Rahguzar

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	Dr. Syeda Saleha Raza Faculty of Computer Science Habib University

Approved by the Faculty of Computer Science on _____.

Dedication

For ammi, abbu, and pappu.

Acknowledgements

We want to thank the CS faculty and ...

Abstract

Abstract goes here

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. It provides flexibility for real-time updates, ensuring adaptability to dynamic business needs. Additionally, Rahguzar determines the optimal number of field representatives required, improving workforce utilization and preventing inefficiencies such as overstaffing or under-hiring.

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

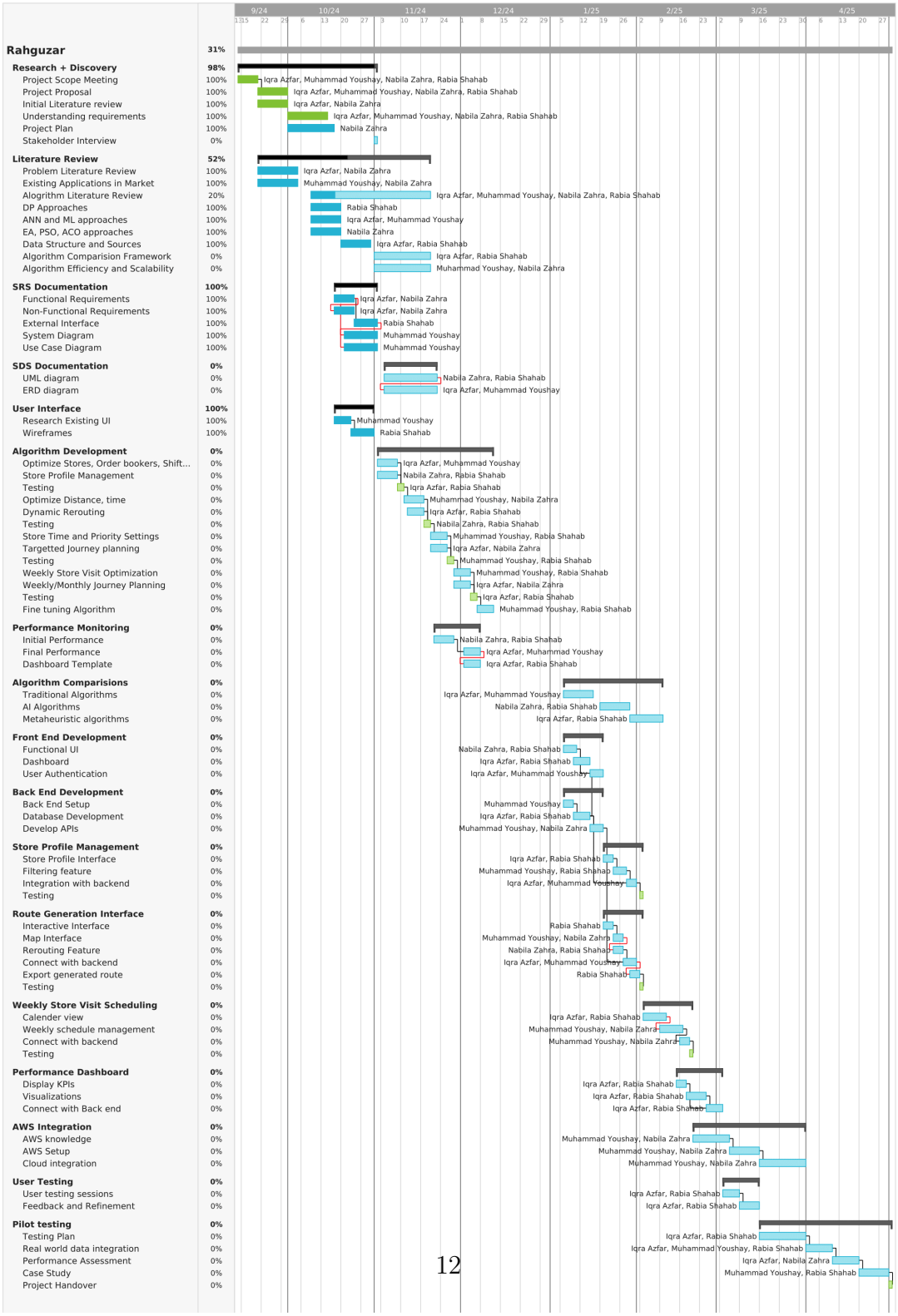

Figure 1.1: Project Gantt Chart

1.5 Key Challenges

Key challenges anticipated in our project will include:

