Rahguzar

Optimizing Field Sales Routes

Kaavish Report
presented to the academic faculty
by

Nabila Zahra	nz07162
Rabia Shahab	rs07528
Iqra Azfar	ia07614
Muhammad Youshay	my07103

In partial fulfillment of the requirements for
Bachelor of Science
Computer Science

Dhanani School of Science and Engineering

Habib University

Spring 2025

Rahguzar

This Kaavish project was supervised by:

Fatima Alvi
Management Trainee
Salesflo Pvt Ltd

Syeda Saleha Raza
Faculty of Computer Science
Habib University

Approved by the Faculty of Computer Science on _____.

Dedication

To our parents, whose unwavering love, endless sacrifices, and heartfelt prayers have been the foundation of everything we've accomplished. Your belief in us, even when we doubted ourselves, gave us the strength to move forward. This journey, and all that it has brought us, is a reflection of your dedication and support.

To our families and loved ones, thank you for your patience, encouragement, and constant presence. Your understanding made room for our long hours, tight deadlines, and moments of exhaustion.

To our teachers and mentors, we are deeply grateful for your guidance and the knowledge you've so generously shared. Your trust in our capabilities challenged us to think bigger and work harder.

To our teammates, thank you for the long nights, the shared goals, and the spirit of collaboration that brought this project to life. Rahguzar is a product of shared vision, hard work, and mutual respect.

Above all, we dedicate this work to Allah (SWT), the source of all knowledge and strength. Without His mercy and guidance, none of this would have been possible.

Acknowledgements

Alhamdulillah, all praise is due to Allah (SWT), whose endless mercy, guidance, and blessings gave us the strength and clarity to complete this project.

We are sincerely grateful to Dr. Syeda Saleha Raza, from the Faculty of Computer Science at Habib University, for her academic mentorship, thoughtful guidance, and unwavering support throughout the course of this project. Her insights and encouragement consistently pushed us to approach our work with both depth and discipline.

We also extend our heartfelt thanks to Ms. Fatima Alvi, Management Trainee at SalesFlo Pvt Ltd, for her practical guidance, domain expertise, and continuous availability. Her support helped us navigate the real-world complexities of the problem we set out to solve.

We are thankful to the Computer Science faculty at Habib University for equipping us with the knowledge and skills that formed the backbone of this project. Every course, discussion, and challenge over the past four years played a role in preparing us for this journey.

We deeply appreciate the collaboration with SalesFlo Pvt Ltd, our industry partner, for providing access to relevant datasets, business context, and timely feedback. Your involvement helped us ground our work in real operational needs.

Our gratitude also goes to the Kaavish Committee and the Office of Undergraduate Research for enabling a structured, well-supported project experience and for providing necessary resources and checkpoints along the way.

To our families and friends, thank you for your constant encouragement, patience, and understanding through all the late nights, deadlines, and moments of stress. Your support made all the difference.

Finally, we acknowledge one another. As teammates, we shared a vision, faced challenges, and built something we are proud of. Rahguzar is a result of that collective effort and commitment.

Abstract

Efficient and scalable journey planning is critical for operational success in distribution and field service management. Manual and semi-automated Permanent Journey Plan (PJP) scheduling methods often lead to inefficiencies such as increased travel times, unbalanced workloads, higher operational costs, and inconsistent service quality. Rahguzar addresses these challenges through a modular, hybrid approach that automates the clustering, scheduling, and routing of field visits. The system employs a hybrid clustering strategy combining minimum spanning tree techniques with K-means refinement for effective territory segmentation, an evolutionary algorithm (EA) for generating balanced, constraint-aware weekly schedules, and Google OR-Tools Traveling Salesman Problem (TSP) solver for optimal intra-day route sequencing. Extensive experiments conducted on real-world datasets demonstrate Rahguzar's ability to reduce travel distances, balance workloads, and maintain computational efficiency at scale. The platform's cloud-based infrastructure, dynamic adjustment capabilities, and KPI-driven monitoring further enhance its real-world applicability. Rahguzar not only automates PJP generation but also lays the groundwork for future enhancements incorporating predictive analytics, real-time dynamic routing, and broader deployment across diverse operational contexts.

Contents

List of Figures

List of Tables

1. Introduction

1.1 Problem Statement

In the dynamic and ever-evolving world of retail and distribution, planning Permanent Journey Plans (PJP) plays a vital role in organizing daily route schedules for field representatives, such as sales executives, to ensure consistent and efficient service to stores. PJPs involve determining which stores to visit on which days and assigning sales representatives while adhering to predefined constraints such as visit frequencies, even visit spacing, workload balancing, consistent assignments, holidays, and minimizing travel distances. However, this planning process is inherently complex. Traditional manual or semi-automated approaches are unable to handle the scale and intricacies of modern Fast-Moving Consumer Goods (FMCG) operations, leading to inefficient routes that increase travel times and operational costs. Misaligned schedules result in missed visits or inconsistent service, while poor resource utilization increases inefficiencies. These inefficiencies also contribute to increased fuel consumption and carbon emissions, further escalating costs and environmental impact. Moreover, managing thousands of stores across diverse regions poses significant scalability challenges.

To overcome these challenges, Rahguzar is proposed as an innovative web-based solution designed to automate and optimize PJP planning, enabling efficient route generation, effective workforce utilization, and reduced operational and environmental costs.

1.2 Proposed Solution

The proposed solution, Rahguzar, is a comprehensive web-based application developed to automate and optimize the planning of PJPs for distributors, managers, and field representatives, including sales executives, merchandisers, and order bookers,

in the FMCG sector. Rahguzar generates efficient, scalable schedules that adhere to critical constraints, such as store visit frequencies, even visit spacing, workload balancing, consistent assignments, holidays, and minimizing travel distances. These tailored schedules ensure that field representatives can execute their tasks effectively while aligning with the organization's operational goals.

The platform features a user-friendly, interactive map interface that allows managers to visualize, adjust, and schedule visits for daily, weekly, or monthly plans. Rahguzar determines the optimal number of field representatives required, improving workforce utilization and preventing inefficiencies such as overstaffing or underhiring.

By optimizing routes, Rahguzar minimizes travel times, operational costs, and fuel consumption, significantly reducing the carbon footprint and promoting sustainability. This automation reduces the need for manual intervention, saving valuable time for managers while ensuring that field representatives have clear, efficient routes tailored to their responsibilities. Rahguzar delivers a scalable, cost-effective, and environmentally sustainable solution for modern FMCG operations, enhancing productivity, consistency, and overall service quality.

1.3 Intended User

This section outlines the target users of Rahguzar, detailing the different types of users in the user base and their interactions with the platform.

- **Managers:** Managers are the primary users of the system, responsible for creating, adjusting, and scheduling optimized journey plans for field representatives. They will interact with the platform through a user-friendly map interface to design, visualize, and modify routes. The system allows managers to generate daily, weekly, or monthly plans, adapt to specific business constraints, and ensure operational efficiency.

- **Field Representatives:** While field representatives, such as sales executives and order bookers, do not directly interact with the system, they are key beneficiaries of its outputs. They rely on the optimized journey plans created by managers to execute their tasks efficiently in the field, ensuring consistent and effective service to stores.

1.4 Project Gantt chart and deliverables

Deliverables for Kaavish I:

- Requirements Specification Document
- System Architecture & Design Diagram
- Prototype for Frontend and Backend Components
- Fully Developed Route Optimization Algorithm
- Performance and Evaluation Report

Deliverables for Kaavish II:

- Comparison with existing Algorithms and improvements
- User-friendly web application with map interface
- Performance Metrics Dashboard
- Pilot Testing Framework and Results
- Final Project Documentation and Evaluation Report

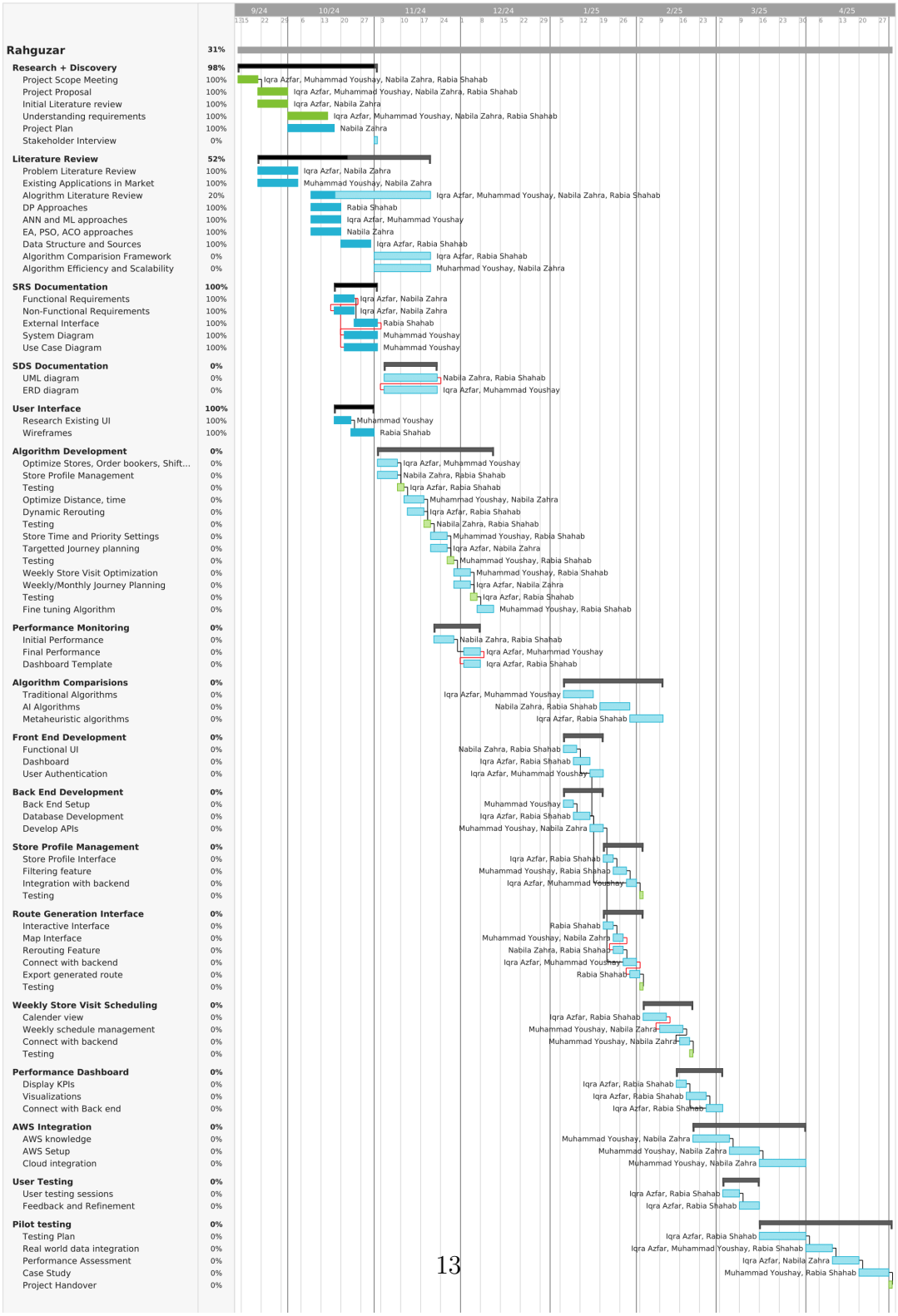

Figure 1.1: Project Gantt Chart

1.5 Key Challenges

This section outlines the major technical and operational challenges we anticipated at the beginning of the project, along with initial strategies considered to address them. A reflection on the challenges actually encountered is presented in later chapters.

- **Data Complexity and Integration:** Managing and integrating diverse datasets, including store profiles, geographic data, and workforce schedules, presents significant complexity, particularly in reconciling inconsistencies and ensuring data accuracy.
  Solution: Utilize specialized data processing libraries to streamline data handling and incorporate iterative testing phases to identify and address integration issues early in development.

- **Multi-Objective Optimization:** Balancing multiple optimization criteria, such as minimizing travel distances, reducing costs, meeting visit frequencies, and ensuring workload balancing, is inherently challenging due to competing priorities.
  Solution: Implement a weighted scoring system to prioritize objectives based on business needs and evaluate outputs from multiple algorithms to identify the most efficient and effective solutions.

- **API Costs and Performance:** Heavy reliance on mapping APIs, such as Google Maps, could lead to high costs and potentially slow performance when processing large-scale data.
  Solution: Optimize API usage through practices like batch processing and caching to reduce the number of API calls. Additionally, utilize trial phases of these services to test and implement them effectively without exceeding cost limits.

- **Adaptability to Business-Specific Constraints:** Organizations may have unique and variable constraints, such as holidays, visit frequency, or workload preferences, which require flexible accommodation in the route planning system.
  Solution: Design the system to allow customization of route parameters and enable managers to manually adjust routes through an intuitive interface, ensuring flexibility for diverse business requirements.

2. Literature Review

This section explores the techniques and methodologies used for optimizing Permanent Journey Plans (PJP), emphasizing their relevance to modern logistics and distribution systems. It highlights a range of optimization algorithms, including Dynamic Programming, Genetic Algorithms, Particle Swarm Optimization, Ant Colony Optimization, and Mixed-Integer Linear Programming, alongside the role of clustering methods like K-means. These approaches address critical challenges such as minimizing travel distances, adhering to time constraints, balancing workloads, and ensuring scalability in large-scale routing problems. The review establishes a foundation for understanding how these techniques contribute to efficient and effective PJP optimization.

2.0.1 Journey Planning for Industries

In today's highly competitive marketplace, the speed and efficiency of transportation can vastly influence the success and failure for industries like fast-moving consumer goods, retailers, logistics, and distribution centers [4]. For these sectors, logistics can account for over 30% of distribution costs and contribute 15-40% to the overall product price [5]. Therefore, businesses need to plan their journey routes with the utmost efficiency to enhance their operational productivity and reduce their costs. The journey plans generated, known as Permanent Journey Plans, or PJPs, have to meet meticulous criteria such as minimizing the distance traveled, making sure time-sensitive plans are met with punctuality, and maximizing the number of places that can be visited on a certain day [21]. Traditional approaches to journey planning often rely on manual processes or heuristic methods, making it a tedious process and unable to handle the complexities required by modern logistics systems, increasing the logistics cost by 10-30% [17, 2]. For developing countries such as Pakistan, the wholesale and retail trade sector contributes to 18% of the overall GDP, emphasizing on the need to further improve the routing efficiency to support the financial stability and growth of businesses in the retail sector [12].

Recent developments in algorithmic approaches, combined with advancements in operations research and computational techniques, have renovated how to assess journey planning problems, and how we can address them [3]. The Vehicle Routing Problem (VRP) is an NP-hard problem widely recognized in journey planning, serving as a benchmark for testing and improving optimization algorithms. These problems are applied in scenarios that require the optimization of Permanent Journey Plans, or PJPs, where the routes and schedules created have to be consistent over extended periods. The created PJPs must account for several constraints such as frequency of visits that ensure locations are serviced at appropriate intervals to meet the customer demands, time window constraints to align the schedules with store operating hours, and distributing workload evenly across the number of available sales force. Moreover, it is vital to balance the rigidity of fixed routes with the dynamic nature of real-world variables, such as changes in traffic, demand fluctuations, and operational constraints.

2.0.2 Optimization Techniques in Route Planning

Solving complex, NP-hard problems like Permanent Journey Plan (PJP) optimization requires advanced techniques to balance constraints and objectives efficiently [1]. Optimization methods such as Mixed-Integer Linear Programming (MILP), Dynamic Programming (DP), Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) have been widely applied to address diverse routing scenarios, including static planning and dynamic adjustments.

An important approach for improving computational efficiency in large-scale problems is clustering, which involves grouping locations based on proximity or other criteria. K-means clustering is one of the most commonly used techniques for this purpose [1], dividing locations into well-defined regions to simplify route optimization. By reducing the problem size, clustering allows optimization algorithms like MILP, GA, or PSO to operate independently within each cluster, making the overall process more computationally feasible.

MILP excels at providing exact solutions by modeling objectives and constraints, such as visit frequencies and workload balancing, as linear functions. Although MILP is computationally expensive for large-scale problems, integrating it with K-means clustering significantly reduces complexity [16]. DP, on the other hand, decomposes problems into smaller subproblems, solving each recursively and storing intermediate results. While useful for smaller PJP tasks, its computational demands make it more practical when combined with other methods for hybrid solutions [25].

Metaheuristic algorithms like GA, PSO, and ACO provide robust alternatives for

handling the multi-objective and dynamic nature of PJP optimization. GA iteratively refines solutions through evolutionary strategies like selection and crossover, effectively balancing travel distances and visit frequencies [15]. PSO, inspired by swarm behavior, optimizes routes by simulating collective decision-making and works particularly well when combined with clustering methods like K-means to refine regional solutions [11]. ACO, modeled on the foraging behavior of ants, is adept at exploring combinatorial solution spaces, avoiding local optima, and identifying efficient routes [20].

Clustering and optimization algorithms work hand in hand for large-scale PJP problems. Clustering simplifies the problem by dividing it into smaller regions, while optimization techniques like GA, PSO, and MILP refine routes within each cluster. This integrated approach ensures computational efficiency, scalability, and practical applicability, making it well-suited for addressing PJP-specific challenges such as visit frequencies, even workload distribution, and time-sensitive constraints.

