the Forgot Password page. 2. Enter the registered email address and click "Send Reset Code." 3. Enter the received reset code (OTP) and a new password. 4. Confirm the password and click "Reset Password."	Password is reset successfully
TC-08	Reset password with	1. Navigate to the Forgot Password page. 2. Enter an incorrect or unregistered email address.	Error message is displayed

	unregistered email	3. Click "Send Reset Code."	
TC-09	Check results on entering an invalid Reset Code	1. Perform the reset flow up to receiving the OTP. 2. Enter an invalid reset code. 3. Click "Reset Password."	The system displays an error message.
TC-10	Check results on entering a weak password	1. Perform the reset flow up to the password entry. 2. Enter a weak password. 3. Click "Reset Password."	The system displays password requirements.

Test Case Scenario:		**Sce-04: Check Edit Profile Functionality.**	
Test Case ID	**Test Case Title**	**Test Steps**	**Expected Result**
TC-11	Update profile with valid data	1. Navigate to the Edit Profile page from the home page. 2. Edit the profile information (profile photo, phone number, username). 3. Click "Save."	Profile is updated successfully
TC-12	Update profile with invalid data	1. Navigate to the Edit Profile page. 2. Enter invalid data or leave required fields empty. 3. Click "Save."	Error message is displayed.

Test Case Scenario:		**Sce-05: Check Change password Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-13	Change password with valid data	1. Navigate to the Change Password page from account settings. 2. Enter the current password, the new password, and confirm it. 3. Click "Save."	Password is updated successfully

TC-14	Change password with incorrect current password	1. Navigate to the Change Password page. 2. Enter an incorrect current password. 3. Click "Save."	Error message is displayed
TC-15	Change password with weak or mismatched password	1. Navigate to the Change Password page. 2. Enter a weak new password or a confirmation that does not match. 3. Click "Save."	Password requirements or mismatch error is displayed

Test Case Scenario:		**Sce-06: Check logout Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-16	Logout	1. Click the "Logout" option.	User is logged out and redirected to login page

Table 24 Test Cases Definition - Common User

2.2 Test cases definition – Admin User

Test Case Scenario:		**Sce-07: Check Create User Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-17	Create user with valid data	1. Navigate to the Users Management page and select "Add New User." 2. Enter valid username, email address, role, phone number, and profile picture. 3. Click "Create."	User is created successfully

TC-18	Create user with invalid data	1. Navigate to the Users Management page and select "Add New User." 2. Enter invalid data (e.g., incorrect email format) or leave required fields empty. 3. Click "Create."	Error message is displayed
TC-19	Create user with duplicate data	1. Navigate to the Users Management page and select "Add New User." 2. Enter an email address or username that already exists. 3. Click "Create."	Duplicate data error is displayed

Test Case Scenario:		**Sce-08: Check Edit User Functionality.**	
Test Case ID	**Test Case Title**	**Test Steps**	**Expected Result**
TC-20	Edit user with valid data	1. Navigate to the Users Management page and select an existing user. 2. Update the required fields (e.g., role, phone number, profile picture). 3. Click "Save."	User details are updated successfully
TC-21	Edit user with invalid data	1. Navigate to the Users Management page and select an existing user. 2. Enter invalid data or clear required fields. 3. Click "Save."	Error message is displayed
TC-22	Edit user with duplicate data	1. Navigate to the Users Management page and select an existing user. 2. Update the email address or username to one that already exists. 3. Click "Save."	Duplicate data error is displayed

Test Case Scenario:		**Sce-09: Check Edit User Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-23	Archive active user	1. Navigate to the Users Management page. 2. Click "Archive" for a selected active user. 3. Confirm the archive action.	User is archived successfully
TC-24	Cancel user archive	1. Navigate to the Users Management page. 2. Click "Archive" for a selected active user. 3. Cancel the confirmation.	User remains active

Test Case Scenario:		**Sce-10: Check Edit User Functionality.**	
Test Case ID	**Test Case Title**	**Test Steps**	**Expected Result**
TC-25	Restore archived user	1. Navigate to the Users Management page and filter to display Archived Users. 2. Select an archived user and choose the "Restore" action. 3. Confirm the restore action.	User is restored to active status
TC-26	Cancel user restore	1. Navigate to the Users Management page and select an archived user. 2. Choose the "Restore" action. 3. Select "Cancel."	No changes are applied