- **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 chapter presents the current state of the art in the domain and talks about other similar work that has been done in this area. It also establishes the novelty of our work by highlighting the differences between the existing work and our work.

We will keep updating this chapter (especially if our project is research-intensive) as our research proceeds and we come across more work related to our problem.

Of course, we take inspiration from [1] but wish the work was typeset in LaTeX[3], e.g. by taking help from [2].

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.

- Include a feature to assign priority to specific parameters to adapt the optimization to business needs.

Function 2: Dynamic Rerouting

- It allows users to interactively adjust key parameters of the route optimization algorithm, such as the number of order bookers and the number of stores per booker, directly on the display screen.
- 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, or three times a week).
- Ensure that the routes align with required visit frequencies without any overlap or gaps.

Function 4: Journey Planning for Different Timeframes:

- Support the generation of routes by allowing monthly PJP creation with customizable weekly patterns (like odd/even weeks) and enable downloads for extended periods, such as a year. 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.

Function 6: Targeted Journey Plans:

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

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 create or adjust routes manually, even if automated suggestions exist.
- 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 2: North Star Metric Tracking:

- Focus on monitoring the primary metric: "Increase in average daily sales per sales representative."

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 2000 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 AI-driven route 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 like Google Maps.
    - 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 following wireframes have been created for our system to give an overview of the features it will provide and what an overview of the interface will look like.