While researching for ways to solve the problem of creating optimal PJPs, we have come accross several approaches that solve a similar problem while optimizing other objectives that are discussed in detail below.

2.0.3 Dynamic Programming (DP) to Solve VRPTW

Dynamic Programming (DP) is a widely-used optimization technique capable of solving complex routing problems with constraints, making it particularly suitable for Permanent Journey Plan (PJP) optimization. Its ability to decompose a problem into smaller, manageable subproblems while ensuring global optimality makes it a valuable tool for addressing challenges such as visit frequencies, even visit spacing, time windows, and workload balancing.

A study [14] applied a DP algorithm to solve the Vehicle Routing Problem with Time Windows (VRPTW). This framework incrementally builds routes by sequentially adding stops while verifying feasibility under constraints. In a PJP context, this approach can be extended to align store visits with predefined schedules and ensure adherence to operating hours. The study reported a reduction in routes by 18% and travel distances by over 5%, highlighting the efficiency gains possible when dynamic adjustments are incorporated into route planning. For PJPs, such gains translate into increased store coverage and reduced operational costs.

Another study by Desaulniers et al. (2002) [6] focused on the use of DP to optimize fleet routing while accounting for time window constraints. Their approach incorporated a state-space relaxation method, reducing computational overhead while maintaining solution quality. For PJP, this method can ensure that sales represen-

tatives follow efficient schedules without exceeding working hour limits or violating store-specific time windows.

Furthermore, a study by Feillet et al. (2004) [7] introduced a label-setting algorithm based on DP to solve VRP with soft time windows. This approach allowed flexibility in time window adherence, a feature particularly useful for PJP scenarios where strict adherence to schedules may not always be possible. By penalizing minor deviations rather than rejecting infeasible solutions outright, this method ensures that PJPs remain robust under dynamic operational conditions.

A hybrid DP approach by Kilby et al. (2002) [13] combined DP with heuristic techniques to address large-scale routing problems. The hybrid method used DP to solve subproblems within clusters, which were formed using heuristic clustering methods. This is particularly relevant for PJPs, where clustering stores into smaller groups based on proximity or workload can significantly reduce the computational complexity of route planning.

2.0.4 Mixed-Integer Linear Programming (MILP) for PJP

Mixed-Integer Linear Programming (MILP) is a mathematical optimization technique widely used in route planning and scheduling problems. It is particularly relevant for optimizing PJPs, where the focus is on visiting stores to generate demand while adhering to constraints such as visit frequencies, time windows, workload balancing, and minimizing travel distances.

MILP allows for the formulation of optimization problems as linear objective functions with linear constraints. For PJP scenarios, the objective might involve minimizing the total distance traveled or the time taken, while constraints ensure adherence to store-specific visit frequencies, consistent assignments for sales representatives, and balanced workloads. Unlike heuristic methods, MILP offers mathematically optimal solutions, making it a valuable tool for small to medium-scale PJP problems [16].

A study by Sierksma and Tijssen (1998) highlights the use of MILP in sales territory alignment and routing problems, which are closely related to PJP optimization. Their approach ensures balanced workload distribution among sales representatives and minimizes travel costs [23].

For larger problems, MILP is often combined with clustering techniques to break down the problem into smaller subproblems. In such cases, clustering algorithms segment stores into manageable groups based on geographical proximity or other criteria, and MILP is applied within each cluster to generate optimal routes [18]. This hybrid approach improves computational feasibility while maintaining solution

quality.

Despite its computational intensity, MILP serves as an excellent benchmark for evaluating heuristic or metaheuristic approaches, such as Genetic Algorithms (GA) and Particle Swarm Optimization (PSO). For example, a study by Schneider et al. (2014) demonstrated that MILP models could effectively optimize vehicle routing problems with time windows, which is directly applicable to PJP scenarios involving time-sensitive store visits [22].

2.0.5 Metaheuristic Approaches for VRPTW

Metaheuristic algorithms like Genetic Algorithms (GA) and Ant Colony Optimization (ACO), widely applied to solve the Vehicle Routing Problem with Time Windows (VRPTW), are also effective for optimizing Permanent Journey Plans (PJP) due to their ability to handle multi-objective and constraint-driven problems. A range of single-objective and multi-objective approaches has been explored to address the Vehicle Routing Problem with Time Windows (VRPTW). Brock University conducted a study [19] comparing single-objective and multi-objective methods, emphasizing a non-biased approach without weighted sum scoring to avoid compromising between vehicle usage and travel distance. The genetic algorithm (GA) used demonstrated optimal or near-optimal results for clustered data but was less consistent for uniformly distributed customer locations. The multi-objective approach faced challenges with Pareto dominance, leading to suboptimal compromises and limited generalizability to complex real-world scenarios.

A novel Multi-Objective Evolutionary Algorithm (MOEA) incorporating Jaccard's coefficient was proposed to improve population diversity and Pareto-based approximations [9]. This method outperformed single-objective algorithms like EA with deterministic or randomized selection, achieving 2% to 5% better outcomes. However, its scalability was limited when applied to datasets with higher travel distances, posing challenges for large or complex datasets due to increased computation time.

A bi-objective GA combined with goal programming was designed to minimize vehicle use and travel costs, leveraging Pareto ranking and fitness evaluation with Pareto ranks [10]. This method showed positive results for clustered data, correlating vehicle count with travel cost. Yet, it struggled with unclustered datasets and mixed scenarios, which led to higher costs and limited effectiveness for larger, dynamically constrained problems.

The Hybrid Ant Colony Optimization (HACO) approach [26] aimed to solve the multi-objective vehicle routing problem with flexible time windows (MOVRPFlexTW)

by balancing customer satisfaction and cost. HACO used a hybrid strategy with multiple mutation operators and incorporated Pareto optimality, achieving good convergence speed and accuracy in Solomon's benchmark problems. While HACO demonstrated robustness and adaptability to higher-dimensional data, finding optimal parameters was sensitive, potentially impacting efficiency and scalability for larger datasets.

These GA-based strategies each have strengths and limitations. The Brock University study highlighted trade-offs and biases when using Pareto dominance in multi-objective approaches. MOEA's incorporation of Jaccard's coefficient improved diversity but faced scalability issues with higher travel distances. The bi-objective GA showed potential with clustered data but lacked stability in non-clustered scenarios. HACO, in contrast, outperformed traditional GAs and EAs, showing potential for real-world application and better scalability. Collectively, these findings highlight the need for further refinement and hybrid solutions to enhance generalizability and computational efficiency for real-world VRPTW problems.

2.0.6 Hybrid Clustering Algorithms

Another method to approach the VRP problem is to combine optimization techniques with clustering algorithms to solve complex VRP problems under real-life constraints. The clustering algorithms will be covered in the sections below.

Various clustering and optimization algorithms have been developed to address the complexities of Vehicle Routing Problems (VRPs). A three-phase algorithm approach from paper [28] that uses clustering, Mixed-Integer Linear Programming (MILP), and a Traveling Salesman Problem (TSP) approach has proven effective, reducing transport costs by 10%-20% in real-world cases. This method clusters customers into manageable subproblems, optimizes routes using MILP, and fine-tunes them with a TSP approach. However, its performance depends on initial clustering and struggles with large-scale problems.

Another approach [27] uses the Discrete Firefly Algorithm (DFA) for clustering and a modified 2-opt algorithm for route optimization. DFA outperformed k-Means and manual methods, achieving up to a 15% cost reduction and faster computation. Its main drawback is the dependence on initial clustering quality and computational intensity.

A hybrid algorithm [24] for VRPs with Time Windows (VRPTW) begins with K-means clustering and refines it using OPTICS. Routes are generated using a Nearest Neighbor heuristic and optimized with 2-opt, showing a 5.06% distance reduction for smaller instances and 11.97% for larger ones. Its limitations include assumptions of

static conditions and a need for parameter tuning.

The KACO algorithm for the Dynamic Location Routing Problem (DLRP) discussed in paper [8] integrates K-means clustering with Ant Colony Optimization (ACO), using immigrant schemes to adapt routes dynamically. It outperformed traditional ACO, showing better tour lengths and adaptability under dynamic conditions. However, challenges with clustering quality and high computational demands remain.

These approaches demonstrate that combining clustering with optimization methods effectively handles VRPs, offering cost and efficiency benefits as compared to only adapting optimization algorithms to solve VRP problems. Moreover, they showcase better adaptability and scalability to real-world datasets as opposed to algorithms that solely relied on optimization to solve VRP that could not perform well with large and real-world datasets.

Drawing from the extensive range of methodologies explored, it is evident that optimization algorithms and clustering techniques play a crucial role in addressing complex routing and scheduling challenges across various domains. Genetic Algorithms (GA) have demonstrated their utility in multi-objective optimization, balancing competing factors such as distance minimization and timely service delivery through evolutionary processes. Similarly, Particle Swarm Optimization (PSO), inspired by the collective behavior of swarms, offers computational efficiency and adaptability in determining optimal routes and clustering solutions. Ant Colony Optimization (ACO), with its biologically inspired probabilistic approach, has proven adept at solving combinatorial problems like the Vehicle Routing Problem (VRP) by navigating through solution spaces to find optimal paths while evading local optima. Dynamic Programming (DP), although computationally intensive, provides exact solutions to problems by systematically breaking them into smaller subproblems.

The integration of these algorithms with clustering techniques, such as those used for grouping locations based on proximity and service requirements, enhances the efficiency of solving the vehicle routing and scheduling problems. Techniques like hierarchical clustering, k-means clustering, and density-based spatial clustering play pivotal roles in defining clusters that optimize routing paths, reduce travel times, and ensure that each route adheres to time windows and service constraints. These clustering approaches, combined with the optimization algorithms, help in forming practical, real-world solutions for complex logistical challenges. Through this literature review, it is clear that continued research and innovation in these areas are vital for developing more robust and scalable solutions to complex optimization problems, ultimately enhancing the efficiency and effectiveness of route optimization and planning in various industries.

3. Software Requirement Specification (SRS)

This Software Requirements Specification (SRS) outlines the software and system requirements for Rahguzar, a route optimization project aimed at improving efficiency in retail and distribution. It details functional requirements for system capabilities and non-functional requirements like performance and scalability. The document also includes system block diagrams, use cases, and external interfaces to clarify the system's design and interactions.

3.1 Functional Requirements

The functional requirements for this system focus on creating an efficient journey-planning solution tailored for order bookers and sales representatives. This system is designed to optimize routes based on key operational parameters and provide dynamic rerouting, while also supporting targeted store visits and frequency schedules. Additional modules include a user-friendly web platform, pilot testing and validation, and performance monitoring to ensure enhanced efficiency and measurable sales impact.The following are the functional requirements for each module and their respective functions.

Module 1: Route Optimization Algorithm

Function 1: Route Generation

- Implement an algorithm that creates journey plans optimized for order bookers and sales representatives.
- The optimization should consider multiple factors:

Sub Function 1: Order Booker Shift Times
Align route planning with the working hours of each order booker.

Sub Function 2: Number of Shops to Visit
Adjust routes to maximize the number of shops visited within constraints.

Sub Function 3: Number of Order Bookers Available
Allocate routes based on the number of available staff.

Sub Function 4: Total Distance Traveled
Minimize the total distance covered during journeys.

Sub Function 5: Total Travel Time
Reduce the time taken to complete the planned routes.

Sub Function 6: Store Service Times
Account for different service time requirements at each store.

Function 2: Dynamic Rerouting

- It allows users to adjust the generated routes based on their preferences. They can move stores between order bookers and modify store status (e.g., active/inactive).
- It simply enables real-time rerouting, with the algorithm instantly updating the allocation of stores and route paths to reflect user preferences, providing flexibility and adaptability in journey planning.

Function 3: Store Visit Frequency:

- Design the algorithm to include visit frequency for each store (e.g., once, twice a week).
- Ensure that the routes align with required visit frequencies without any overlap and maintains at least a 3 day gaps between repeated visits.

Function 4: Journey Planning for Different Timeframes:

- Support the generation of routes by allowing custom day plan creation and enable downloads for extended periods, based on the target timeframe.
- Ensure that stores are scheduled accurately according to their frequency needs while minimizing travel and maximizing coverage.

Function 5: Store Profiling:

- Maintain detailed profiles for each store, including:

  Sub Function 1: Geographical hierarchy (region, zone, territory, town).

  Sub Function 2: Sales channel type (e.g., wholesale or retail).

- Enable users to filter stores by parameters like region, sales channel, or visit frequency. Allow custom filters linked to all stores by uploading relevant data (e.g., location, sales volume), enabling dynamic application across stores. This approach enhances flexibility and control in route optimization.

- Enable users to modify store status (active/inactive) directly within the system.

Function 6: Targeted Journey Plans:

- Enable the system to generate routes targeting specific store types (e.g., only wholesale stores in a particular region).

- Allow users to select areas and stores for targeted visits, ensuring that the algorithm generates routes based on these selections.

Module 2: Web Application Platform

Function 1: User Interface for Route Management:

- Create a user-friendly web interface for managers to access and manage journey plans.

Function 2: Integrate Map Interface:

- Designing, viewing, and adjusting routes.

- Displaying essential shop details (e.g., store profiles, geographical location).

- Defining and viewing boundaries like regions, territories, and areas.

Function 3: Manual Override and Adjustment:

- Provide a manual override feature that allows managers to adjust routes manually.
- Include input options for master data (store profiles, visit frequencies, geographical details) directly within the platform.

Function 4: Downloadable Reports:

- Allow users to export optimized or manually adjusted routes in a downloadable format for offline analysis or distribution.

Module 3: Pilot Testing and Validation

Function 1: Test Plan Creation:

- Develop a comprehensive pilot testing framework to validate the system's effectiveness.

Function 2: Real-World Data Integration:

- Gather real-world data during pilot testing, ensuring data is accurate and reliable.

Function 3: Performance Assessment:

- Collect and analyze data (travel time, number of shops visited, sales impact, etc.) to assess system performance.
- Implement an adjustment process based on pilot feedback to enhance the algorithm.

Function 4: Case Study Development:

- Compile a case study summarizing the outcomes, findings, and system performance.

Module 4: Performance Monitoring and Reporting

Function 1: Key Performance Indicators (KPIs):

- Track and display metrics like:

  Sub Function 1: Average travel time per route.

  Sub Function 2: Number of shops visited per shift.

  Sub Function 3: Number of order bookers required per distribution.

  Sub Function 4: Sales uplift due to route optimization.

  Sub Function 5: Algorithm response time for route recalculations.

Function 3: Metrics Dashboard:

- Develop a dashboard to visualize KPIs, providing insights into efficiency improvements, time savings, and sales increases.

3.2 Non-functional Requirements

The Non-functional requirements define the quality and performance attributes of this system, focusing on scalability, usability, and reliability. They ensure optimal performance under varying conditions while maintaining data security and adaptability for user needs. Additionally, following requirements also facilitate maintainability for future enhancements, ensuring the system remains effective over time.

1. **Scalability, Efficiency, and Memory Optimization**

   - The system is designed to efficiently handle large datasets and scale up to 1000+ stores effectively as the amount of data increases, ensuring optimal performance and quick response times within 2 minutes for route generation.
   - It should also have a low memory footprint, ensuring that it can scale effectively without degrading system performance or requiring excessive computational resources.

2. **Usability**

- A user-friendly web application interface is required, allowing users to manage, optimize, and adjust journey plans easily.
- The map interface must provide intuitive visualization and interactivity for planning and real-time adjustments.

3. **Reliability and Performance**

   - The optimization algorithm should be robust, quickly generating optimal routes based on parameter changes, such as order booker shifts or the number of stores per booker.
   - The system should maintain a minimum uptime of 95%, allowing for up to 36 hours of downtime per month for a smooth user experience.

4. **Security**

   - Data security protocols should protect sensitive information such as store profiles and geographical hierarchies, especially when integrating third-party APIs.
   - User authentication mechanisms should be implemented to ensure that only authorized personnel can access sensitive data and functionalities within the system.

5. **Adaptability and Flexibility**

   - Manual override functionality should allow users to adjust routes when needed, ensuring flexibility in route planning.
   - The system should support long-term planning options, including daily, weekly, or monthly route generation and optimization.

6. **Maintainability**

   - The architecture should allow for future enhancements, such as integrating additional parameters or algorithms for optimization, without significant rework.

3.3 External Interfaces

3.3.1 User Interfaces

The Web application interface is designed to be user-friendly and intuitive, allowing managers to easily navigate through the system. The interface includes various features such as route management, store profiling, and performance monitoring. The design focuses on providing a seamless experience for users, enabling them to efficiently manage their journey plans and access relevant data.