Test Case Scenario:		**Sce-11: Check View Users List Functionality.**	
Test Case ID	**Test Case Title**	**Test Steps**	**Expected Result**
TC-27	View users list	1. Navigate to the Users Management page.	Users list is displayed
TC-28	Search users by username	1. Navigate to the Users Management page. 2. Enter a keyword (username) in the search field.	Matching users are displayed
TC-29	Filter users list	1. Navigate to the Users Management page. 2. Select one or more filter criteria (e.g., role, status).	Filtered users are displayed

Test Case Scenario:		**Sce-12: Check Manage Vehicles Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-30	Add vehicle with valid data	1. Navigate to the Vehicle Management page. 2. Select "Add Vehicle." 3. Enter valid vehicle data and click "Save."	Vehicle is added successfully
TC-31	Edit vehicle details	1. Select an existing vehicle and choose "Edit." 2. Update the data and click "Save."	Vehicle details are updated
TC-32	Delete vehicle	1. Select an existing vehicle and choose "Delete." 2. Confirm the deletion action.	Vehicle is deleted successfully
TC-33	Validate vehicle data	1. Attempt to Add or Edit a vehicle with missing or invalid data. 2. Click "Save."	Error message is displayed

Test Case Scenario:		**Sce-13: Check Manage Bins Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-34	Add bin with valid data	1. Navigate to the Bins Management page. 2. Select "Add Bin." 3. Enter bin name, GPS coordinates, capacity, and municipality. 4. Click "Save."	Bin is added successfully
TC-35	Edit bin details	1. Select a bin from the list and click "Edit." 2. Update the bin details (e.g., capacity) and click "Save."	Bin details are updated
TC-36	Delete bin	1. Select a bin and click "Delete." 2. Confirm the deletion.	Bin is deleted successfully
TC-37	Validate bin data	1. Attempt to Add or Edit a bin with invalid input. 2. Click "Save."	Error message is displayed

Test Case Scenario:		**Sce-14: Check Manage Municipalities Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-38	Add municipality with valid data	1. Navigate to the Municipalities Management page. 2. Select "Add Municipality." 3. Enter name and center coordinates. 4. Click "Save."	Municipality is added successfully
TC-39	Edit municipality details	1. Select a municipality from the list and click "Edit." 2. Update the name or coordinates and click "Save."	Municipality details are updated
TC-40	Municipality is deleted successfully	1. Select a municipality with no linked assets (bins/vehicles) and click "Delete." 2. Confirm the deletion.	Delete municipality without dependencies.
TC-41	Error message is displayed	1. Attempt to Add or Edit a municipality with invalid input (e.g. duplicate name). 2. Click "Save."	Validate municipality data

Test Case Scenario:		**Sce-15: Check Manage Landfills Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-42	Add landfill with valid data	1. Navigate to the landfill management page. 2. Select "Add Landfill." 3. Enter name, GPS coordinates, and select associated municipalities. 4. Click "Save."	Landfill is added successfully
TC-43	Edit landfill details	1. Select a landfill from the list and click "Edit." 2. Modify the details and click "Save."	Landfill details are updated

TC-44	Delete landfill	1. Select a landfill and click "Delete." 2. Confirm the deletion.	Landfill is deleted successfully

Table 25 Test Cases Definition - Admin User

2.3 Test Cases Definition - Planner User

Test Case Scenario:		**Sce-16: Check Create Collection Scenario Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-45	Create collection scenario	1. Go to "Collection Plans" page and click "Add Plan." 2. Select municipality and collection date. 3. Select a suitable vehicle from the filtered list. 4. Select target Bins and click "Save."	Scenario is created successfully
TC-46	Create scenario with busy resource	1. Attempt to select a vehicle or bin that is already assigned on the selected date. 2. Click "Save."	Resource conflict error is displayed

Test Case Scenario:		**Sce-17: Check View & Filter Collection Plans Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-47	View collection plans	1. Navigate to the Scenarios management page.	Collection plans list is displayed
TC-48	Search collection plans	2. Navigate to the Scenarios management page. 3. Enter a keyword (e.g., "Midan Zone") in the search field.	Matching plans are displayed
TC-49	Filter plans by status	1. Navigate to the Scenarios management page. 2. Select "Archived" from the status filter.	Archived plans are displayed

TC-50	No plans found	1. Apply search or filter criteria that return no matching scenarios.	No plan's found message is displayed