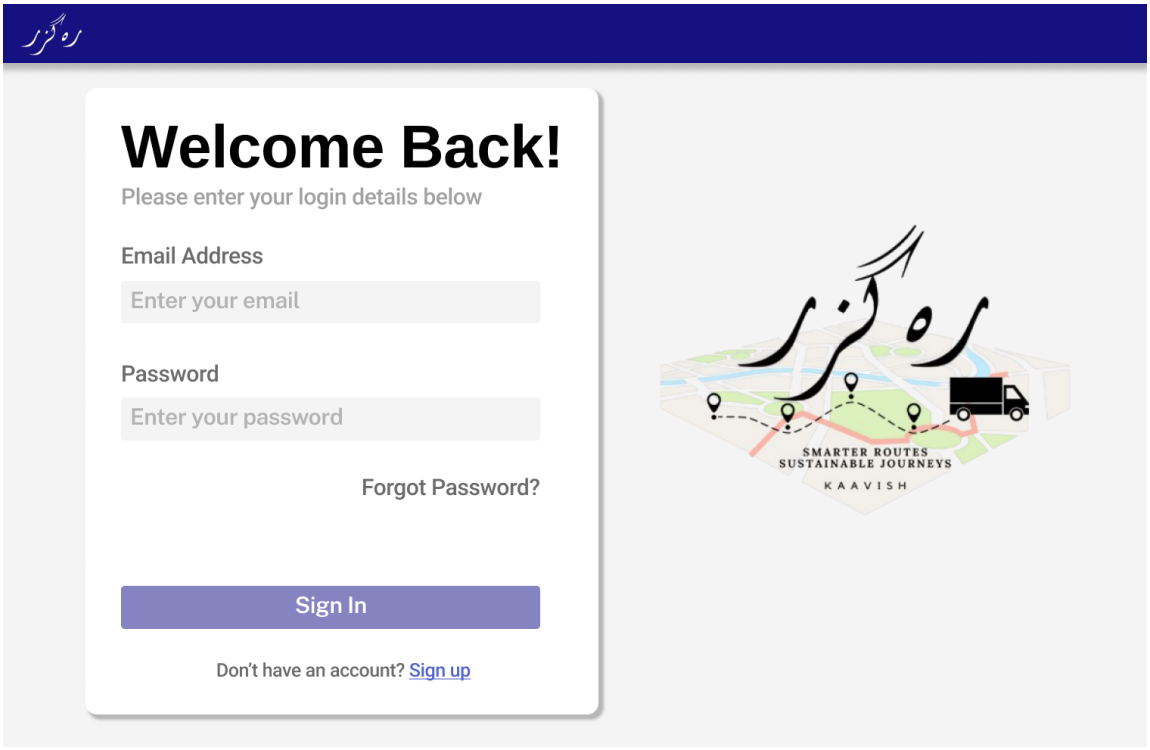

Figure 3.1: Sign In Screen

The sign-in screen is shown in Figure 3.1, letting the user enter their email and password. They also have the option to access the sign up screen using the hyperlink in the image called "Sign up".

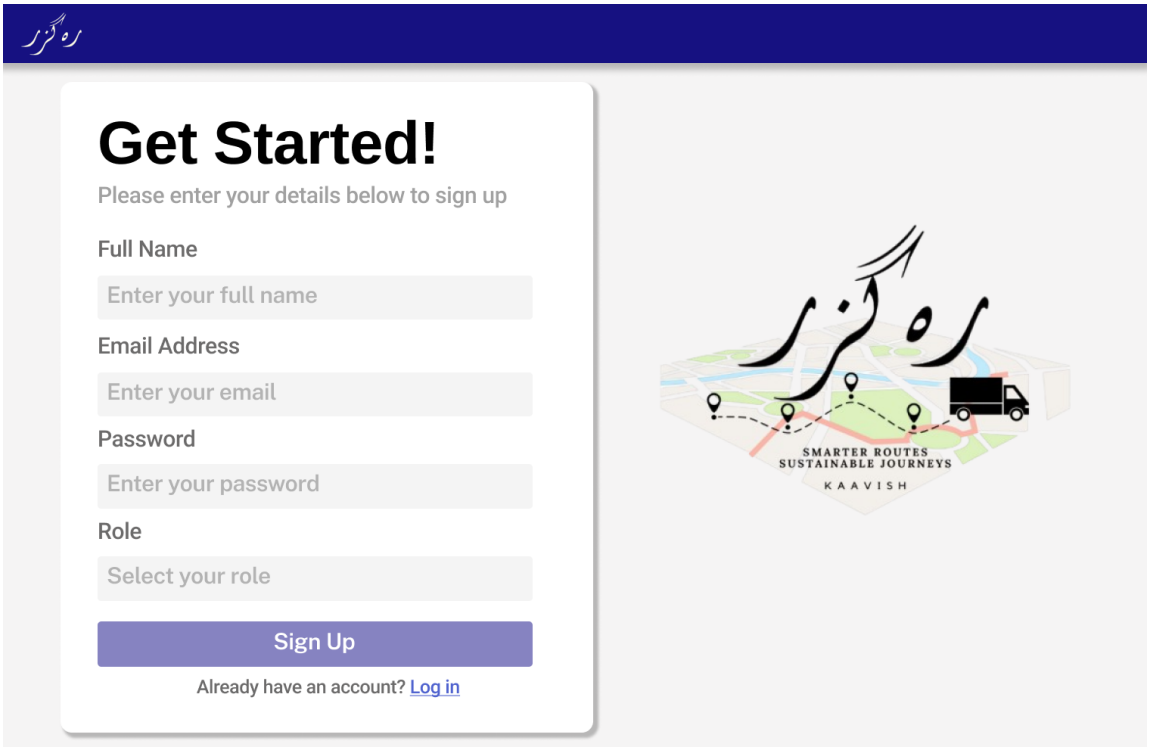

Figure 3.2: Sign Up Screen

The sign-up screen is shown in Figure 3.2, letting the user enter their full name, email and password to create an account.