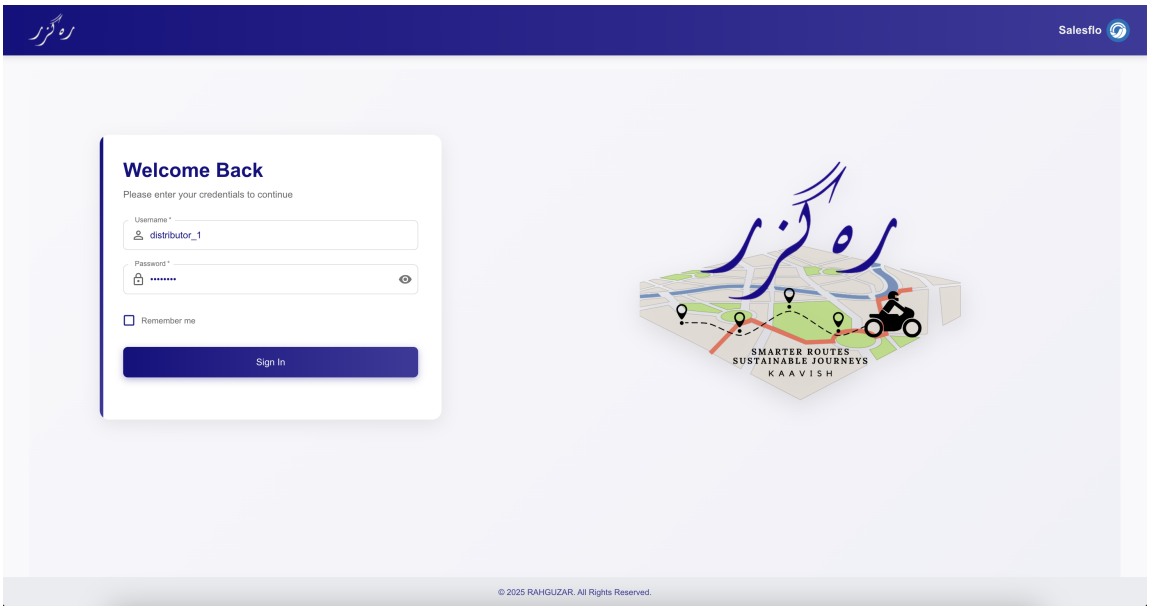

Figure 3.1: Login Screen

The login screen is shown in Figure 3.1, letting the user enter their username and password to login.

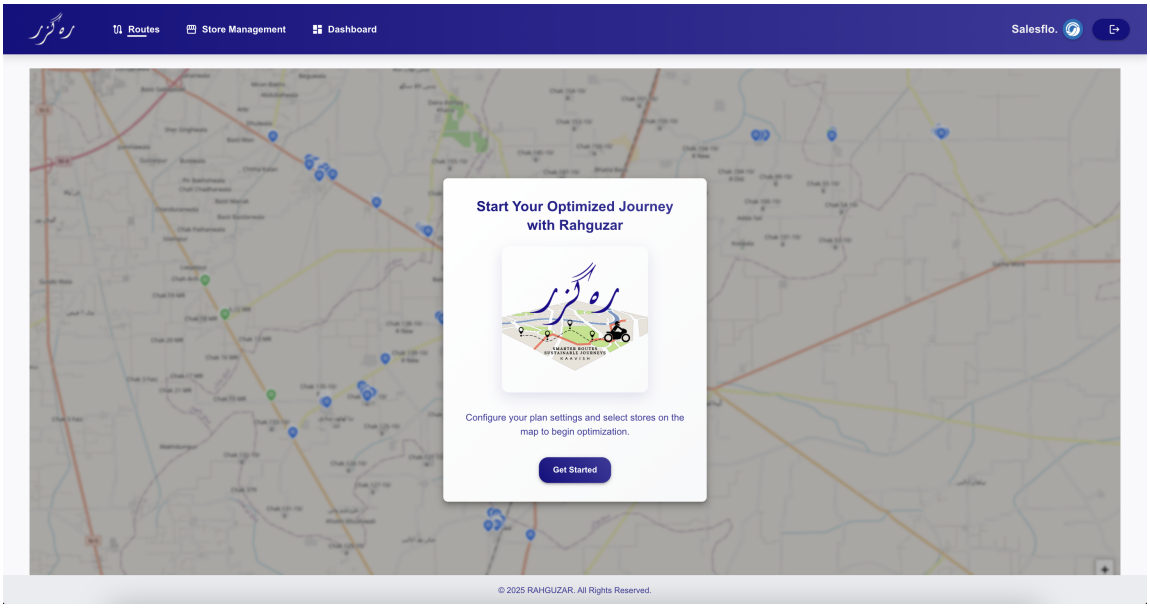

Figure 3.2: Home Page

The “Home Page” screen is shown in Figure 3.2. The Route plan generator is the main screen where the instructions to start are provided.

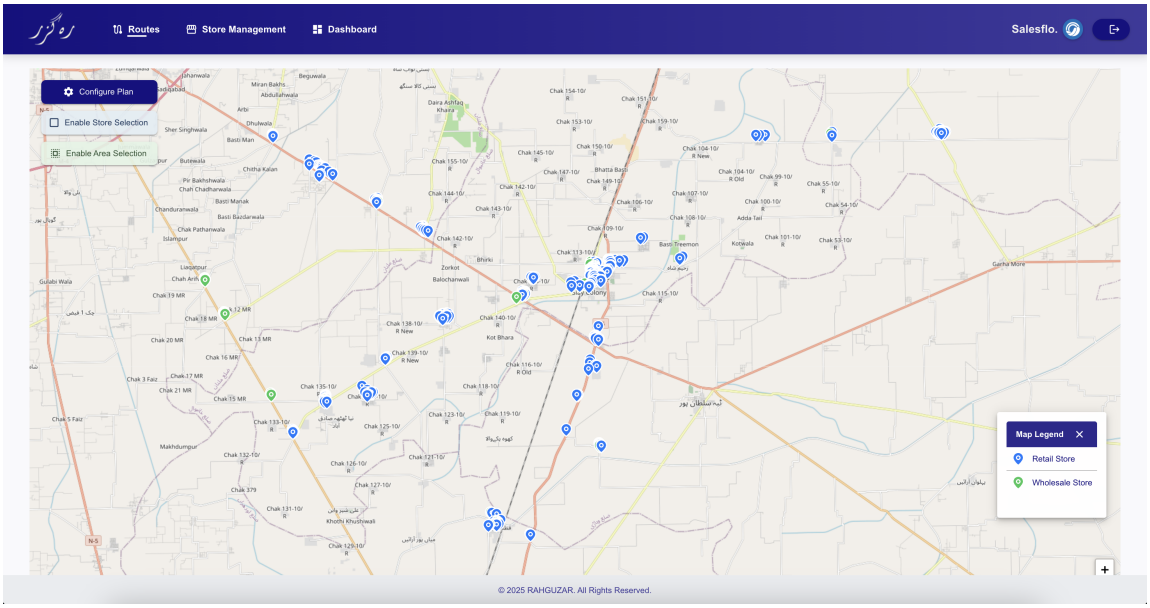

Figure 3.3: Plan Route Page

The “Map” screen is shown in Figure 3.4. In this screen, the manager will be able to view the map and the stores on it. The manager can select stores using area selection or point selection which will be used in the plan.

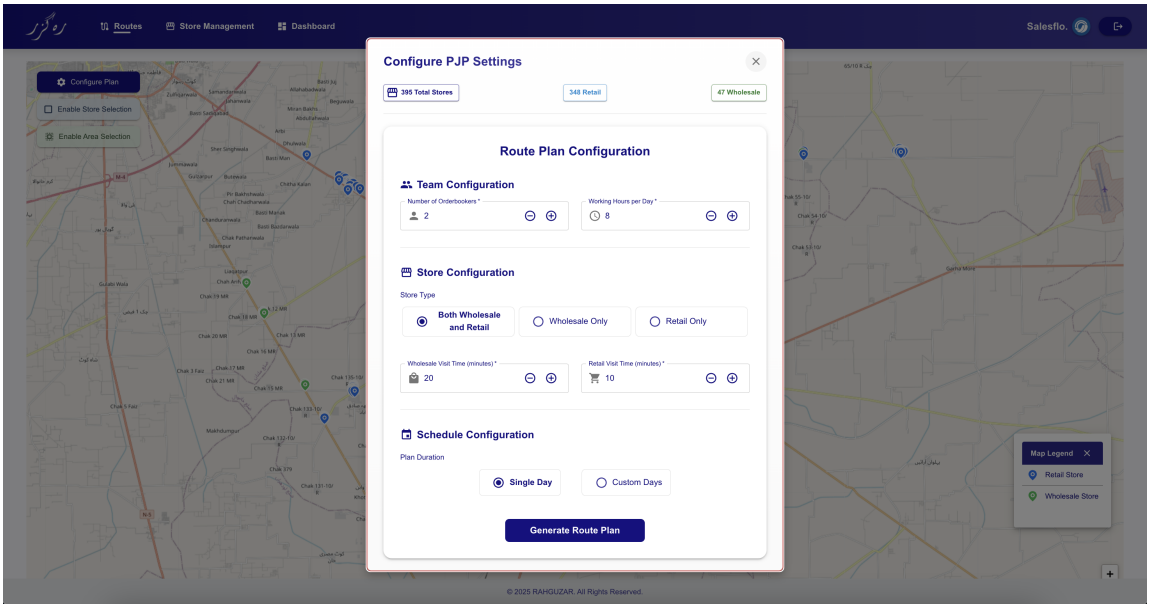

Figure 3.4: Configure Route Parameters

The manager can configure the route parameters in the "Configure Route" screen shown in Figure 3.4. In this screen, the manager will be able to set the shift timings, number of order bookers, and the number of stores to visit. The manager can also set the travel time and service time for each store.

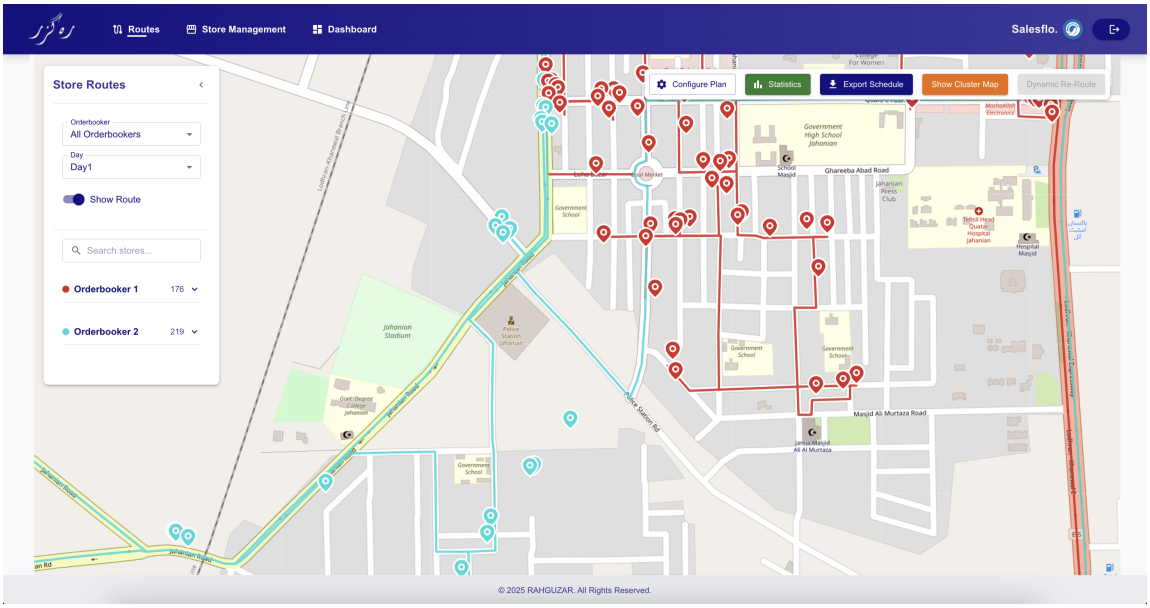

Figure 3.5: View Routes Page

The "View Route" screen is shown in Figure 3.8. In this screen, the manager will be able to view the routes generated by the system. The manager can also view the route details such as the distance, travel time, and service time for each store. The manager can also view the route on the map and download it in a CSV format.

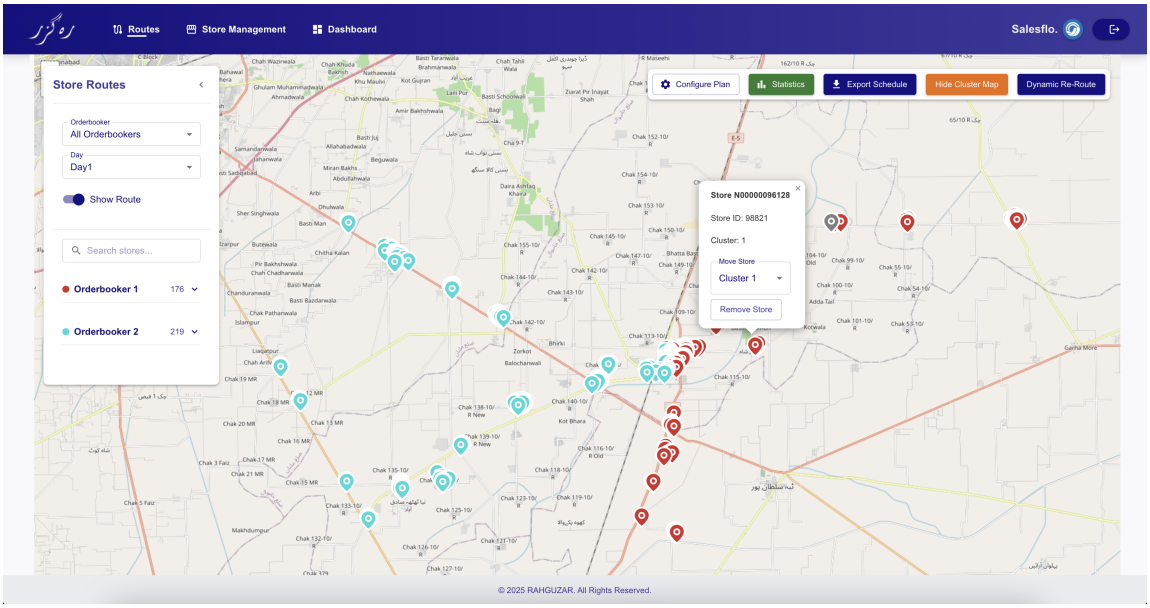

Figure 3.6: Cluster Map Page

The "Cluster Map" screen is shown in Figure 3.8. In this screen, the manager will be able to view the clusters generated by the system. The manager can modify the stores in clusters by removing them or moving them to another cluster, after which they can dynamically reroute the plan.

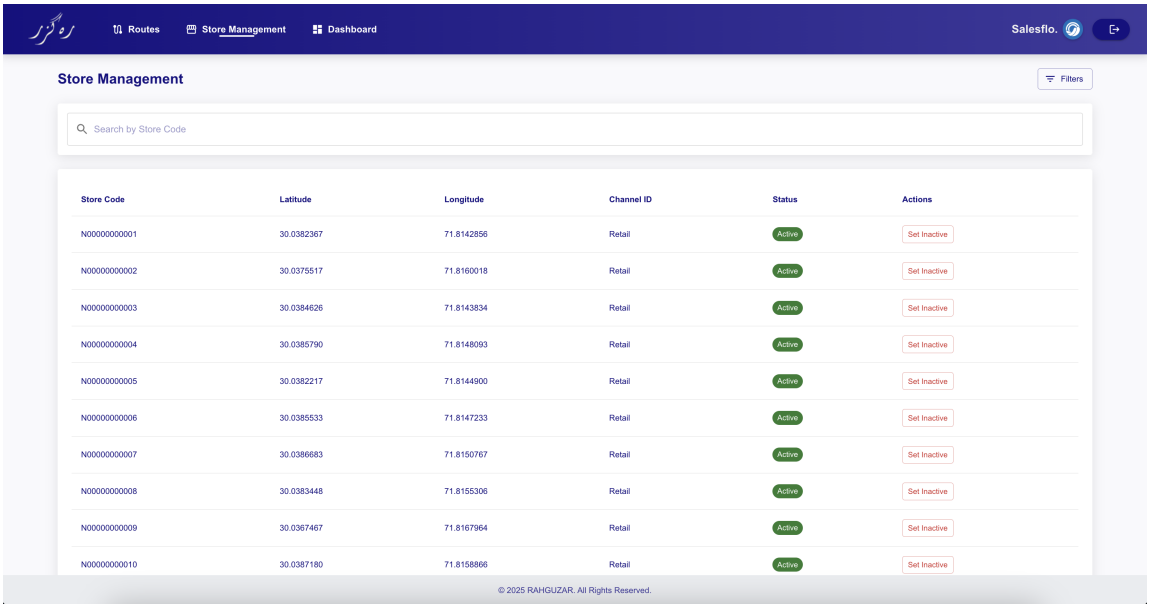

Figure 3.7: Stores Management Page

The "Stores Management" screen is shown in Figure 3.7. In this screen, the manager will be able to search a certain store, apply filters in their search, and view its information such as its coordinates. The manager has the option to view more than 1 store's information at a time, and to decide to set a store active or inactive.