Test Case Scenario:		**Sce-18: Check Update Scenario Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-51	Update scenario	1. Select "Edit" for a specific scenario. 2. Modify scenario inputs (e.g., add boxes or change assigned vehicle). 3. Click "Save."	Scenario is updated successfully
TC-52	Update scenario without permission	1. Attempt to edit a scenario created by another user.	Access denied error is returned

Test Case Scenario:		**Sce-19: Check Update Scenario Functionality.**	
Test Case ID	Test Case Title	Test Steps	Expected Result
TC-53	Delete scenario	1. Select "Delete" for a specific scenario. 2. Confirm the deletion warning.	Scenario is deleted successfully
TC-54	Delete scenario without permission	1. Attempt to delete a scenario created by another user.	Access denied error is returned

Table 26 Test Cases Definition - Planner User

3. Test cases execution

Test Case ID	Test Case Title	Tested Data	Expected Result	Actual Result	Pass /Fail
TC-01	Activate Account + Valid Data	Valid Email, Valid OTP, Strong Password	The system validates data, activates account, and logs user in.	The system validates data, activates account, and logs user in.	Pass
TC-02	Activate Account + Incorrect Email	Invalid/Unregistered Email	The system displays an error message.	The system displays an error message.	Pass
TC-03	Activate Account + Invalid OTP	Valid Email, Invalid OTP	The system displays an error message.	The system displays an error message.	Pass
TC-04	Activate Account + Weak Password	Valid Email, Valid OTP, Weak Password	The system displays password requirements.	The system displays password requirements.	Pass
TC-05	Login + Valid Credentials	Valid Email, Valid Password	The system validates credentials and redirects to main page.	The system validates credentials and redirects to main page.	Pass
TC-06	Login + Invalid Credentials	Incorrect Email or Password	The system displays an error message.	The system displays an error message.	Pass

TC-07	Reset Password + Valid Data	Valid Email, Valid OTP, New Strong Password	The system validates data and updates password successfully.	The system validates data and updates password successfully.	Pass
TC-08	Reset Password + Incorrect Email	Unregistered Email	The system displays an error message.	The system displays an error message.	Pass
TC-09	Reset Password + Invalid OTP	Valid Email, Invalid OTP	The system displays an error message.	The system displays an error message.	Pass
TC-10	Reset Password + Weak Password	Valid Email, Valid OTP, Weak Password	The system displays password requirements.	The system displays password requirements.	Pass
TC-11	Edit Profile + Valid Informatio n	New Username, New Phone Number	The system validates info and saves changes successfully.	The system validates info and saves changes successfully.	Pass
TC-12	Edit Profile + Invalid Data	Empty Required Fields	The system displays error message for invalid/missing fields.	The system displays error message for invalid/missing fields.	Pass
TC-13	Change Password + Valid Data	Current Password, New Strong Password	The system validates data and updates password successfully.	The system validates data and updates password successfully.	Pass

TC-14	Change Password + Incorrect Old Pass	Incorrect Current Password	The system displays an error message.	The system displays an error message.	Pass
TC-15	Change Password + Weak New Pass	Weak New Password	The system displays password requirements or mismatch error.	The system displays password requirements or mismatch error.	Pass
TC-16	Logout + Logged In User	N/A	The system terminates session and redirects to login page[16].	The system terminates session and redirects to login page.	Pass
TC-17	Create User + Valid Data	Valid Username, Email, Role, Phone	The system creates user, displays success message, updates list[17].	The system creates user, displays success message, updates list.	Pass
TC-18	Create User + Invalid Data	Invalid Email Format	The system displays descriptive error messages[18].	The system displays descriptive error messages.	Pass
TC-19	Create User + Duplicate Data	Existing Email or Username	The system prevents creation and notifies administrator[19].	The system prevents creation and notifies administrator.	Pass
TC-20	Edit User + Valid Data	Updated Role, Updated Phone	The system saves changes, displays success message, updates list[20].	The system saves changes, displays success message, updates list.	Pass