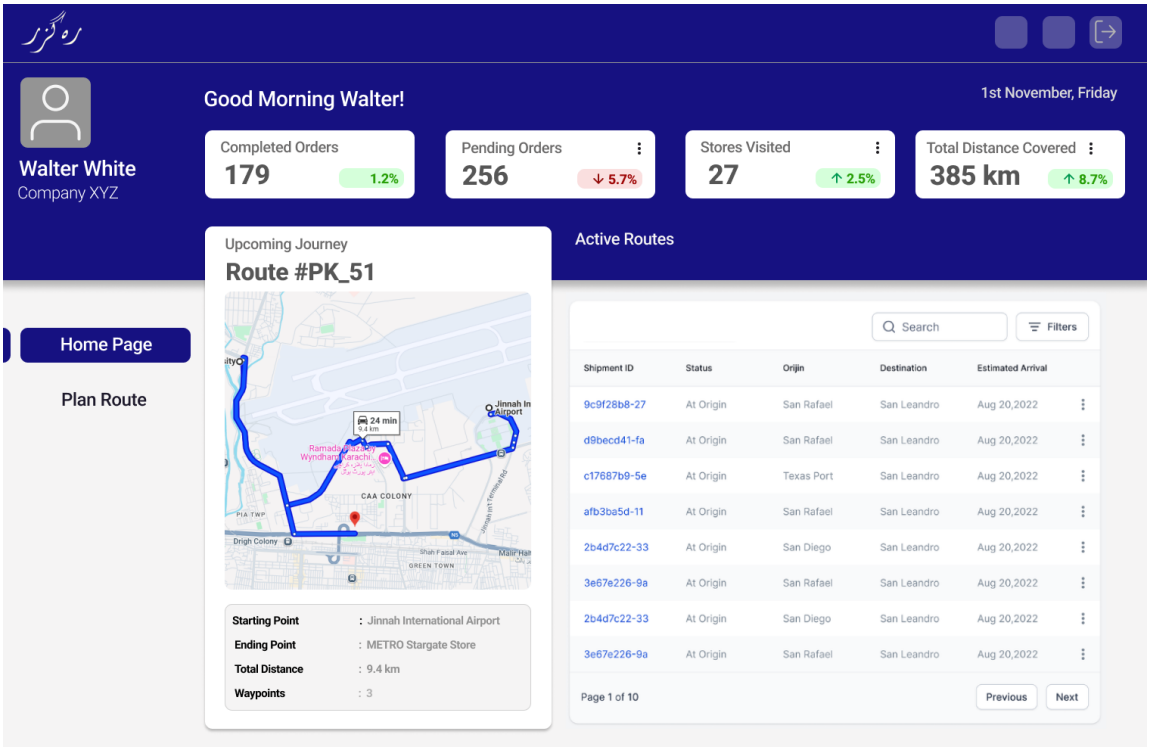

Figure 3.3: Home Page

The "Home Page" screen is shown in Figure 3.3, letting the user see the statistics of the day such as pending and completed orders. The user can view the upcoming recent journey and its related information such as starting and ending points, the total distance of the journey. The user will also be seeing the list of active routes and journeys that are/ will be taking place throughout the day.