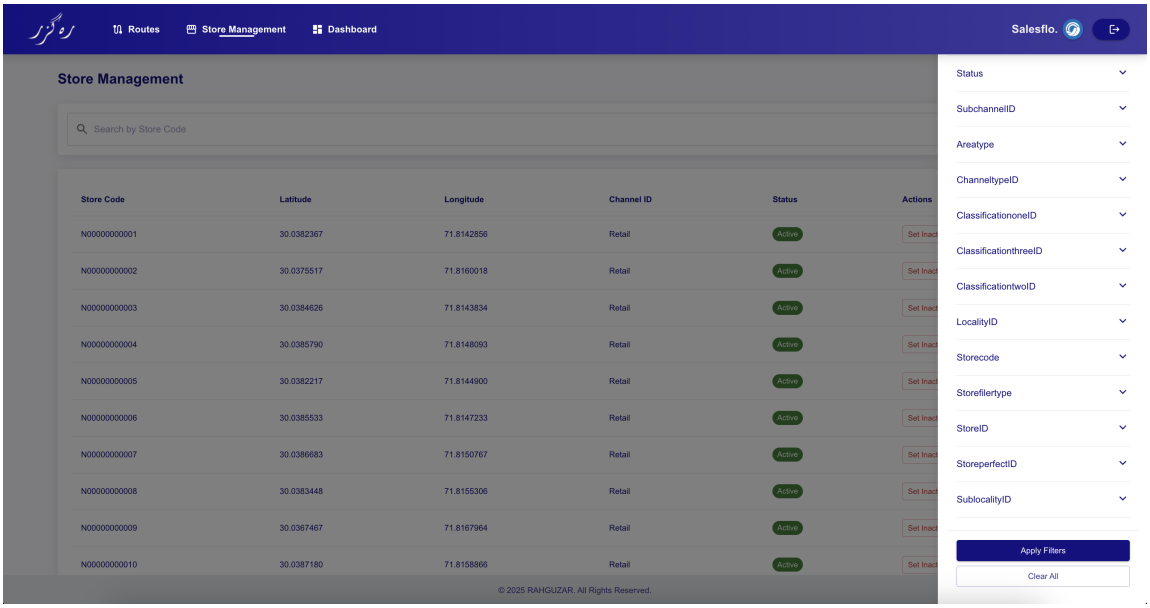

Figure 3.8: Store Filters

The “Store Filters” screen is shown in Figure 3.8. In this screen, the manager will be able to view the filters applied to the stores.

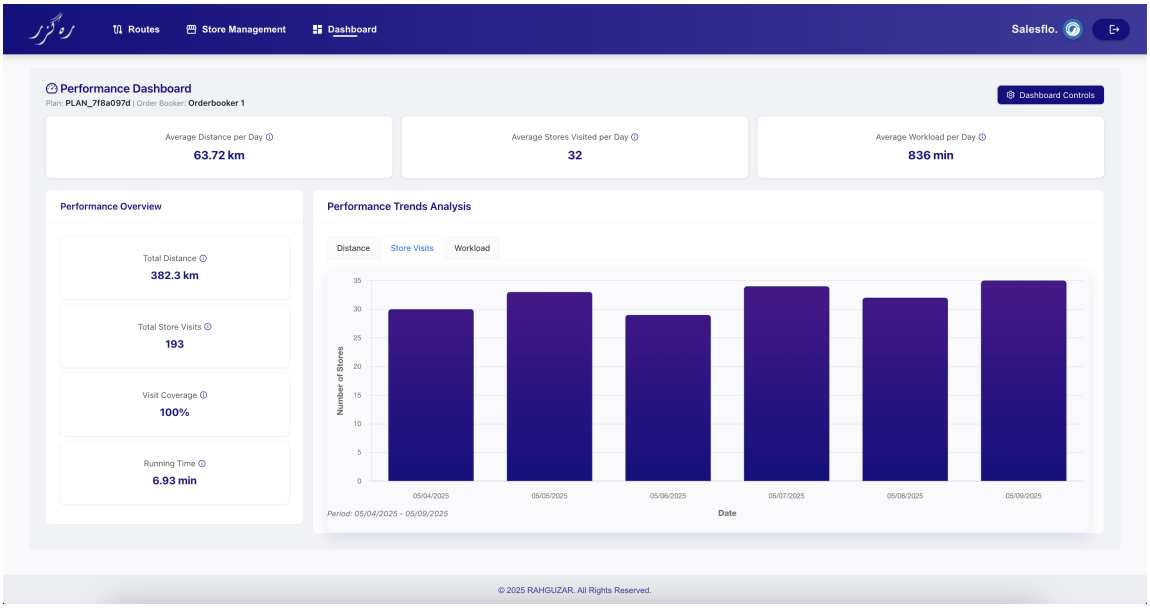

Figure 3.9: Dashboard Page

The "Dashboard" screen is shown in Figure 3.9. In this screen, the manager will be able to view certain KPIs that indicate the overall performance of the routes such as Average Daily Distance and Workload per Representative, Average travel time per per orderbooker or over all orderbookers.

3.3.2 Application Program Interface (API)

The API layer is the backbone of the Rahguzar system, enabling seamless communication between the frontend, backend, and external services. Designed using REST principles, the API ensures efficient data exchange, operational scalability, and secure access control.

REST API Endpoints

The system is built with RESTful API endpoints that manage data transfer between components. Key endpoints include:

- **Route Management Endpoints:**

  - `POST /optimize` — Submits parameters to generate optimized journey plans.
  - `POST /reroute-clusters` — Dynamically reassigns stores and reroutes OBs.
  - `GET /get_routes` — Retrieves finalized optimized routes.

- **Store Profile Endpoints:**
  - `GET /api/stores` — Fetches all store profiles.
  - `POST /assign_store` — Assigns a store to an order booker.
  - `PUT /api/stores/{store_id}` — Updates metadata of a specific store.

- **Dashboard and KPI Endpoints:**
  - `GET /get_kpi_data` — Returns key performance indicators.
  - `GET /get_graph_data` — Fetches time-series analytics for visualization.

Authentication and Authorization

To secure API access, the Rahguzar system employs JWT (JSON Web Tokens) for user authentication.

- `POST /login` — Authenticates user credentials and returns a JWT token.
- **Token Usage:** The JWT token must be attached to the `Authorization` header as `Bearer <token>` in all subsequent API requests.

External Services Integration

To support essential routing and mapping functionalities, the system integrates with third-party services:

- **Open Source Routing Machine (OSRM):**
  - Dockerized and hosted on an EC2 instance for internal routing.
  - **Endpoint:** `GET /route/v1/driving/{lon,lat;lon,lat,...}` — Returns route geometry and travel metadata in GeoJSON format.

Route Optimization Engine

The route optimization component is embedded in the backend, using hybrid algorithms for clustering, scheduling, and routing.

- **Python + OR-Tools:** Implements clustering, scheduling and route generation logic using Evolutionary Algorithms and TSP solvers.

Data Storage and Synchronization

The system utilizes a relational data model hosted on AWS RDS using PostgreSQL.

- **PostgreSQL (via RDS):** Stores all persistent data including users, stores, routes, plans, and KPI metrics.
- **Synchronization Mechanisms:** Data flows between backend modules and frontend views are synchronized via well-defined API transactions and database-level consistency checks.

Data Monitoring

To enable effective monitoring, the API supports data updates and performance tracking:

- **Performance Analytics:** Endpoints such as `/get_kpi_data` and `/get_graph_data` provide real-time insights on route efficiency, travel time, visit compliance, workload distribution, and other KPIs to support ongoing performance tuning and planning.

3.3.3 Hardware/Communication Interfaces

Our System relies on specific hardware and communication protocols to ensure seamless operation and efficient data management.

Client-Side Hardware Requirements

Manager Workstations: Managers access the system through web browsers on desktops or laptops for route planning and monitoring. Recommended specifications include:

- **Processor:** Dual-core or higher.

- **RAM:** 4GB minimum.
- **Internet Connection:** Stable broadband with at least 5 Mbps for real-time data processing.
- **Browser Compatibility:** Supports modern web browsers like Chrome, Firefox, and Edge.

Server-Side Infrastructure

The backend and database components are hosted on cloud infrastructure (AWS) to ensure availability, fault tolerance, and scalability.

- **Application Server (EC2):**
  - Hosted on an Amazon EC2 instance (e.g., t3.medium or higher).
  - Runs the Flask-based backend application, route optimization engine, and API endpoints.
  - Configured with Gunicorn and optionally Nginx for request handling and reverse proxy.
  - Operating System: Ubuntu Server 20.04 LTS or equivalent.
- **Database Server (RDS):**
  - PostgreSQL instance managed via Amazon RDS.
  - Stores all system data including user credentials, store metadata, plan configurations, route outputs, and KPI records.
  - SSD-based storage with automated backup and optional multi-AZ redundancy.
- **Storage:** EC2 instance uses SSD storage for intermediate computation results and application logs.
- **Networking:** Secured via AWS security groups. Backend accessible via HTTPS; database accessible only through private IPs or controlled access rules.

Communication Interfaces

- **Internet Connection:** All interactions between the frontend and backend rely on a stable internet connection for data synchronization and API requests.

- **Data Communication Protocols:**
  - **HTTPS:** Secure communication between frontend, backend, and external services like Google Maps API.
  - **API Calls:** RESTful API requests enable data transfer for route management, visit updates, and user authentication.

Routing Service (Dockerized OSRM)

The system utilizes a containerized instance of the Open Source Routing Machine (OSRM) to compute real-world travel distances and generate accurate road-based routes.

- **Deployment Method:** OSRM is deployed via Docker on an EC2 instance to ensure isolation, portability, and ease of deployment.
- **Routing Backend:** Based on preprocessed OpenStreetMap data using the ‘osrm-backend‘ image.
- **Port Configuration:** Accessible via HTTP on port 5002 internally, secured behind firewall rules.
- **Container Runtime:** Docker Engine on Ubuntu Server.
- **Data Source:** OSM PBF file for the region of operation (e.g., Pakistan or selected city).
- **Use Case Integration:** Accessed by the backend during route optimization to compute driving distances between stores.

3.4 Use Cases

- **User Authentication and Access Control**
  - **Description:** Ensure secure access to the system for both managers and field staff.
  - **Actions:**
    - As a manager, I want to log in securely to access my dashboard and create journey plans.

    - As the system, I need to authenticate users to ensure only authorized managers and field staff access the application.

- **Route Plan Creation, Configuration, and Optimization**
  - **Description:** Enable the creation of optimized routes based on configurable parameters.
  - **Actions:**
    - As a manager, I want to create a new optimized route for my field team, configuring parameters like shift timings and visit frequency to align with operational requirements.
    - As the system, I need to generate optimized routes using parameters such as shift times, visit frequencies, and priorities, ensuring efficient route suggestions for managers.

- **Plan Generation**
  - **Description:** Generate long-term journey plans based on store visit requirements.
  - **Actions:**
    - As a manager, I want the ability to generate custom day journey plans, providing my team with a consistent schedule for store visits.
    - As the system, I need to automatically generate journey plans for custom day schedules based on specified visit frequencies.

- **Dynamic Rerouting and Manual Overrides**
  - **Description:** Allow for real-time adjustments to routes, either dynamically by the system or manually by the manager.
  - **Actions:**
    - As a manager, I want to make manual adjustments to routes to adapt to urgent needs or unexpected changes.
    - As a manager, I want the system to dynamically reroute based on parameter changes after the route is generated.
    - As the system, I need to recalculate routes dynamically if the manager makes changes to parameters, ensuring updated routes without disrupting existing operations.

- **Route Sharing and Export**
  - **Description:** Facilitate the sharing of finalized routes with field staff.
  - **Actions:**
    * As a manager, I want to export and share finalized routes with field staff so they have clear guidance on their daily tasks and destinations.
    * As the system, I need to support data export so that managers can download route plans and reports for offline access if needed.
- **Store Profile Integration and Viewing**
  - **Description:** Provide access to detailed store profiles to aid in route planning.
  - **Actions:**
    * As a manager, I want to view detailed store profiles, including location and sales channel, to prioritize visits and plan routes effectively.
    * As the system, I need to integrate and maintain comprehensive store profiles, making it easier for managers to access relevant data during route planning.
- **Interactive Map Visualization**
  - **Description:** Enable map-based visualization of routes and store locations.
  - **Actions:**
    * As a manager, I want to see routes on an interactive map, allowing me to visualize store locations and make adjustments as needed.
    * As the system, I need to display routes and navigation data on an interactive map interface for both managers and field staff.
- **Performance Tracking and KPI Monitoring**
  - **Description:** Track and analyse performance metrics to evaluate route effectiveness.
  - **Actions:**
    * As a manager, I want to track KPIs like travel time and visit completion rates to assess route performance and make improvements.

* As the system, I need to automate KPI tracking (e.g., average travel time, total distance, workload) to provide ongoing insights without requiring manual input from managers.

- **System Data Synchronization**
  - **Description:** Sync store and route data to ensure accurate and up-to-date information.
  - **Actions:**
    * As a manager, I want the system to view store information and modify store statuses, giving me reliable data for decision-making.
    * As the system, I need to synchronize with external databases to maintain current and accurate store and route data for all users.

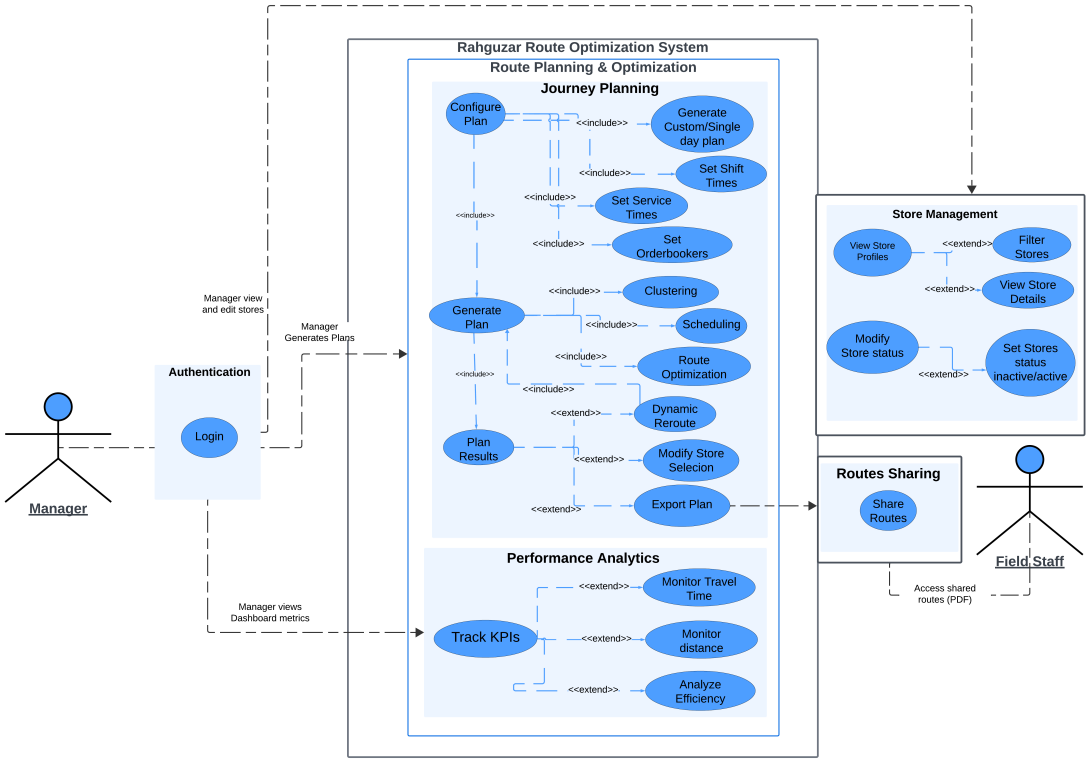

Figure 3.10: Rahguzar's Use Case Diagram

3.5 Datasets

This section outlines the primary datasets that will support Rahguzar's intelligent route planning and scheduling functionalities. These datasets provide crucial information on stores, visits, journey plans, and operational factors essential for optimized route generation. As per the NDA with SalesFlo, we are not permitted to share or disclose the data provided to us.

Overview of Datasets

1. **Store Universe**
   This dataset provides foundational information on each store, including location, operational status, and distributor associations. Key fields include:

   - **storecode:** Unique identifier for each store.
   - **storestatus:** Operational status (active or inactive).
   - **latitude and longitude:** Geographic coordinates.
   - **distributorcode:** Code for the distributor linked to the store.
   - **pjpcode:** Code for the Permanent Journey Plan (PJP) associated with the store.

   This dataset is essential for determining which stores to include in routes based on location and operational status.

2. **Store Hierarchy**
   This dataset offers classification and hierarchical details, allowing route customization based on store attributes such as region or sales channel. Key fields include:

   - **storecode:** Identifier for each store.
   - **areatype, townid, localityid:** Geographical identifiers.
   - **channeltypeid and channelid:** Sales channel identifiers (e.g., retail, wholesale).
   - **storeclassificationIDs:** These classification IDs will be used to further distinguish between certain store categories.

   This data enables Rahguzar to organize routes by region and sales channel, optimizing for specific geographic and operational criteria.

3. **Visits**
   The Visits dataset records historical visit information for each store, including time spent, visit order, and outcomes. Key fields include:

   - **visitid:** Unique identifier for each visit.
   - **pjpcode:** Journey plan code associated with the visit.
   - **visitdate, visitstarttime, visitendtime:** Time data for analyzing visit durations.
   - **visitspenttimeinseconds:** Total time spent at the store during the visit.
   - **orderstatus:** Status of any order placed during the visit.
   - **visiststatus:** Status determining if a visit has been completed or not.
   - **syncdown, syncup, syncdowndatetime:** These fields track the synchronization status and timing of visit records: syncdown shows if data was downloaded to the device, syncup if it was uploaded to the server, and syncdowndatetime logs the last download timestamp.