TC-21	Edit User + Invalid Data	Empty Required Fields	The system displays descriptive error messages.	The system displays descriptive error messages.	Pass
TC-22	Edit User + Duplicate Data	Existing Email or Username	The system prevents saving and notifies administrator.	The system prevents saving and notifies administrator.	Pass
TC-23	Archive User + Active User	Selected Active User	The system archives user and updates list with success message.	The system archives user and updates list with success message.	Pass
TC-24	Archive User + Cancel Action	Selected Active User	The system keeps the user active.	The system keeps the user active.	Pass
TC-25	Restore User + Archived User	Selected Archived User	The system changes status to active and displays success message.	The system changes status to active and displays success message.	Pass
TC-26	Restore User + Cancel Action	Selected Archived User	The system cancels restore operation without changes.	The system cancels restore operation without changes.	Pass
TC-27	View Users + All Users	N/A	The system displays paginated list of all users.	The system displays paginated list of all users.	Pass
TC-28	Search Users + Keyword	Username Keyword	The system filters list and displays matching results.	The system filters list and displays matching results.	Pass

TC-29	Filter Users + Criteria	Selected Role or Status	The system updates list to display matching users.	The system updates list to display matching users.	Pass
TC-30	Add Vehicle + Valid Data	Vehicle Name, Capacity, Coordinates	The system saves vehicle, displays success message, updates list.	The system saves vehicle, displays success message, updates list.	Pass
TC-31	Edit Vehicle + Valid Data	Updated Vehicle Details	The system saves changes and updates vehicles list[31].	The system saves changes and updates vehicles list.	Pass
TC-32	Delete Vehicle + Existing Vehicle	Selected Vehicle	The system deletes vehicle and updates vehicles list[32].	The system deletes vehicle and updates vehicles list.	Pass
TC-33	Manage Vehicle + Validation Error	Missing/Invalid Vehicle Data	The system displays descriptive error messages[33].	The system displays descriptive error messages.	Pass
TC-34	Add Bin + Valid Data	Bin Name, GPS, Capacity, Municipality	The system creates bin record, displays success message, updates list[34].	The system creates bin record, displays success message, updates list.	Pass
TC-35	Edit Bin + Valid Data	Updated Capacity	The system updates bin record successfully[35].	The system updates bin record successfully.	Pass

TC-36	Delete Bin + Existing Bin	Selected Bin	The system permanently deletes bin and updates list[36].	The system permanently deletes bin and updates list.	Pass
TC-37	Manage Bin + Invalid Data	Invalid Input	The system displays an error message[37].	The system displays an error message.	Pass
TC-38	Add Municipality + Valid Data	Name, Center Coordinates	The system creates record, displays success message, updates list[38].	The system creates record, displays success message, updates list.	Pass
TC-39	Edit Municipality + Valid Data	Updated Name or Coordinates	The system updates municipality record successfully[39].	The system updates municipality record successfully.	Pass
TC-40	Delete Municipality + No Dependencies	Municipality (No Bins/Vehicles)	The system deletes municipality and updates list[40].	The system deletes municipality and updates list.	Pass
TC-41	Manage Municipality + Validation Error	Duplicate Name / Invalid Coordinates	The system displays an error message[41].	The system displays an error message.	Pass

TC-42	Add Landfill + Valid Data	Name, GPS, Associated Municipalities	The system creates record, displays success message, updates list[42].	The system creates record, displays success message, updates list.	Pass
TC-43	Edit Landfill + Valid Data	Modified Details	The system updates landfill record successfully[43].	The system updates landfill record successfully.	Pass
TC-44	Delete Landfill + Existing Landfill	Selected Landfill	The system deletes landfill record and updates list[44].	The system deletes landfill record and updates list.	Pass
TC-45	Create Scenario + Valid Data	Municipality, Date, Vehicle, Bins	The system generates name and saves scenario as unsolved[45].	The system generates name and saves scenario as unsolved.	Pass
TC-46	Create Scenario + Busy Resource	Vehicle/Bin assigned on same date	The system prevents saving and displays error message[46].	The system prevents saving and displays error message.	Pass
TC-47	View Scenarios + Default View	N/A	The system displays paginated list of scenarios[47].	The system displays paginated list of scenarios.	Pass
TC-48	Search Scenarios + Keyword	Scenario Name Keyword	The system filters list and displays matching scenarios[48].	The system filters list and displays matching scenarios.	Pass