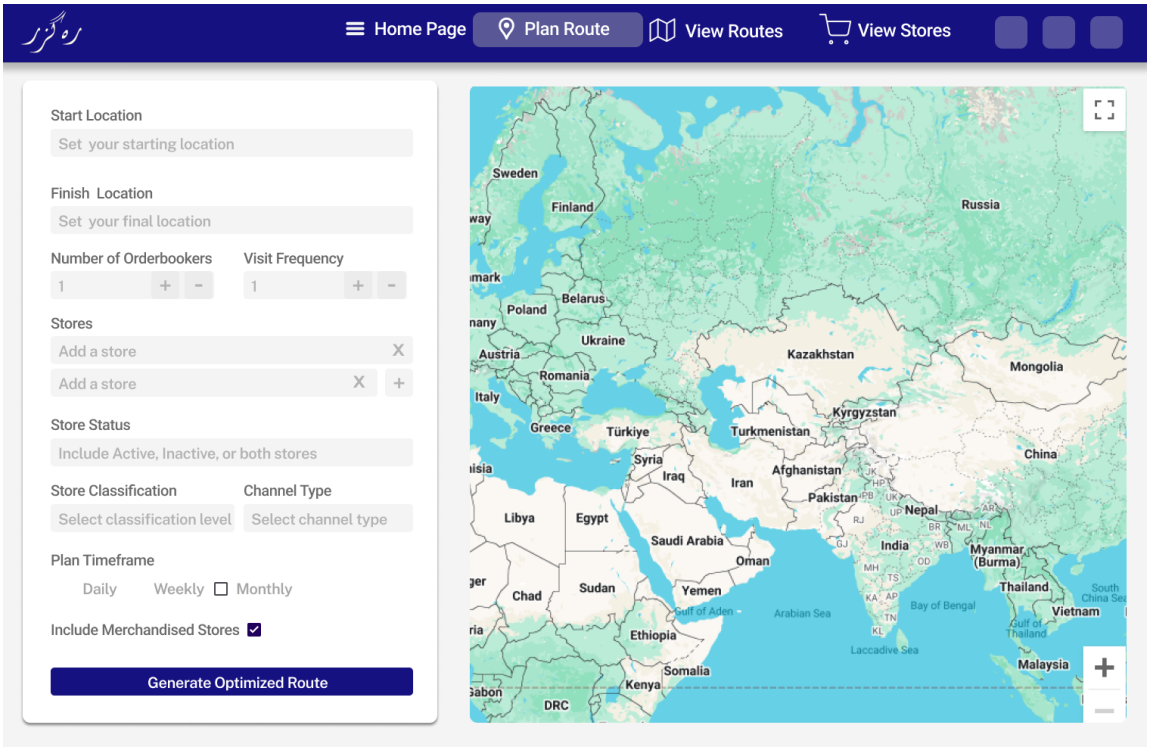

Figure 3.4: Plan Route Page

The “Plan Route” screen is shown in Figure 3.4. In this screen, the manager will be able to create routes by setting the starting point, adding the stores to stop at, the number of orderbookres that will be involved in the journey, apply filters to stores by their classification code and choosing if they want to include only merchandised stores in the plan. Moreover, the manager can make these routes for a single day, week, or a month.