   This dataset supports route optimization by providing insights into visit frequency and duration, refining future scheduling.

Data Utilization in Rahguzar

Together, these datasets will support Rahguzar's core functionalities, including:

- **Efficient Route Planning:** Leveraging store locations, operational statuses, and road networks for optimal routing.
- **Traffic and Geographic Optimization:** Utilizing geographic data to plan routes that avoid congestion and optimize for specific zones or territories.
- **Route Efficiency Analysis:** The Visits dataset will provide insights into average visit durations and travel times, which will inform the route optimization algorithm and support KPI tracking for system performance.

3.6 System Diagram

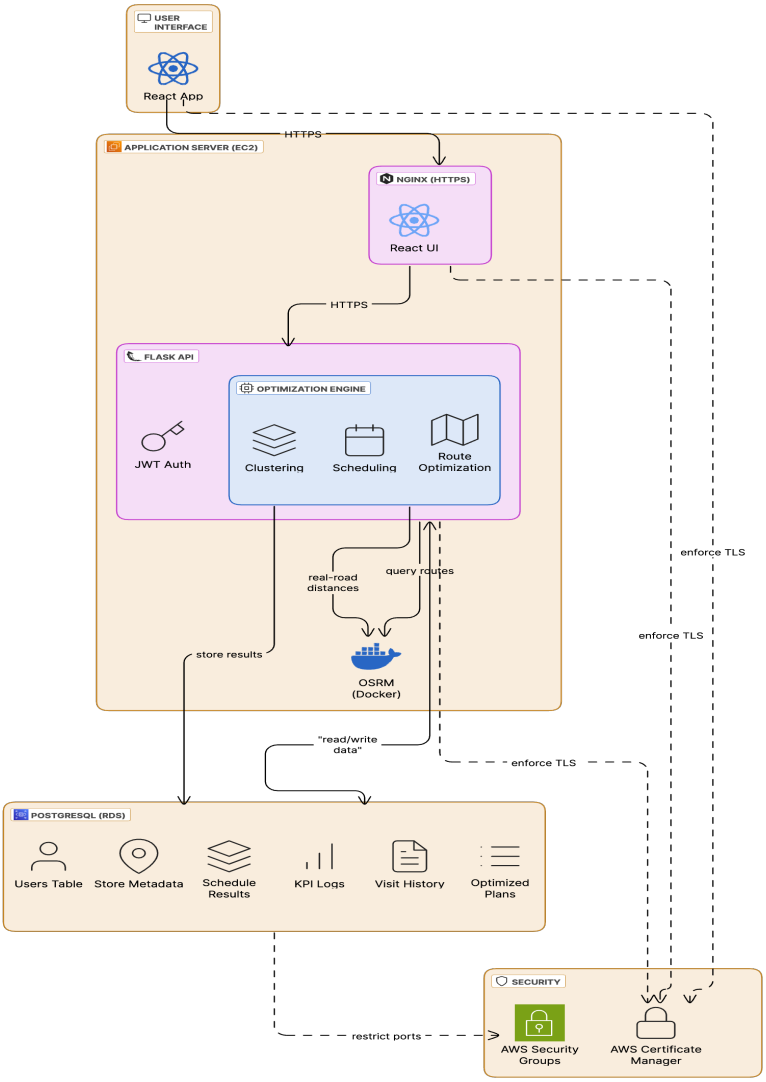

Figure 3.11: Rahguzar's System Architecture Diagram

4. Software Design Specification (SDS)

The Software Design Specifications outlines the key design components of the Rahguzar system, including its software architecture, database schema, and core optimization logic.

4.1 Software Design

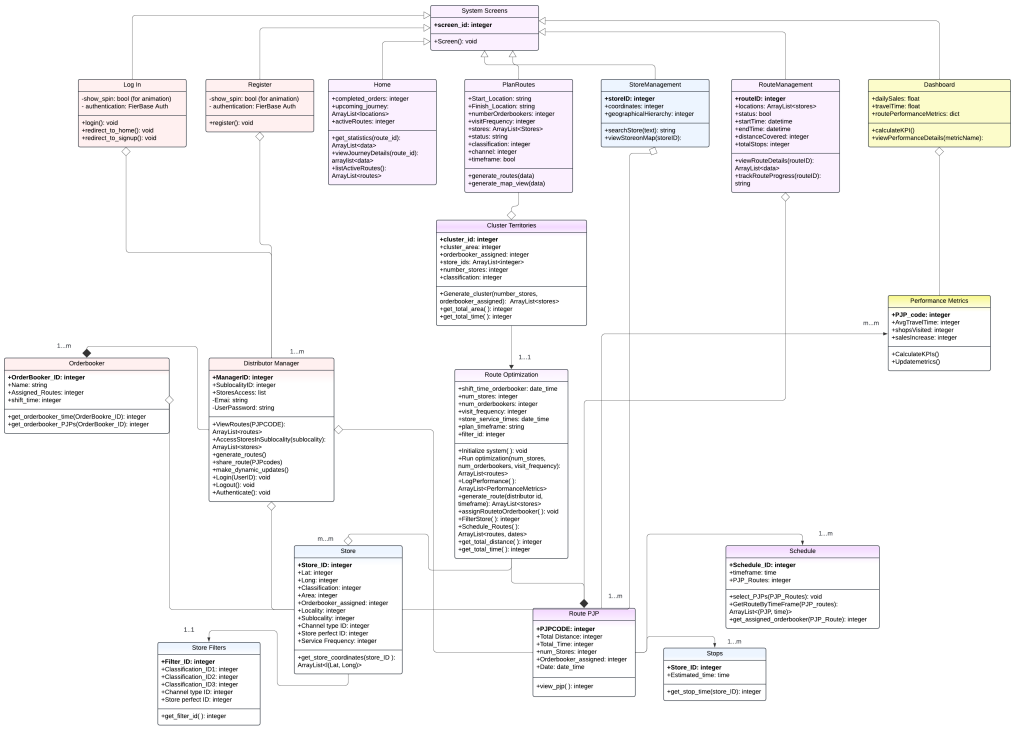

Figure 4.1: UML Class Diagram

As shown in Figure 4.3, the UML diagram starts off by showing the screens that are connected to the system. The Register screen creates the object for the distributor manager, giving them access to the system and screens mentioned such as home, plan routes, route management, and etc. Next, the user has the option to Plan Routes from that screen where they give certain parameters mentioned in the attributes. This creates a cluster object for different areas containing different stores. Next, the optimal PJP route will be determined in each cluster that creates the object for clusters. After this, the stores in each cluster will be connected to form a route, creating a routes class that is the optimal PJP plan. These routes can also be viewed on a schedule view to visually analyze how the upcoming PJPs look like. Moreover, a separate stores table is connected to the child table of stores filters that provides the further classification of each store id. Lastly, the dashboard screen gets the KPIs from the Performance Metrics class to display how the performance of each PJP is and to visualize it in graphs and display the metrics.

4.2 Data Design

This section presents the structure of our database that caters to persistent data storage in our project. The structure is shown as a normalized data model for relational databases. It clearly shows entities, attributes, relationships with their cardinalities, and primary and foreign keys. We have used Postgresql ERD to build our data model.

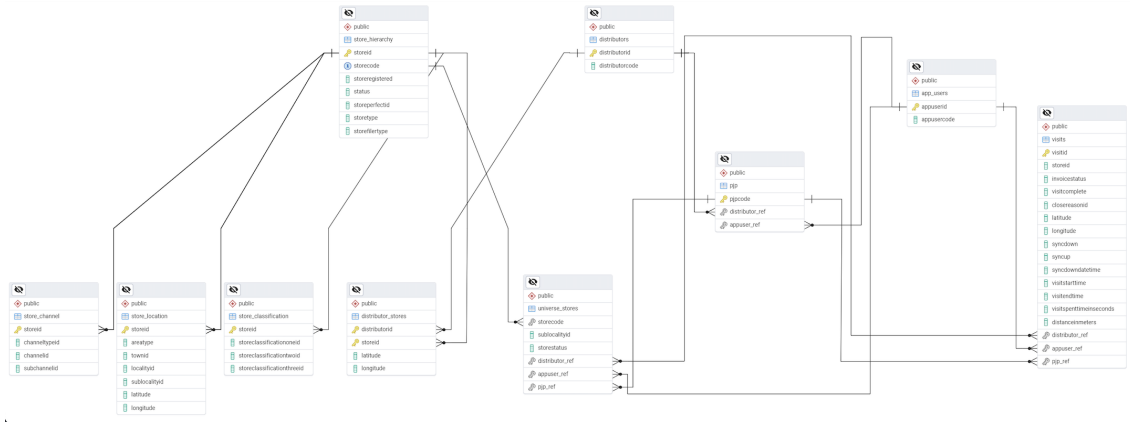

Figure 4.2: ERD Diagram

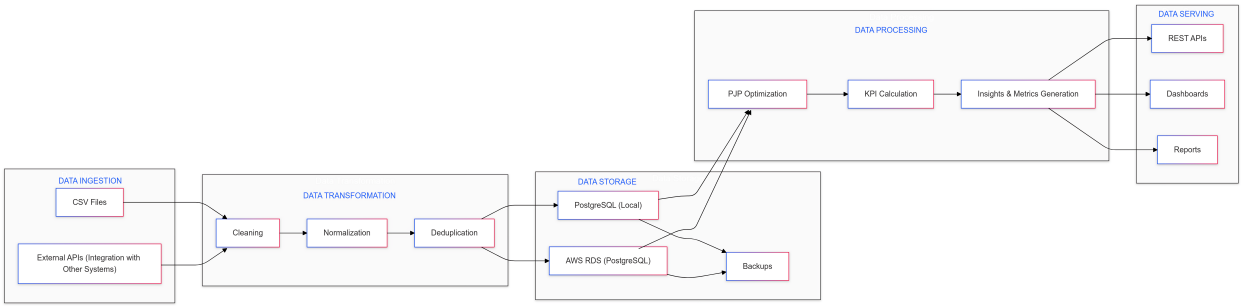

Figure 4.3: End-to-end data pipeline showcasing ingestion, transformation, storage, processing, and serving

4.3 Technical Details

The Rahguzar system uses a multi-stage route optimization algorithm composed of clustering, scheduling, and routing. This pipeline ensures operational efficiency while adhering to business constraints such as working hours, visit frequencies, and travel limits.

- **Clustering:** Stores are grouped based on geographic proximity using density- or centroid-based clustering.
- **Scheduling:** An Evolutionary Algorithm assigns stores to order bookers across a multi-day plan, optimizing workload balance and visit coverage.
- **Routing:** Store sequences are optimized using road-based routing with OSRM, minimizing total travel time and distance.

The algorithm aims to maximize shop visits, minimize total distance traveled, and reduce overall time, including both travel and in-store duration. It operates under key constraints such as the number of order bookers, shift timings, and fixed visit durations per store, ensuring realistic and efficient route plans.

The system supports dynamic rerouting, manual overrides, and constraint-based optimization to remain adaptive to real-world field operations. Full methodological details including algorithmic structure, fitness design, and tuning are provided in the Methodology section.

5. Development

The development of Rahguzar was guided by the goal of building a robust, scalable, and user-friendly platform capable of automating Permanent Journey Plan (PJP) scheduling in FMCG field operations. The project was developed in phases, each aligning with different milestones from backend API construction to frontend deployment, database integration, and cloud migration.

5.1 Backend Development and API Architecture

The backend of Rahguzar was developed using Flask, a lightweight Python-based microframework. Flask was chosen for its flexibility and simplicity, allowing us to quickly develop RESTful APIs that catered to the core functionalities of route generation, user management, and data synchronization. The API layer served as the communication bridge between the frontend and the backend logic, as well as with third-party services.

The route optimization engine was integrated into this backend stack using custom Python scripts for solving routing problems. This engine consumed input parameters—such as store locations, visit frequencies, shift timings, and the number of available order bookers—and returned optimized PJPs. These responses were served through API endpoints that exposed functionalities for plan generation, route export, real-time updates, and manual overrides.

Throughout development, emphasis was placed on designing clean, modular API endpoints. Authentication and authorization mechanisms were implemented using JSON Web Tokens (JWT), ensuring secure access control for different types of users, particularly field managers.

5.2 Frontend Development with React

The frontend was implemented using React, which offered a highly responsive and component-driven interface for interacting with Rahguzar's planning and monitoring features. React's ability to efficiently manage state and update the DOM in real-time was crucial for building features like dynamic map rendering, drag-and-drop store reordering, and dashboard KPI tracking.

The user interface included several key screens such as the sign-in and registration pages, a dashboard with route statistics, a map-based route planner, and modules for viewing stores and journeys. Special attention was given to providing managers with a seamless experience for planning, editing, and visualizing routes, supported by an interactive map.

Frontend and backend integration was handled via asynchronous API calls using Axios. These calls ensured real-time feedback during route planning sessions, allowing users to view updated PJPs instantly after adjusting route parameters or filters.

5.3 Database Design and Integration

Initially, PostgreSQL was selected as the relational database for managing store profiles, visit logs, user data, and system configurations. The schema was normalized and designed to accommodate the system's growing need for relational queries, especially during the optimization phase where data consistency and referential integrity were crucial.

The data design followed an entity-relationship model, with core tables representing stores, visits, routes, and users. Each route was linked to a cluster of stores and assigned to specific order bookers based on availability and priority. Visit records captured timestamps, sync states, and performance metrics for later analysis.

During development, we encountered limitations with local hosting, especially as the dataset size and user load increased. This led to the decision to migrate the database to a cloud-hosted environment.

5.4 Migration to AWS and Cloud Deployment

To ensure cost-effective scalability and streamlined management, the Rahguzar system was migrated to Amazon Web Services (AWS). All core components—including

the frontend interface, backend logic, routing engine, and database client—were deployed on a single `t3.medium` EC2 instance. This consolidated deployment simplified system configuration, reduced latency between services, and ensured consistent performance during pilot testing.

The frontend, built using React, was compiled locally and deployed on the EC2 instance using Nginx to serve static assets. This allowed for efficient content delivery and responsive user interaction, even during concurrent access by multiple users. The backend, implemented in Flask, was hosted on the same instance and served via Gunicorn. It acted as the central controller for all major operations including store clustering, Evolutionary Algorithm-based scheduling, route optimization, and communication with both the routing engine and the database.

The Open Source Routing Machine (OSRM) was containerized using Docker and also deployed on the EC2 instance. It was used to compute real-world road routes and travel times between store locations. The backend accessed OSRM via local RESTful API calls, retrieving optimized polylines and distance estimates that were directly integrated into the scheduling logic and frontend map rendering. OSRM was configured to run privately within the EC2 environment, ensuring that routing operations remained secure and isolated from public access.

In parallel with the EC2 setup, the PostgreSQL database was migrated from a local development environment to Amazon RDS. The migration process involved exporting local snapshots and importing them into a managed PostgreSQL instance on RDS. Following the migration, the Flask backend was reconfigured to connect securely to RDS using environment-secured credentials. To ensure security, inbound traffic to the RDS instance was restricted to only allow requests from the EC2 instance. The use of RDS provided several operational advantages, including automated backups, failover recovery, and real-time performance monitoring through AWS CloudWatch. These features contributed to improved database reliability and enhanced query performance, especially during intensive route generation and dashboard analytics tasks.

All inter-service communication occurred within the EC2 instance or over secured private channels, with environment variables used to manage API keys and database credentials securely. Although this architecture runs all services on a single machine, it is designed for modular scaling. As system usage grows, individual components can be independently scaled or containerized further without requiring architectural changes. This deployment forms a stable, cloud-native foundation that is capable of supporting Rahguzar's expansion into multiple regions and cities within the FMCG domain.

5.5 Integration with External Services

The Rahguzar system integrates key external services to enable accurate, real-world route optimization and dynamic geospatial visualization. These integrations play a critical role in both the backend logic responsible for route computation and the frontend interface that supports interactive user experiences.

The Open Source Routing Machine (OSRM) is deployed as a Docker container on the same AWS EC2 instance that hosts the rest of the system. Serving as the core routing engine, OSRM processes map-based routing queries using preprocessed OpenStreetMap (OSM) data. It computes real-world travel distances and optimal driving paths between store locations, which are essential inputs for the route optimization workflow. The Flask backend communicates with OSRM through RESTful API calls using endpoints. These endpoints return data including total travel time, total distance, and a GeoJSON-encoded polyline representing the optimized route geometry. This data is used to generate routes that reflect real driving conditions and constraints, enhancing the operational feasibility of the journey plans.

On the frontend, Leaflet.js is used to render interactive maps within the React interface. It visualizes optimized routes, displays store markers, and supports features such as polygon selection and zoom-based filtering. The route geometry returned by OSRM integrates seamlessly with Leaflet's polyline rendering features, enabling the display of accurate road-based paths instead of abstract straight-line connections. Additional customization, such as colored cluster markers and dynamic route overlays, is implemented using Leaflet's plugin architecture and open API design. Together, the integration of OSRM and Leaflet enables a robust map-based planning experience for managers and provides clear, actionable route visualization for field teams.