TC-49	Filter Scenarios + Archived Status	Status: Archived	The system displays scenarios with past collection dates[49].	The system displays scenarios with past collection dates.	Pass
TC-50	View Scenarios + No Results	Non-matching Filter Criteria	The system displays a "No plans found" message.	The system displays a "No plans found" message.	Pass
TC-51	Update Scenario + Valid Data	Modified Bins or Vehicle	The system deletes old solutions and saves updated details.	The system deletes old solutions and saves updated details.	Pass
TC-52	Update Scenario + Permission Denied	Scenario created by another user	The system blocks request and returns access denied (403).	The system blocks request and returns access denied (403).	Pass
TC-53	Delete Scenario + Valid Data	Selected Scenario	The system deletes scenario and related RouteSolutions.	The system deletes scenario and related RouteSolutions.	Pass
TC-54	Delete Scenario + Permission Denied	Scenario created by another user	The system blocks action and denies access.	The system blocks action and denies access.	Pass

Table 27 Test Cases Execution

4. Requirements Traceability Matrix – RTM

Req_ID	Use cases	analysis	Detailed design	coding	Test cases
RE-FR-AM-01	UC-07	analysis	design	source	TC-17 TC-18 TC-19
RE-FR-AM-02	UC-08	analysis	design	source	TC-20 TC-21 TC-22
RE-FR-AM-03	UC-11	analysis	design	source	TC-27
RE-FR-AM-04	UC-09	analysis	design	source	TC-23 TC-24
RE-FR-AM-05	UC-10	analysis	design	source	TC-25 TC-26
RE-FR-AM-06	UC-11	analysis	design	source	TC-28
RE-FR-AM-07	UC-11	analysis	design	source	TC-29
RE-FR-AM-08	UC-02	analysis	design	source	TC-05 TC-06
RE-FR-AM-09	UC-03	analysis	design	source	TC-07 TC-08 TC-09 TC-10
RE-FR-AM-10	UC-01	analysis	design	source	TC-01 TC-02 TC-03 TC-04
RE-FR-AM-11	UC-04	analysis	design	source	TC-11 TC-12
RE-FR-AM-12	UC-05	analysis	design	source	TC-13 TC-14 TC-15
RE-FR-AM-13	UC-06	analysis	design	source	TC-16

RE-FR-IM-15	UC-14	analysis	design	source	TC-38
RE-FR-IM-16	UC-14	analysis	design	source	TC-39
RE-FR-IM-17	UC-14	analysis	design	source	TC-38
RE-FR-IM-18	UC-14	analysis	design	source	TC-40
RE-FR-IM-19	UC-15	analysis	design	source	TC-42
RE-FR-IM-20	UC-15	analysis	design	source	TC-43
RE-FR-IM-21	UC-15	analysis	design	source	TC-42
RE-FR-IM-22	UC-15	analysis	design	source	TC-44
RE-FR-IM-23	UC-12	analysis	design	source	TC-30
RE-FR-IM-24	UC-12	analysis	design	source	TC-31
RE-FR-IM-25	UC-12	analysis	design	source	TC-30
RE-FR-IM-26	UC-12	analysis	design	source	TC-32
RE-FR-IM-27	UC-13	analysis	design	source	TC-34
RE-FR-IM-28	UC-13	analysis	design	source	TC-35
RE-FR-IM-29	UC-13	analysis	design	source	TC-34
RE-FR-IM-30	UC-13	analysis	design	source	TC-36
RE-FR-PM-31	UC-16	analysis	design	source	TC-45 TC-46
RE-FR-PM-32	UC-16	analysis	design	source	TC-46
RE-FR-PM-33	UC-18	analysis	design	source	TC-51 TC-52
RE-FR-PM-34	UC-17	analysis	design	source	TC-47
RE-FR-PM-35	UC-17	analysis	design	source	TC-48
RE-FR-PM-36	UC-17	analysis	design	source	TC-49 TC-50
RE-FR-PM-37	UC-19	analysis	design	source	TC-53 TC-54

Table 28 Requirements Traceability Matrix – RTM

5. Summary

This chapter documented the comprehensive testing strategy adopted to validate the Route Optimization System. It detailed the Test Case Definitions covering all functional modules (Account Management, Infrastructure, and Optimization Scenarios). The testing phase included the execution of 54 distinct test cases, covering both positive and negative flows. The results, as recorded in the Test Execution Log, confirmed a high pass rate, validating the system's stability and adherence to requirements. Finally, the Requirements Traceability Matrix (RTM) established a direct mapping between the initial requirements (Chapter 4) and the verified test cases, ensuring 100% test coverage of the system's core functionalities.