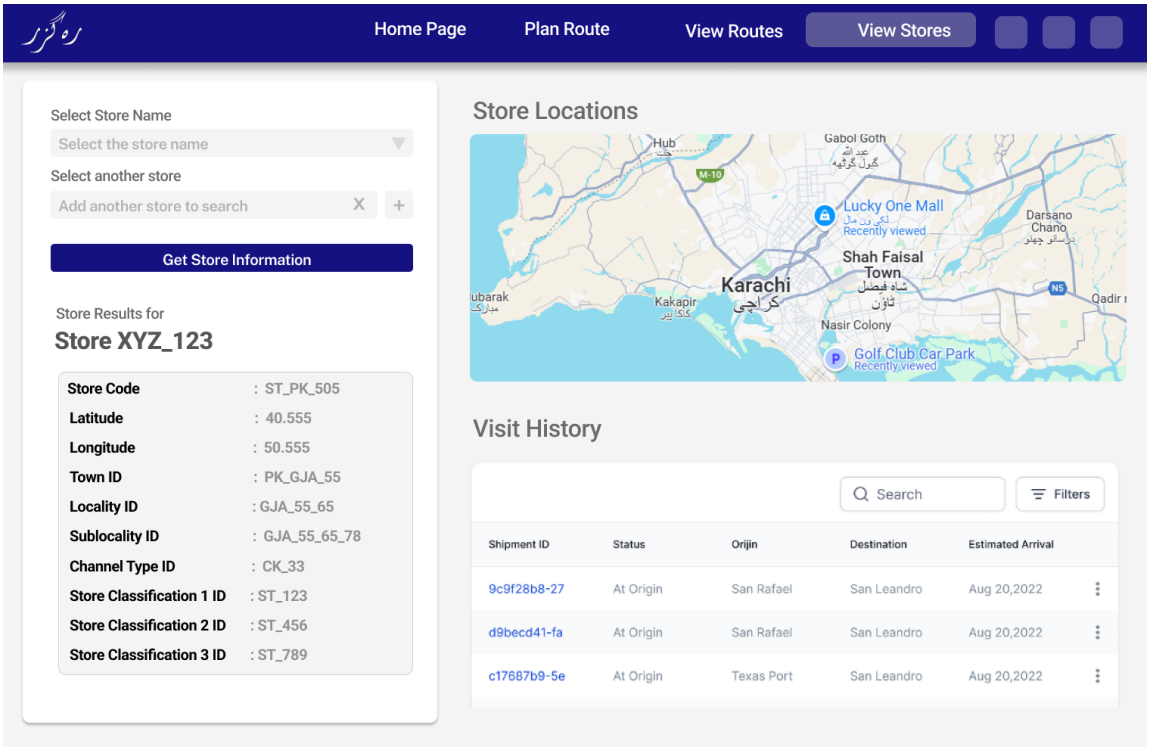

Figure 3.5: View Stores Page

The "View Stores" screen is shown in Figure 3.5. In this screen, the manager will be able to search a certain store and view its information such as its coordinates, its geographical hierarchy. They will be able to see the stores location on the map along with past orderbooker visits data. The manager has the option to view more than 1 store's information at a time.

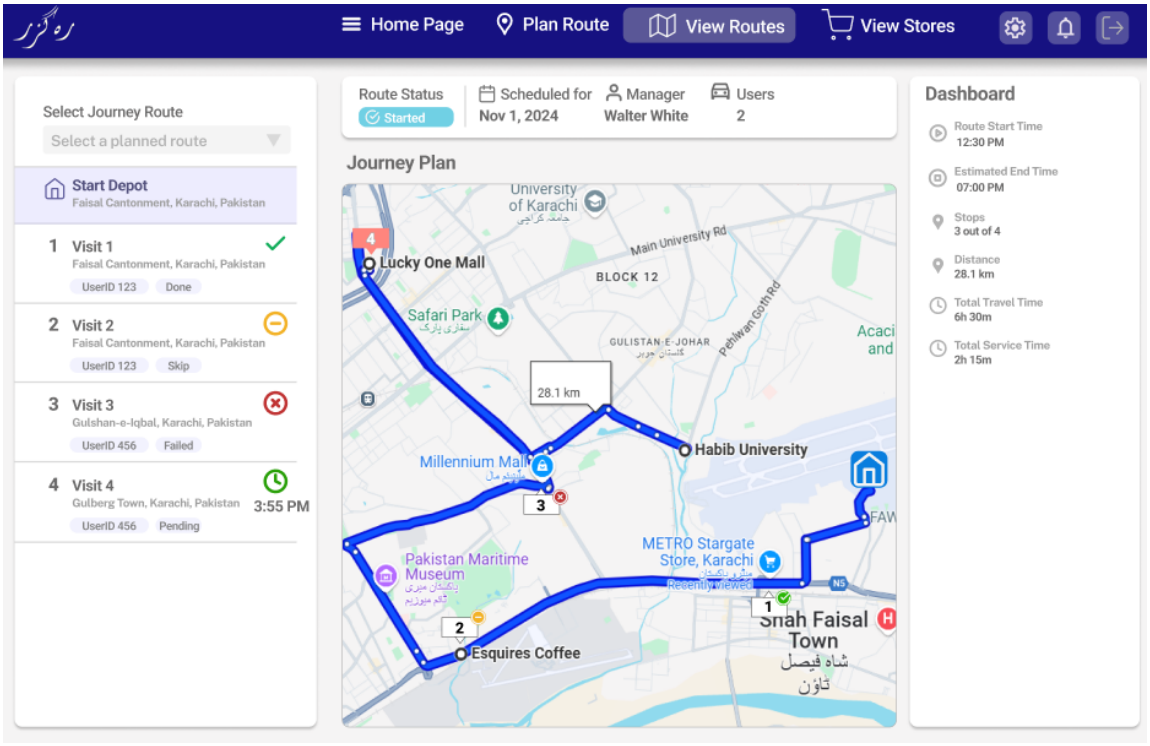

Figure 3.6: View Routes Page

The "View Routes" screen is shown in Figure 3.6. In this screen, the manager will be able to select a certain route and view its information such as the sets of locations it is comprised of, the status of the route. If a certain route has started the manager can see the status of each store location (whether it has been visited or not, the estimated time of visiting). Moreover, the manager can access the dashboard that displays route information such as the starting adn estimated end time, the total distance of the route, the distance covered by it and the number of stops involved.