5.6 Testing and Deployment

Throughout development, both unit and integration tests were written to ensure API stability and frontend-backend communication integrity. Postman and pytest were used for backend testing, while Cypress and manual QA testing supported frontend validation.

The final deployment used an EC2-based architecture for the backend Flask server, React served via an S3 bucket and CloudFront, and PostgreSQL managed through AWS RDS. This cloud-based setup ensured high uptime, fast response times, and readiness for real-world deployment at scale.

6. Methodology

The methodology for Rahguzar was designed as a structured, research-driven response to the operational inefficiencies in manual Permanent Journey Plan (PJP) scheduling. The project aimed to generate optimized PJPs that adhered to real-world constraints in the FMCG distribution landscape of Pakistan. Given the scale and complexity of the problem, we adopted a modular and algorithmically hybrid approach, structured into three main phases: Clustering, Scheduling, and Route Optimization.

Our methodology combined custom algorithm development with empirical evaluation on real-world datasets. This allowed us to balance theoretical rigor with practical usability, ensuring that the final solution could be deployed under real operational conditions.

6.1 Problem Formulation

The PJP problem is modeled as a weekly, constraint-aware variation of the VRPTW. Given a set of stores with known coordinates, visit frequencies, service durations, and territory assignments, the objective is to generate an optimized journey plan for each order booker such that:

- Store visits satisfy required weekly frequencies
- Visits are spaced evenly across working days
- Total route time (travel + service) per day remains within shift limits (typically 8 hours)
- Order booker workload is balanced across the team
- Routes are geographically compact and operationally realistic

Due to the NP-hard nature of the problem, exact optimization methods are impractical at city scale. Instead, we use a hybrid, heuristic-based approach structured into three phases.

6.2 Three-Phase Algorithmic Architecture

6.2.1 Clustering

The clustering phase assigns stores to order bookers through a hybrid clustering pipeline that balances geographic proximity with workload equity. This process allows the system to divide a large set of stores into manageable groups, which can then be assigned to specific order bookers in a logical and efficient manner.

Graph-Based Clustering. A Minimum Spanning Tree (MST) is constructed using store location data. The MST connects all stores with the minimal possible total edge weight, ensuring that geographically close stores are grouped together with minimal travel distance. This forms the initial clusters that are further refined in the next steps.

K-Means Refinement. Each graph-based cluster is refined by minimizing the internal variance within the cluster using the K-means algorithm. The centroid C_i of each cluster is recalculated using:

$$C_i = \frac{1}{N_i} \sum_{x \in S_i} x$$

where:

- C_i: The new centroid of cluster i
- N_i: The number of stores in cluster i
- $x \in S_i$: The coordinates of each store x in cluster S_i

Outlier Reassignment. After initial refinement, stores that lie significantly far from their assigned cluster centroid are identified using:

$$D_{s,c} > \mu + \sigma \cdot k$$

where:

- $D_{s,c}$: Distance of store s from the centroid of its assigned cluster c

- μ: Mean distance of all stores in the cluster from the centroid
- σ: Standard deviation of the distances
- k: Scaling factor used for thresholding

Such outliers are then reassigned to the nearest neighboring cluster. This improves overall cluster consistency and prevents inefficiencies in route planning due to geographical anomalies.

Workload Balancing. To ensure equitable distribution of stores across order bookers, workloads are balanced by minimizing the squared workload error:

$$SSE = \sum_{c} (W_c - \bar{W})^2$$

where:

- W_c: Total workload of cluster c
- $\bar{W}$: Average workload across all clusters

This multi-step clustering strategy ensures both spatial efficiency and fair workload distribution across field staff. The use of MST helps with initial proximity grouping, K-means ensures tight clusters, outlier reassignment handles anomalies, and workload balancing ensures equity. Together, they set a strong foundation for downstream scheduling and routing.

6.2.2 Scheduling

After clustering, each order booker's list of stores is scheduled across the working week using a custom Evolutionary Algorithm (EA). This approach is designed to distribute store visits over time in a way that balances efficiency with business constraints.

Population Initialization. Initial schedules are generated randomly and based on geographic heuristics. This combination provides enough diversity for the EA to explore a wide range of configurations and forms the basis for iterative improvements.

Fitness Evaluation. Each schedule is evaluated using the following fitness function:

$$F = T_{\text{total}} + P_{\text{mismatch}} + P_{\text{imbalance}} + P_{\text{geo}}$$

where:

- T_{total}: Total time taken for travel and service across all days
- P_{mismatch}: Penalty for not meeting store visit frequency
- $P_{\text{imbalance}}$: Penalty for workload imbalance across order bookers
- P_{geo}: Penalty for geographical inefficiencies in route shape

This function balances multiple scheduling objectives. A lower score indicates a more optimal schedule under all constraints.

Constraints. Daily work limits are enforced using:

$$T_{\text{day}} = \sum_{i=1}^{n} T_{\text{travel},i} + \sum_{i=1}^{n} T_{\text{service},i} \leq 480$$

where:

- $T_{\text{travel},i}$: Travel time to store i
- $T_{\text{service},i}$: Service time at store i
- n: Number of stores in the daily route

This ensures feasibility within an 8-hour workday and aligns schedules with operational constraints.

Crossover and mutation operations are applied to refine the population of schedules over successive generations. Tournament selection is used to identify high-performing candidates, from which crossover generates new schedules by combining elements of the best solutions. Mutation then introduces small random changes to maintain diversity in the population and prevent premature convergence to local optima. Together, these mechanisms drive the evolutionary algorithm toward more optimal and robust scheduling outcomes.

6.2.3 Route Optimization

For each day's assigned store list, the visit sequence is optimized using a two-step approach: a heuristic-based Nearest Neighbor (NN) method followed by optimization with the OR-Tools Traveling Salesman Problem (TSP) Solver.

Step 1: Nearest Neighbor (NN). A route is constructed starting from the depot by repeatedly selecting the nearest unvisited store. This provides a quick, feasible route with low computational cost, suitable for initialization.

Step 2: OR-Tools TSP Solver. The initial sequence is refined by minimizing the tour length:

$$\min \sum_{i=1}^{r-1} D_{i,i+1} + D_{r,1}$$

where:

- r: Total number of stores in the daily route
- $D_{i,i+1}$: Distance between consecutive stores i and $i+1$
- $D_{r,1}$: Distance from the final store back to the depot

Thus, the routes are constrained to a maximum daily duration of 480 minutes. This two-step method combines the speed of Nearest Neighbor with the precision of OR-Tools, ensuring route quality, feasibility, and scalability for FMCG operations.

6.3 Data Preprocessing Strategy

The input datasets, provided by SalesFlo in Excel format, were preprocessed using a structured pipeline to ensure data quality and consistency. Parsing and cleaning were performed using Python, followed by merging multiple files based on unique store identifiers. The cleaned and integrated data was then migrated into a PostgreSQL database to enable efficient querying and relational operations. For the pilot implementation, stores were filtered by predefined operational zones to restrict testing to relevant geographic clusters. This preprocessing ensured that the data was normalized, reliable, and ready for direct integration into the clustering, scheduling, and routing optimization pipeline.

6.4 Assumptions

To simplify computation and focus on core scheduling and routing challenges, several assumptions were applied throughout the system design. Travel times were treated as static averages, without modeling real-time traffic conditions. All routes were assumed to start and end at a fixed distributor location. Store service durations were considered known and constant across the planning horizon. Additionally, planning was conducted on a weekly basis, excluding Sundays, to align with standard operational cycles. These assumptions allowed the algorithmic components to remain tractable while preserving alignment with real-world FMCG workflows.

6.5 Algorithmic Complexity

Rahguzar's modular pipeline breaks the PJP problem into three heuristically optimized stages: clustering, scheduling, and routing. While the overall problem is NP-hard, each component is designed to ensure scalability through local heuristics and bounded iterations.

- **Clustering:** Graph-based clustering is performed using pairwise haversine distances with complexity $O(n^2)$, followed by K-Means refinement ($O(nki)$) and outlier reassignment ($O(n)$). This phase is executed once and scales well for realistic values of n (number of stores) and k (number of order bookers).

- **Scheduling (EA):** The Evolutionary Algorithm runs for G_{EA} generations (typically 80–100) with a population size P (typically 40–50). Fitness evaluation includes total route time, visit frequency matching, and workload balancing. Complexity per generation is $O(P \cdot n)$, giving:

$$O(P \cdot n \cdot G_{EA})$$

- **Routing (Nearest Neighbor + OR-Tools):** For each day, an initial path is constructed using the Nearest Neighbor (NN) heuristic. This is followed by route refinement using Google's OR-Tools TSP solver, which minimizes the total travel distance:

$$\min \sum_{i=1}^{r-1} D_{i,i+1} + D_{r,1}$$

  For r stores per route across d daily routes, the empirical time complexity is approximately:

$$O(d \cdot r^2)$$

  This two-step routing ensures scalable and high-quality solutions for daily planning under operational constraints.

Total System Complexity:

Combining the three stages, the total time complexity of Rahguzar's algorithm is:

$$O(n^2 + nk + P \cdot n \cdot G_{EA} + d \cdot r^2)$$

This formulation reflects both theoretical bounds and empirical performance. In practice, early stopping, parallelism, and per-cluster decomposition significantly reduce runtime. Rahguzar consistently generated optimized plans for hundreds of stores in under 5 minutes on a standard multi-core machine.

7. Experiments and Results

This chapter aims to present the results of the experiments conducted across Rahguzar's clustering, scheduling, and routing phases to evaluate their impact on route optimization, workload balancing, and service coverage. Each algorithm was tested individually and as part of the integrated pipeline. Additionally, Rahguzar's performance was benchmarked against Salesflo using real distributor datasets, comparing outcomes in terms of total travel distance, execution time, and overall planning efficiency.

7.1 Algorithmic Phases and Experiment Results

This section presents the methodology and results from a series of experiments conducted to determine the optimal combinations of algorithms for different phases of the optimization process.

7.1.1 Phase 1: Hybrid Clustering Approach

The first aim of the algorithm is to divide the list of stores to be serviced into a number of clusters which is determined by the number of orderbookers. The objective is to group close stores in the same cluster, so that the orderbookers can service them as the scheduler will decide. To achieve this, clustering algorithms were tested on their own adn in a hybrid approach.

Clustering Algorithms Tested

The following clustering algorithms were evaluated for grouping stores among orderbookers based on geographical proximity and workload balancing:

- **KMeans Clustering**

  Groups stores based on latitude and longitude using the KMeans algorithm. It minimizes the sum of squared distances to cluster centroids. After initial

clustering, stores are reallocated from overloaded to underloaded clusters to ensure balanced workload distribution.

- **Hierarchical Clustering**

  This method uses Ward's linkage to merge clusters based on minimum variance increase. Clusters are formed by cutting the dendrogram at the level corresponding to the number of orderbookers, resulting in compact and cohesive geographic groupings.

- **Gaussian Mixture Models**

  Clusters are modeled as Gaussian distributions with flexible shapes and orientations. Stores are assigned to clusters based on their probability of belonging to a particular Gaussian component. The model uses "full" covariance to allow for non-spherical clusters.

- **Hybrid Approach 1 (Graph-Based Clustering + KMeans)**

  This approach begins by forming clusters based on graph connectivity within a specific distance threshold. These clusters are then refined using KMeans to match the desired number of clusters, ensuring both geographic cohesion and count alignment.

- **Hybrid Approach 2 (Graph-Based Clustering + KMeans + Geographical Constraints)**

  This builds on Hybrid 1 by identifying and reassigning outlier stores. Outliers are detected based on their distance from the cluster centroid exceeding a defined threshold. These stores are then reassigned to the nearest appropriate cluster to enhance geographic cohesion and workload balance.

Moreover, to ensure that optimal clusters were being made, cluster balancing was employed to redistribute stores between clusters to ensure workloads—based on service effort and travel time—are roughly equal. Overloaded clusters transfer nearby stores to underloaded ones, and this process repeats until workloads fall within a set tolerance or a maximum number of iterations is reached, promoting fair and efficient distribution.

7.1.2 Phase 1: Experiments Results

The silhouette score measures how well points are grouped in a clustering task. It checks if points are close to others in their own cluster, indicating a good score, or if

they are far from points in other clusters (also good score). The score ranges from -1 to 1, where 1 means perfect clusters, 0 means overlap, and negative scores indicate that points might be in the wrong cluster.

Distributor ID	KMeans	Gaussian Mixture Models	Hierarchical	Hybrid Approach-1	Hybrid Approach-2
1	0.3537	0.2835	0.3537	0.5044	0.5044
6	0.1839	0.2130	0.1839	0.1839	0.1684
7	0.2737	0.3376	0.3892	0.3980	0.4437

Table 7.1: Silhouette Score Comparison for Different Clustering Algorithms

From the results in Table 7.1, it can be observed that the Hybrid Approach-2 outperformed all other clustering algorithms, achieving the highest silhouette score of 0.5044 for Distributor 1 and 0.4437 for Distributor 7. This indicates that the clusters formed using this approach were more cohesive and well-separated compared to those formed by the other algorithms. Moreover, it finds a good balance between workload distribution and geographic proximity. For example, for distributor ID 7, there's a little variance in the clusters being formed however, it has the highest silhouette score as geographical proximity has been set as the priority.

7.1.3 Phase 2: Scheduling with Evolutionary Algorithm (EA)

Once clusters have been formed in Phase 1, the next critical step is to generate a feasible schedule that defines which shops should be visited on each day of the planning period (either a single day or a custom range, as selected by the user). The primary objective during scheduling is to balance the workload across both the available days and all assigned orderbookers. This ensures that no orderbooker is disproportionately burdened and that their daily tasks are distributed as evenly as possible.

To identify the most effective scheduling strategy, a series of experiments were conducted using various algorithms. Each algorithm was evaluated based on its ability to generate balanced, constraint-compliant schedules. At the core of this evaluation process is the fitness function.

The fitness function plays a vital role in determining how practical and efficient a schedule is. It computes a cost value for each candidate schedule—lower values indicate better solutions. Hard constraints are applied first: for example, if the total route time on any day exceeds a predefined daily limit (e.g., 480 minutes), a substantial penalty is applied to immediately discourage infeasible schedules. It also checks for visit mismatches—stores not visited the required number of times—penalizing any deviations.

Additionally, soft constraints are incorporated to further refine the schedule. These include a day-balancing penalty when daily route times fall outside the ideal range (e.g., 360–420 minutes), and a geographical penalty when neighboring stores requiring only one visit are assigned to different days. These constraints collectively guide the algorithm toward generating schedules that are not only valid but also optimized for balance, practicality, and real-world efficiency. The following scheduling algorithms were used:

- Simulated Annealing
- Ant Colony Optimization (ACO)
- Mixed Integer Linear Programming (MILP)
- Evolutionary Algorithm (EA)

Through a series of experiments, the best combination for the scheduling phase was determined, which was Evolutionary Algorithm (EA) for scheduling. Algorithms like ACO showed that the in serach for the optimal schedule, it can often get trapped in a local optima. For MILP and simulated annealing, generating a schedule can be computationally expensive and time consuming. On the other hand, EA showed optmial results with genetic diversity maintained that avoids premature convergence, hence EA is the best choice for scheduling.

7.1.4 Phase 3: Route Optimization

After a schedule has been created, the next step is to optimize the routes for each orderbooker. This involves determining the most efficient path for each orderbooker to follow, ensuring that they can visit all assigned stores in the shortest possible time while adhering to any constraints (e.g., time windows, vehicle capacity). The goal is to minimize travel distance and time while maximizing efficiency. This phase is crucial for ensuring that the orderbookers can complete their routes within the allocated time and resources, ultimately leading to improved service levels and reduced operational costs.

Moreover, it helps determine the order of visiting the stores and the best route to take, considering factors such as road networks and other logistical constraints. For route optimization, several algorithms were tested to identify the most effective one. The route optimization algorithms evaluated were:

- Particle Swarm Optimization (PSO)

- Ant Colony Optimization (ACO)
- Google Optimization Tools (OR-Tools)
- Evolutionary Algorithm (EA)
- Dynamic Programming

Among the tested route optimization algorithms, Google OR-Tools showed the most consistent and optimal performance in minimizing total travel distance and time. While ACO and PSO showed promise in smaller test cases, they frequently got trapped in local optima or exhibited longer computation times in large-scale datasets. OR-Tools, by contrast, delivered high-quality solutions across varying distributor profiles and constraints, making it the most robust choice for route optimization within Rahguzar's pipeline.

7.1.5 Phase 2 and 3: Experiments Overview

A series of experiments were conducted across different distributors with varying numbers of stores and orderbookers. The following distributors were involved in the experiments:

Distributor ID	**Number of Stores**	**Number of Orderbookers**
1	395	2
6	426	4
7	580	5
131	42	1

Table 7.2: Distributors and their Assigned Stores and Orderbookers

The purpose of the experiments was to determine the best combinations of scheduling and route optimization algorithms for each distributor. The experiments were designed to evaluate the performance of different algorithmic combinations in terms of distance and travel time minimization. This was done by analyzing the total distance and the total traveltime minimized across all orderbookers (results show average of all distributors), and the stores serviced distribution per orderbooker for each day.