Chapter 8 Conclusion

1. Introduction

This project presented the design and implementation of a **Route Optimization System for Garbage Collection** aimed at modernizing waste management operations in Damascus city. Moving away from traditional, static scheduling, the system introduces a dynamic, algorithmic approach to logistics. By leveraging the power of **Google OR-Tools** for solving the Capacitated Vehicle Routing Problem (CVRP) and integrating **OSRM** for real-world geospatial data, we successfully built a web-based solution that allows municipal planners to generate optimal collection routes, minimize travel distance, and effectively manage fleet and infrastructure assets.

2. Project Achievements

Throughout the development lifecycle, the following key objectives were met:

1. **Algorithmic Efficiency:** Successfully implemented a VRP solver that respects vehicle capacity constraints and optimizes routes based on actual road networks, not just straight-line distances.
2. **Architectural Robustness:** Delivered a scalable **Three-Tier Architecture** (React Frontend, Django Backend, PostgreSQL Database) that ensures separation of concerns and maintainability.
3. **User-Centric Design:** Developed an interactive dashboard allowing planners to visualize bins, vehicles, and routes on a map, transforming complex mathematical results into actionable visual plans.
4. **System Reliability:** Verified the system through comprehensive testing (Unit, Integration, and System Testing), achieving high test coverage and ensuring secure access via JWT authentication.

3. Limitations

While the system achieves its primary goals, certain limitations exist due to project scope and resource constraints:

- **Lack of Real-Time Traffic Data:** The current OSRM integration calculates routes based on static road graph data. It does not account for real-time traffic jams or sudden road closures in Damascus.
- **Manual Fill-Level Entry:** The system currently relies on manual input (or estimation) of waste bin fill levels. It lacks integration with IoT sensors, meaning the "optimal" route is only as accurate as the data entered by the planner.
- **Absence of Driver Application:** The current solution is designed for the *Planner*. There is no dedicated mobile application for the *Driver* to receive navigation instructions turn-by-turn; they must rely on the generated map or printouts.

4. Future Work

To further enhance the system and transition it into a fully smart-city solution, we propose the following future developments:

1. **IoT Integration (Smart Bins):** Installing ultrasonic sensors in waste bins to transmit real-time fill-level data to the system. This would allow for fully automated "Demand-Driven" route generation without human intervention.
2. **Driver Mobile App:** Developing a mobile application for truck drivers that fetches the assigned route from the backend and provides voice-guided navigation using the OSRM response.

3. **Machine Learning for Prediction:** Implementing ML models to analyze historical data and predict waste generation patterns (e.g., predicting that bins near markets fill up faster on Fridays), allowing for proactive planning.
4. **Live Fleet Tracking:** integrating GPS hardware on trucks to monitor their real-time location against the planned route, enabling the planner to detect deviations or delays instantly.

5. Summary

The "Route Optimization System" represents a significant step towards digitizing municipal services. By combining operational research with modern web technologies, this project demonstrates that mathematical optimization can directly contribute to cleaner cities, reduced operational costs, and a more sustainable environment.

Appendices

- Backend Repository
- Frontend Repository

References

1. **Toth, P., & Vigo, D.** (2014). *Vehicle Routing: Problems, Methods, and Applications*. Society for Industrial and Applied Mathematics.

2. **Google OR-Tools.** (2024). *Vehicle Routing Problem Documentation*. Retrieved from https://developers.google.com/optimization/routing

3. **Luxen, D., & Vetter, C.** (2011). *Real-time routing with OpenStreetMap data*. In Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems.

4. **Mishra, A., & Ray, A.** (2020). *IoT cloud-based cyber-physical system for efficient solid waste management in smart cities*. IET Cyber-Physical Systems: Theory & Applications.

5. **Han, H., & Ponce-Cueto, E.** (2015). *Waste Collection Vehicle Routing Problem: Literature Review*. Promet – Traffic & Transportation, 27(4).

6. **MDPI.** (2024). *The Impact of IoT-Enabled Routing Optimization on Waste Collection Distance: A Systematic Review*. Sustainability Journal.

7. **Django Software Foundation.** (2024). *Django REST Framework Documentation*. Retrieved from https://www.django-rest-framework.org/

8. **Leaflet.** (2024). *Leaflet: An open-source JavaScript library for mobile-friendly interactive maps*. Retrieved from https://leafletjs.com/