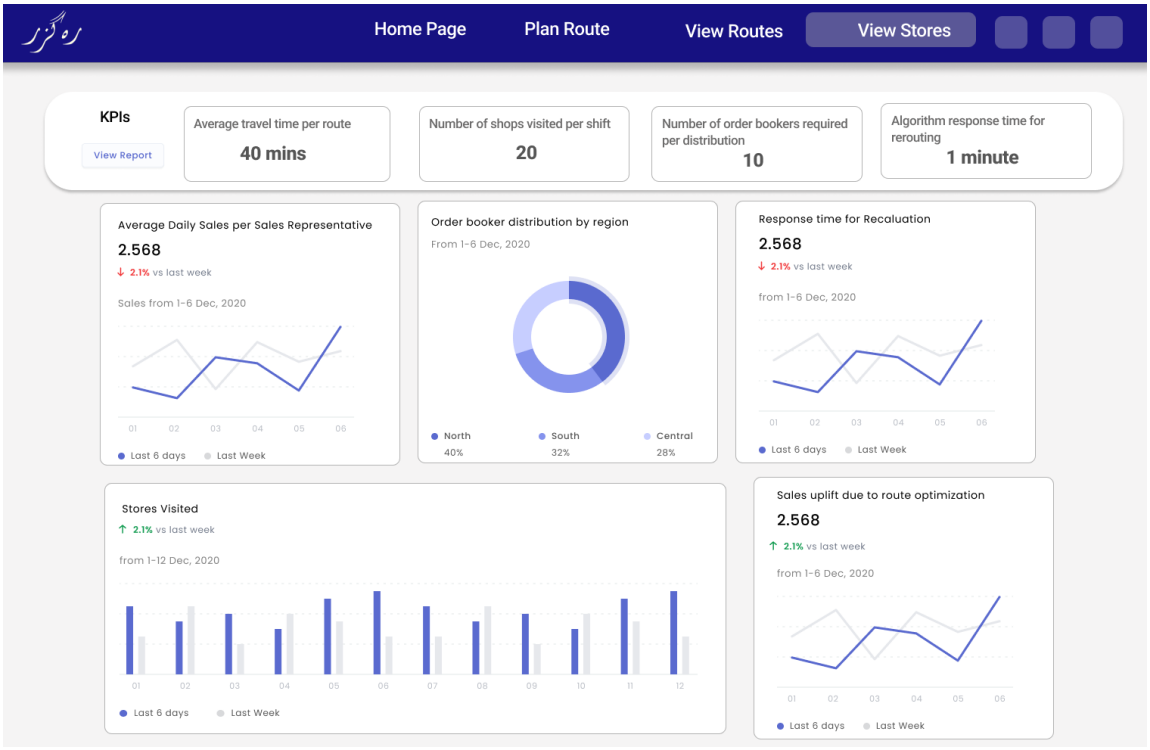

Figure 3.7: Dashboard Page

The "Dashboard" screen is shown in Figure 3.7. In this screen, the manager will be able to view certain KPIs that indicate the overall performance of the routes such as Average Daily Sales per Sales Representative, Average travel time per route, and etc.

3.3.2 Application Program Interface (API)

The API layer is the backbone of our system, enabling seamless communication between the frontend, backend, and external services. Designed to support efficient data exchange and operational scalability, this API structure ensures the system remains responsive and adaptable to business needs.

REST API Endpoints

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

- **Route Management Endpoints:** These endpoints facilitate route creation, configuration, and optimization. Managers can define routes, set parameters such as shift times, service times, and visit frequency, and retrieve optimized route plans tailored to specific business needs.
- **Visit Management Endpoints:** Used to track and manage visits by field representatives. These endpoints enable updates on visit status, start and end times, store visit durations, and compliance with predefined visit schedules.
- **Store Profile Endpoints:** This set of endpoints allows access to and modification of store data. Store profiles include geographic location, classification, operational hours, and specific requirements, supporting managers in executing their tasks efficiently.