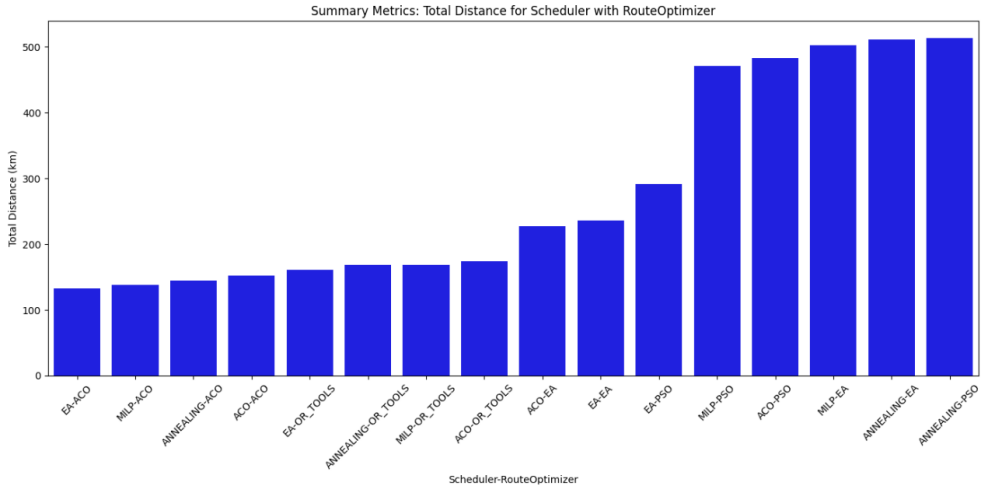

Figure 7.1: Total distance for all distributors compared over different combinations.

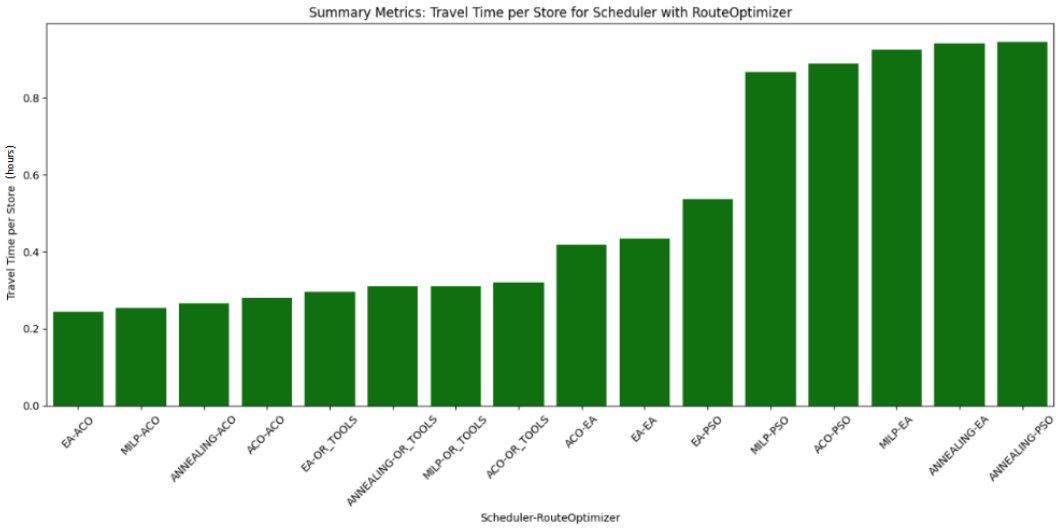

Figure 7.2: Total time for all distributors compared over different combinations.

The results in Figure 7.1 and Figure 7.2 show that one of the top-performing scheduler-route optimizer combinations was EA-OR Tools, which demonstrated strong results in minimizing both travel distance and time, by traveling to as less as 150 km in 0.2 hours, as compared to almost 500 kms and 1 hour for the worst combination. OR Tools consistently ranked among the top-performing route optimizers

across multiple test cases, whereas the least optimal performance was showed EA nad PSO route optimizers, and MILP and simulated annealing schedulers.

7.1.6 Phase 2 and 3: Experiment Results

The experiments were conducted by combining different scheduling algorithms with different route optimization algorithms. The findings of each graph is summarized below.

The best performing scheduler-route optimizer combinations for each distributor are as follows:

Table 7.3: Best Performing Scheduler-Route Optimizer Combinations by Distributor

Distributor	**Best Distance Minimized Scheduler-Route Optimizer**	**Best Travel Time Minimized Scheduler-Route Optimizer**
Distributor 7	EA-OR-Tools	EA-OR-Tools
Distributor 6	EA-OR-Tools	EA-OR-Tools
Distributor 1	EA-OR-Tools	EA-OR-Tools
Distributor 131	EA-ACO	EA-ACO

7.2 Results Comparison: Salesflo PJP vs Rahguzar PJP

This section presents a comparative analysis of Rahguzar's performance against Salesflo's pre-existing PJP plans across three real-world test distributors. Each distributor varied in terms of store count, OB allocation, and planning duration, allowing for a diverse evaluation of route efficiency, time optimization, and system scalability. Metrics analyzed include total distance traveled and total route time under the same operational constraints for both systems.

7.2.1 Test Distributor 1 – Results Comparison

To evaluate Rahguzar's performance, the first distributor involved 365 stores serviced by 2 Order Bookers (OBs) over a 3-day plan. The results are summarized below:

Table 7.4: Performance Comparison – Test Distributor 1

Metric	Salesflo	Rahguzar
Total Shops	365	365
Order Bookers	2	2
Plan Duration (days)	3	3
Total Distance (km)	227.87	218.77
Total Time (hrs)	35.53	35.39

Rahguzar reduced the total travel distance by **9.1 km** and total time by **0.14 hours**, reflecting modest spatial and temporal efficiency.

7.2.2 Test Distributor 2 – Results Comparison

The second distributor included 1153 stores and 3 OBs over a 6-day schedule:

Table 7.5: Performance Comparison – Test Distributor 2

Metric	Salesflo	Rahguzar
Total Shops	1153	1153
Order Bookers	3	3
Plan Duration (days)	6	6
Total Distance (km)	929.25	545.01
Total Time (hrs)	113.77	108.54

Rahguzar achieved a reduction of **384.24 km** in distance and **5.23 hours** in time, demonstrating high scalability and optimization capability.

7.2.3 Test Distributor 3 – Results Comparison

This case involved 659 stores and 2 OBs, also over a 6-day cycle:

Table 7.6: Performance Comparison – Test Distributor 3

Metric	Salesflo	Rahguzar
Total Shops	659	659
Order Bookers	2	2
Plan Duration (days)	6	6
Total Distance (km)	143.66	142.46
Total Time (hrs)	58.95	58.99

Rahguzar reduced the distance by **1.2 km**, with a slight increase in time of **0.04 hours**, indicating comparable efficiency.

7.2.4 Summary Across All Distributors

To illustrate overall performance across all scenarios:

Table 7.7: Combined Performance Summary

Distributor	Platform	Total Distance (km)	Total Time (hrs)
1	Salesflo	227.87	35.53
	Rahguzar	218.77	35.39
2	Salesflo	929.25	113.77
	Rahguzar	545.01	108.54
3	Salesflo	143.66	58.95
	Rahguzar	142.46	58.99

Across all test cases, Rahguzar reduced the cumulative travel distance by **540.79 km** and time by **5.41 hours**, validating its effectiveness in optimizing field operations.

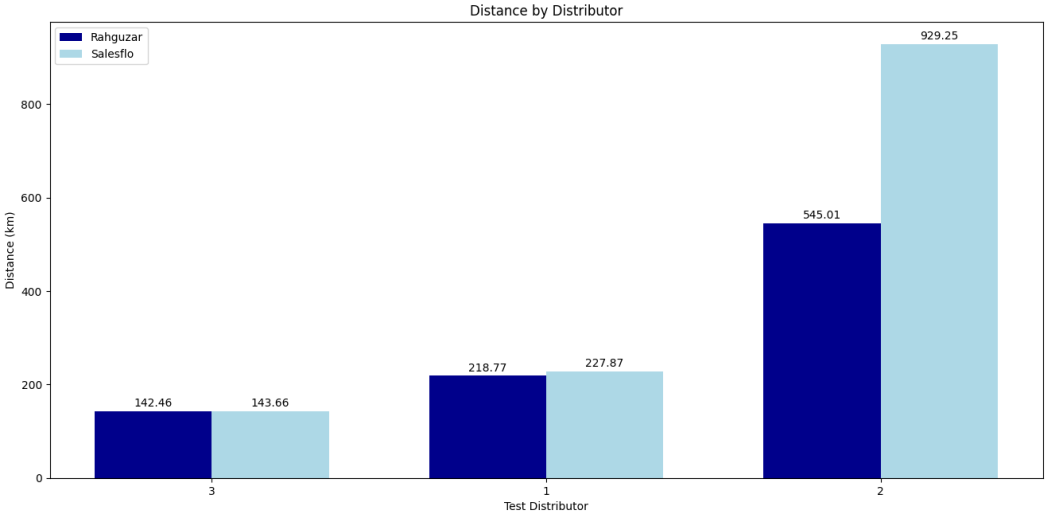

Figure 7.3: Total distance comparision for the test distributors.

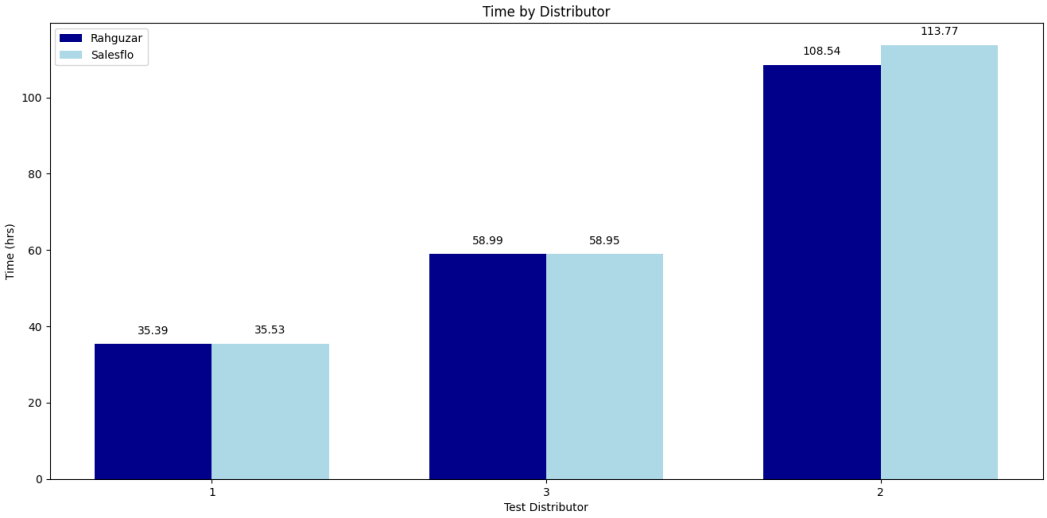

Figure 7.4: Total time comparision for the test distributors

Based on the results of the experiments, the Evolutionary Algorithm (EA) consistently demonstrated strong performance in the scheduling phase, effectively balancing workloads and meeting visit constraints across varying distributor profiles. For route optimization, Google OR-Tools emerged as the most robust and scalable

solution, outperforming other algorithms—including ACO and PSO—particularly in large-scale and constraint-heavy scenarios.

The combination of EA for scheduling and OR-Tools for routing provided the most efficient and adaptable pipeline, delivering optimal results in both distance and time minimization. This pairing proved especially effective in consistently handling real-world distributor datasets.

Furthermore, when benchmarked against Salesflo, Rahguzar showed clear improvements in planning efficiency, with notable reductions in total distance traveled and route time. These findings highlight Rahguzar's potential as a scalable and intelligent planning system that can adapt to operational complexity while delivering measurable gains in field performance.

8. Conclusion and Future Work

The Rahguzar project was conceptualized and developed in response to the growing operational challenges faced in manual PJP planning. Through a modular, heuristic-driven methodology combining clustering, scheduling, and routing optimizations, Rahguzar demonstrated substantial improvements in route efficiency, workload distribution, and scalability. The following sections summarize the project's key outcomes, identify its limitations, and propose directions for future research and system enhancements to further advance intelligent journey planning.

8.1 Conclusion

In this project, we presented Rahguzar, a comprehensive solution for optimizing Permanent Journey Plan (PJP) scheduling in the Fast-Moving Consumer Goods (FMCG) industry. Our system integrates customer clustering, schedule generation via Evolutionary Algorithms, and route optimization using OR-Tools, delivering a robust end-to-end pipeline for sales team planning and management.

Through the integration of real-world datasets and a user-friendly web-based interface, Rahguzar enables dynamic and efficient route planning, helping sales managers minimize travel costs while maximizing coverage and visit compliance. Performance evaluations demonstrate the system's ability to produce realistic, optimized schedules with significantly improved route efficiency compared to manually designed plans.

The modular architecture allows easy adaptation to various geographical scales, constraints, and business-specific requirements. Moreover, our backend pipeline has been successfully deployed and integrated with a React-based frontend, creating a seamless experience for end users.

8.2 Future Work

In future iterations of Rahguzar, several enhancements can be explored to improve its effectiveness and integration with real-world FMCG workflows. One key direction is the incorporation of real-time traffic data and dynamic scheduling capabilities, enabling the system to adapt routes on the fly in response to disruptions such as delays or cancellations. Integration with Salesflo's existing mobile platform—which is already widely used by field agents—can also be explored to enable seamless access to route plans, check-ins, and live updates. This would allow on-ground activities to be synchronized with the backend optimization engine, enhancing both usability and data flow. Another promising direction is the use of historical and real-time sales data to drive scheduling decisions. By aligning visit frequency and prioritization with customer-specific sales performance, seasonal trends, and product movement, Rahguzar can help optimize not only travel efficiency but also commercial outcomes. Together, these enhancements can transform Rahguzar into a fully adaptive, data-driven system for sales force automation.

9. Reflections

The Rahguzar project presented an opportunity not only to apply theoretical concepts in a practical setting but also to experience the real-world challenges of system development, collaboration, and research-driven problem solving. This section provides a comprehensive reflection on individual contributions, team dynamics, and process evolution throughout the project lifecycle. By analyzing the gap between initial plans and actual achievements, we aim to capture the key lessons learned and areas for personal and collective growth.

9.1 Individual Reflections

- **Nabila Zahra**: Working on Rahguzar has been one of the most rewarding experiences of my academic life, helping me grow both technically and personally. It was the first time I contributed to a project of this scale, developed in close collaboration with an industry partner—SalesFlo—where every component, from algorithm design to deployment, had real-world implications. This made the journey both challenging and rewarding.

  One of the most meaningful parts of the project for me was working on the scheduling system using an Evolutionary Algorithm. I had previously studied EAs in my Computational Intelligence course at Habib University, but this project allowed me to apply that theoretical knowledge in a much more complex and realistic setting. Translating business constraints into fitness functions, tuning parameters based on performance, and integrating the scheduler into a broader optimization pipeline helped me understand the practical challenges of implementing heuristic algorithms. It reinforced that academic understanding is only the starting point—real-world systems require adaptability, debugging, and iteration beyond what textbooks teach.

  Beyond the algorithm itself, I worked extensively on both the backend and

frontend of our application. I developed APIs to serve the optimized schedules and routing data, and helped integrate them with our interactive map interface. Seeing the UI reflect our algorithmic output in real time was incredibly satisfying—it reminded me why I enjoy building end-to-end systems.

I also stepped out of my comfort zone by handling deployment tasks. I learned how to use Docker to containerize OSRM, and how to set up and configure AWS EC2 and RDS with PostgreSQL. These were tools I had not worked with before, and figuring them out taught me how to be resourceful and persistent when dealing with unfamiliar systems. At times, debugging deployment issues felt frustrating, but I now see how those moments pushed me to grow the most.

What stood out to me throughout this project was how much value lies in cross-functional thinking—understanding how the backend affects the frontend, how the database interacts with APIs, and how deployment choices affect performance. Working on Rahguzar helped me connect these dots and become more confident in navigating complex, integrated systems.

Most importantly, this project reminded me that challenges are learning curves. Whether it was refining the EA scheduler, troubleshooting server errors, or adapting designs based on feedback, each obstacle became an opportunity to learn, grow, and build something better. I walk away from this project with a deeper appreciation for teamwork, a stronger technical foundation, and a clearer vision of the kind of systems I want to build in the future.

- **Iqra Azfar**: Over the duration of the project, I was not only dedicated to doing extensive research and examining different facets of the issue, but also to developing essential technical and soft skills on an ongoing basis. The methodology included picking up new tools, studying intricate concepts, and implementing them to solve real-world problems. Collaboration with my team members was an enriching experience, where I actively participated in brainstorming meetings, solution creation, and decision-making. Working with others showed me the value of effective communication, flexibility, and respect towards one another, particularly in dealing with conflicts or differing opinions. These experiences helped me learn more about teamwork, problem-solving, and lifelong learning.

- **Muhammad Youshay**: Embarking on this project was an enriching experience, particularly due to the opportunity to solve an actual industry problem presented by SalesFlo in the FMCG sector—a field entirely new to me. This required a deep understanding of the intricacies involved in permanent journey

plan (PJP) scheduling, pushing me to thoroughly analyze operational constraints, optimization criteria, and stakeholder needs. The Human-Centered Design (HCD) approach was central to our methodology, enabling us to systematically empathize with end-users, define core problems, ideate viable solutions, prototype concepts, and iterate based on real-world feedback.

The experience significantly expanded my skill set. Leveraging my academic knowledge from Habib University's courses, particularly Computational Intelligence, was crucial in developing the evolutionary algorithm component of our project. This course provided the theoretical foundation and practical techniques that enabled the design and fine-tuning of our hybrid optimization approach, effectively addressing complex, multi-objective problems. Additionally, my problem-solving abilities and logic-building skills greatly improved as I navigated challenges such as workload balancing, geographic clustering, and routing optimization. Exploring the problem from different perspectives and applying diverse analytical lenses was particularly enlightening.

From a technical standpoint, I enhanced my proficiency in Python, OR-Tools, and data preprocessing techniques. Furthermore, interacting extensively with PostgreSQL and cloud-based systems like AWS strengthened my database management and backend development capabilities, preparing me for more complex technical tasks in the future.

Working in a team presented both enriching experiences and notable challenges. Our team brought together individuals with different ideas, approaches, and technical expertise. Achieving consensus required effective communication, open-mindedness, and patience. Navigating through varied perspectives and integrating everyone's inputs to form cohesive solutions taught me invaluable lessons in teamwork, collaboration, and leadership.

Overall, the experience was immensely rewarding, significantly boosting both my technical expertise and soft skills. I have emerged more confident in my ability to approach complex real-world problems with structured, innovative solutions, and equipped to thrive in collaborative environments.

- **Rabia Shahab**: Working on Rahguzar has been one of the most impactful experiences of my undergraduate journey, offering a chance to apply technical skills to a meaningful real-world problem. Collaborating with SalesFlo introduced a level of complexity and accountability that shaped how we approached both design and implementation. Our work wasn't just theoretical—it had to make sense in a business context and stand up to practical constraints, which

made the learning curve steep but highly rewarding.

My core contributions focused on the areas of developing the Evolutionary Algorithm (EA) scheduler and designing a performance dashboard. The EA scheduler pushed me to move beyond textbook models and think critically about how to encode real-world business requirements—such as fairness in visit distribution and route efficiency—into algorithmic logic. Balancing multiple objectives, tuning parameters, and debugging unexpected behaviors taught me that optimization in real systems is messy, and that iteration is essential.

Simultaneously, building the dashboard gave me a deeper appreciation for data interpretation and user-centered design. It helped me identify and prioritize the KPIs that actually matter—things like distance coverage, visit gaps, and time allocation per resource. Creating a clear interface to display these metrics not only made the system more usable, but also helped bridge the communication gap between the technical team and non-technical stakeholders and enable them to be more data-driven. It was a reminder that even the best algorithm needs to be transparent and interpretable to deliver real value.

Throughout the project, I also explored new technical tools that were initially unfamiliar. I worked with OSRM to integrate routing functionality and used Docker to containerize services for a smoother deployment process. These experiences gave me hands-on exposure to backend infrastructure and helped me understand how different system components—schedulers, route engines, data pipelines—must work together in sync.

Equally important was the experience of working with my teammates. We each brought different strengths to the table, and there were moments where progress felt uncertain—whether due to technical issues or conflicting ideas. However, through open discussion, shared debugging sessions, and brainstorming under pressure, we consistently found ways forward. I learned that collaboration isn't just about dividing tasks—it's about supporting each other, challenging assumptions, and finding common ground when faced with complexity.

Rahguzar taught me that building real systems requires not just technical skills but also adaptability, empathy, and trust in your team. I leave this project with greater confidence in my ability to contribute to large, interdisciplinary efforts, and a stronger sense of the kind of systems and teams I want to be a part of in the future.

9.2 Team Reflection

Working together on Rahguzar was a deeply collaborative experience that taught us the value of shared vision, consistent communication, and mutual accountability. While we were united by a shared goal, the process of getting there involved navigating a variety of challenges that tested our coordination, communication, and adaptability. One of the most significant challenges we encountered was working at different paces. Each team member had their own academic and personal commitments, and synchronizing our work schedules proved difficult at times. There were also disagreements on how to approach certain parts of the problem, particularly in algorithm design, interface behavior, and system architecture. At one point, we struggled to align our approaches to problem-solving, particularly in how to structure the optimization logic, divide backend responsibilities, and sequence integration with the frontend. Each of us had different interpretations of the problem and varying ideas on how best to implement certain features. These disagreements sometimes led to delays and overlapping efforts. However, they also turned out to be important learning opportunities. What helped us move forward was our ability to step back, have honest discussions, and actively listen to one another. The guidance and support we received from our supervisor and mentors during these phases were especially valuable. Their input helped us untangle complex decisions, identify a shared direction, and turn moments of friction into productive design conversations. Despite the differences in working styles and technical opinions, we learned to trust each other's strengths and make space for every perspective. As the project progressed, we grew more coordinated and developed a rhythm that allowed us to work more effectively together. Ultimately,Rahguzar was not just a technical achievement, but a product of collective resilience, trust, and shared commitment to solving a real-world problem together.

9.3 Process Reflection

Looking back at the process, one of the things that worked particularly well was our decision to follow an iterative and research-driven development approach. Early in the project, we focused on understanding the problem deeply through industry interviews, literature review, and small-scale experiments. This helped us avoid jumping prematurely into implementation and instead guided our design with clarity and purpose. Another strength was our consistent engagement with real-world data and constraints. By validating our design choices against actual operational needs

provided by SalesFlo, we ensured that our solution stayed grounded and practical.

However, we also faced challenges that impacted our efficiency. A major bottleneck occurred when all of us were working on different parts of the codebase simultaneously. While this parallel effort helped accelerate feature development, it made integration complex and time-consuming. Merging work from multiple branches while ensuring compatibility and system stability required multiple debugging sessions and technical compromises. In hindsight, a more modular and version-controlled integration plan from the start could have reduced friction.

We also underestimated the time needed for certain tasks, particularly data integration and testing. Delays in data availability affected our ability to validate early outputs, and unexpected inconsistencies in the dataset required additional preprocessing.

Despite these challenges, the process evolved and improved over time. We became more deliberate in dividing tasks, more consistent in communication, and more disciplined in tracking progress. This experience highlighted the importance of not just technical knowledge, but also process design and team coordination in building real-world systems.

9.4 Plan vs Achievment

9.5 Plan vs Achievement

The initial vision for the Rahguzar system was shaped by academic research, stakeholder discussions, and a focus on building a modular, data-driven solution for journey plan optimization. While the core objectives remained consistent throughout the project, many aspects of the system evolved significantly during development. These changes were driven by technical constraints, real-world deployment challenges, and feedback from pilot testing. As a result, several design decisions were revised, new features were added, and some proposed components were removed—ultimately leading to a more practical, efficient, and scalable system.

Algorithm Design: Initially, a basic heuristic-based approach was proposed for assigning stores to days and order bookers. However, this proved insufficient to meet complex requirements such as visit frequency enforcement, workload balancing, and multi-day planning. We pivoted to a custom-built Evolutionary Algorithm (EA), which allowed encoding multiple operational constraints into a single fitness function. This shift enabled more flexible and scalable scheduling under real-world conditions.

Routing Engine: The original system design relied on the Google Maps API for generating travel distances and times. During development, cost limitations and API quotas led us to migrate to a Dockerized, self-hosted instance of the Open Source Routing Machine (OSRM), which offered a scalable, open-source alternative. OSRM integrated seamlessly into our Flask backend and provided accurate routing using OpenStreetMap data, supporting high-performance route sequencing at no additional cost.

Technology Stack: While Power BI was initially considered for dashboard visualization, it lacked the flexibility needed for real-time updates and seamless backend integration. This limitation prompted a shift to a custom React-based dashboard, with Flask serving data endpoints. The result was a more interactive and responsive UI, with tighter control over both the visual and functional layers of the system.

Data Management: Contrary to initial assumptions, store data and geographic metadata required substantial preprocessing. Duplicates, missing coordinates, and inconsistent fields had to be addressed through custom scripts and validation steps. These preprocessing efforts became crucial to maintaining accuracy in clustering, scheduling, and route generation.

Integration Strategy: Our original plan was to integrate all components toward the final phase of development. This approach proved problematic, especially as different modules evolved in parallel. As a result, we adopted a continuous integration model midway through the project, merging backend and frontend functionality incrementally to detect compatibility issues earlier.

Performance Metrics: Initially, success was to be measured in terms of algorithm speed and distance minimization. However, real-world feedback highlighted the need for more meaningful metrics. We revised our KPIs to include visit frequency adherence, shift time compliance, workload standard deviation, and travel vs in-store time ratios. These metrics offered more actionable insights and improved the system's operational relevance.

Feature Additions:

- **Custom Day Planning:** Instead of fixed weekly or monthly planning cycles, the system now allows users to define custom durations for journey plans, providing greater flexibility.
- **Automatic Extra Day Allocation:** If the number of available days is insufficient to accommodate all required store visits, the system intelligently adds extra working days to complete the plan while maintaining constraint compliance.

- **Dynamic Rerouting and Manual Overrides:** Managers can trigger rerouting in real time by modifying store assignments, improving adaptability.

Feature Removals:

- **Sales Data Integration:** Early plans included integration with historical sales data to prioritize high-performing stores. This feature was removed due to unavailability of sales records and limited scope in the pilot phase.
- **Mobile Application Support:** Native mobile deployment was removed from the scope in favor of a responsive web-based interface, reducing complexity while retaining cross-device accessibility.
- **OB Recommendation System:** A proposed OB-store matching suggestion engine was deprioritized due to time constraints and shifting focus toward core scheduling logic.

In summary, while several originally proposed features were re-evaluated or removed, new capabilities were introduced that better addressed stakeholder needs and system scalability. These adaptive decisions reflect the project's iterative development philosophy and contributed to a more robust, usable, and deployment-ready solution.

A. Code

Our code can be found at Rahguzar GitHub Repository .

References

[1] Mohammad Khurshed Alam and Mohd Herwan Sulaiman. "Optimal loading analysis with penalty factors for generators using brute force method". In: *Lecture Notes in Electrical Engineering* (2022), pp. 37–46. DOI: 10.1007/978-981-16-8690-0_4.

[2] Alexis. *How route optimization can save time and money*. Dec. 2023. URL: http://gofleet.com/how-route-optimization-can-save-time-and-money/.

[3] Dalia Essa Almaatani. "New Computational Approaches for the Transportation Models". Accessed: 2024-12-06. MA thesis. Laurentian University of Sudbury, Sept. 2014. URL: https://laurentian.scholaris.ca/items/3c4e7436-4b97-4664-9289-957f8690fddf.

[4] Karina Corona-Gutiérrez, Samuel Nucamendi-Guillén, and Eduardo Lalla-Ruiz. "Vehicle routing with cumulative objectives: A state of the art and analysis". In: *Computers & Industrial Engineering* 169 (2022), p. 108054. ISSN: 0360-8352. DOI: 10.1016/j.cie.2022.108054. URL: https://www.sciencedirect.com/science/article/pii/S0360835222001243.

[5] Xin Cui. "Logistics cost control research of FMCG industry". In: *Advanced Materials Research* 468-471 (Feb. 2012), pp. 1866–1869. DOI: 10.4028/www.scientific.net/amr.468-471.1866.

[6] Guy Desaulniers, Jacques Desrosiers, and Marius Solomon. "Accelerating strategies for column generation methods". In: *Transportation Science* 36.2 (2002), pp. 213–226. DOI: 10.1287/trsc.36.2.213.5720.

[7] Dominique Feillet et al. "An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems". In: *Networks* 44.3 (2004), pp. 216–229. DOI: 10.1002/net.20034.

[8] Shangce Gao et al. “Ant colony optimization with clustering for solving the dynamic location routing problem”. In: *Applied Mathematics and Computation* 285 (July 2016). Available online 16 April 2016, pp. 149–173. DOI: 10.1016/j.amc.2016.04.021. URL: https://doi.org/10.1016/j.amc.2016.04.021.

[9] Abel Garcia-Najera and John A. Bullinaria. “An improved multi-objective evolutionary algorithm for the vehicle routing problem with time windows”. In: *Computers & Operations Research* 38.1 (Jan. 2011), pp. 287–300. DOI: 10.1016/j.cor.2010.05.004.

[10] Keivan Ghoseiri and Seyed Farid Ghannadpour. “Multi-objective vehicle routing problem with time windows using goal programming and genetic algorithm”. In: *Applied Soft Computing* 10.4 (Sept. 2010), pp. 1096–1107. DOI: 10.1016/j.asoc.2010.04.001.

[11] Martin Hughes, Marc Goerigk, and Trivikram Dokka. “Particle swarm metaheuristics for robust optimisation with implementation uncertainty”. In: *Computers & Operations Research* 122 (Oct. 2020), p. 104998. DOI: 10.1016/j.cor.2020.104998.

[12] Umair Javed. *The Undocumented Wholesale-Retail Sector*. Insights for Change. Accessed: 2024-12-06. Consortium for Development Policy Research, Oct. 2023. URL: https://cdpr.org.pk/wp-content/uploads/2023/10/The-Undocumented-Wholesale-Retail-Sector.pdf.

[13] Philip Kilby, Patrick Prosser, and Paul Shaw. “Dynamic VRP using local search heuristics”. In: *Operational Research Society* 53.2 (2002), pp. 235–244. DOI: 10.1057/palgrave.jors.2601298.

[14] A. Kok et al. “Dynamic programming algorithm for the vehicle routing problem with time windows and EC social legislation”. In: *Journal of Chemical Physics - J CHEM PHYS* (Jan. 2009).

[15] Abdullah Konak, David W. Coit, and Alice E. Smith. “Multi-objective optimization using Genetic Algorithms: A tutorial”. In: *Reliability Engineering & System Safety* 91.9 (Sept. 2006), pp. 992–1007. DOI: 10.1016/j.ress.2005.11.018.

[16] Gilbert Laporte. “Vehicle Routing Problems with Time Windows”. In: *Transportation Science* 39.1 (2017), pp. 119–129. DOI: 10.1287/trsc.1050.0127.

[17] Raman Maini and Rajeev Goel. “Vehicle routing problem and its solution methodologies: A survey”. In: *International Journal of Logistics Systems and Management* 28.4 (2017), pp. 419–432. DOI: 10.1504/ijlsm.2017.10008188.

[18] Gábor Nagy and Said Salhi. "Heuristic Algorithms for Single and Multiple Depot Vehicle Routing Problems with Pickups and Deliveries". In: *European Journal of Operational Research* 177.2 (2007), pp. 1620–1641. DOI: 10.1016/j.ejor.2005.10.030.

[19] Beatrice Ombuki, Brian J. Ross, and Franklin Hanshar. "Multi-Objective Genetic Algorithms for Vehicle Routing Problem with Time Windows". In: *Applied Intelligence* 24.1 (Feb. 2006), pp. 17–30. DOI: 10.1007/s10489-006-6926-z. URL: https://link.springer.com/article/10.1007/s10489-006-6926-z.

[20] Wan Othman et al. "Solving Vehicle Routing Problem using Ant Colony Optimisation (ACO) Algorithm". In: *International Journal of Research and Engineering* 5 (Nov. 2018), pp. 500–507. DOI: 10.21276/ijre.2018.5.9.2.

[21] Sandhya and Rajeev Goel. "Multi objective vehicle routing problem: A survey". In: *Asian Journal of Computer Science and Technology* 7.3 (Nov. 2018), pp. 1–6. DOI: 10.51983/ajcst-2018.7.3.1903.

[22] Matthias Schneider, Alexander Stenger, and Dominik Goeke. "The Electric Vehicle-Routing Problem with Time Windows and Recharging Stations". In: *Transportation Science* 48.4 (2014), pp. 500–520. DOI: 10.1287/trsc.2013.0490.

[23] Gerard Sierksma and Gerard A. Tijssen. "Sales Territory Alignment Using Mathematical Programming". In: *European Journal of Operational Research* 105.3 (1998), pp. 567–581. DOI: 10.1016/S0377-2217(96)00497-0.

[24] Andrés Villalba and Elsa Cristina Gonzalez la Rotta. "Clustering and heuristics algorithm for the vehicle routing problem with time windows". In: *International Journal of Industrial Engineering Computations* 13 (2022), pp. 165–184. DOI: 10.5267/j.ijiec.2021.12.002.

[25] Feidiao Yang et al. "Boosting Dynamic Programming with Neural Networks for Solving NP-hard Problems". In: PMLR, 2018, pp. 726–739.

[26] Huizhen Zhang et al. "A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows". In: *Information Sciences* 490 (July 2019), pp. 166–190. DOI: 10.1016/j.ins.2019.03.070.

[27] Emir Zunic et al. "A Cluster-Based Approach to Solve Rich Vehicle Routing Problems". In: Mar. 2021, pp. 103–123. ISBN: 978-3-030-71845-9. DOI: 10.1007/978-3-030-71846-6_6.

[28] Emir Zunic et al. “Cluster-based approach for successful solving real-world vehicle routing problems”. In: *Volume 21* (2020), pp. 619–626. DOI: 10.15439/2020F184.