Authentication and Authorization

To secure API access, the Rahguzar system employs JWT (JSON Web Tokens) for user authentication, ensuring that only authorized users can access or modify data.

- **JWT Authentication:** When a user logs in, a JWT is issued and must be included with all subsequent requests. This token-based approach enhances security by verifying user identity on each API call.

External Services Integration

To support essential functionalities like mapping, the Rahguzar system integrates with external services:

- **Google Maps API:** This API provides map visualization and geolocation services, enabling managers and field staff to view real-time route details. It is also used for calculating distances, travel times, and generating optimized paths between stops on a route.

Route Optimization Engine

The Route Optimization Engine is a core component of Rahguzar, leveraging advanced algorithms to generate efficient routes.

- **Python + OR-Tools Integration:** Built on Python with OR-Tools, this engine generates optimized routes by balancing factors such as total travel time, number of stops, and time spent at each store. This feature is essential

for ensuring that routes are aligned with operational goals while maximizing efficiency.

- **Geospatial Services (Geopy):** Geopy is used for geocoding services and calculating geospatial data, helping to accurately map store locations and account for distances between various stops. This service plays a key role in generating precise, reliable routes for field operations.

Data Synchronization

The system employs MongoDB as the primary data store, providing robust data storage and synchronization capabilities.

- **Database Sync with MongoDB:** All route data, store profiles, and visit records are stored in MongoDB, ensuring persistent storage of essential information. MongoDB's flexibility supports the diverse data needs of the Rahguzar system, from route optimization to visit management.

- **Data Update Mechanisms:** The system includes mechanisms to synchronize data across different modules and handle real-time updates, especially for managers and field staff working offline. This ensures data integrity and consistency, even in dynamic environments. To ensure scalability and high availability, MongoDB and related services are hosted on a cloud platform such as AWS, enabling the system to handle larger datasets and user traffic efficiently as the application scales.

Data Monitoring

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

- **Performance Analytics:** API endpoints allow managers to monitor route efficiency through Key Performance Indicators (KPIs), such as average travel time, visit completion rates, and operational metrics. This feature facilitates continuous improvement of route plans and decision-making based on real data, backed by scalable cloud infrastructure for robust data processing and analytics.

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

Cloud-Based Servers: The backend system, hosted on a cloud platform such as AWS, provides scalability, availability, and fault tolerance. Specifications include:

- **Processor:** Multi-core virtual CPUs (e.g., AWS EC2).
- **Memory:** 4GB or more, scalable as required.
- **Storage:** SSD-based storage for rapid access to route, store, and visit data.
- **Network Configuration:** Configured for secure HTTPS communication to maintain data integrity and privacy.

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.

External Services Communication

Google Maps API: Used for map visualization, geolocation, and distance calculation, requiring internet connectivity for location-based data.

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

- **Plan Generation (Weekly/Monthly)**
  - **Description:** Generate long-term journey plans based on store visit requirements.
  - **Actions:**
    * As a manager, I want the ability to generate weekly or monthly journey plans, providing my team with a consistent schedule for store visits.
    * As the system, I need to automatically generate journey plans for weekly and monthly schedules based on specified visit frequencies.

- **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, visit completion) 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 route 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.

- **Store Visit Recording and Analysis**

- **Description:** Record visit details and enable analysis of visit data for operational insights.
- **Actions:**
  - As a manager, I want each store visit's details recorded, allowing me to analyse time spent and identify areas for improvement.
  - As the system, I need to log each store visit's start time, end time, and duration, providing accurate records for analysis and performance tracking.

- **Pilot Testing and Validation**
  - **Description:** Conduct pilot tests to validate system effectiveness in real-world scenarios.
  - **Actions:**
    - As a manager, I want to test the system in a pilot environment to evaluate its performance and impact before full deployment.
    - As the system, I need to support pilot testing by recording metrics during live tests, allowing managers to assess effectiveness and identify areas for improvement.

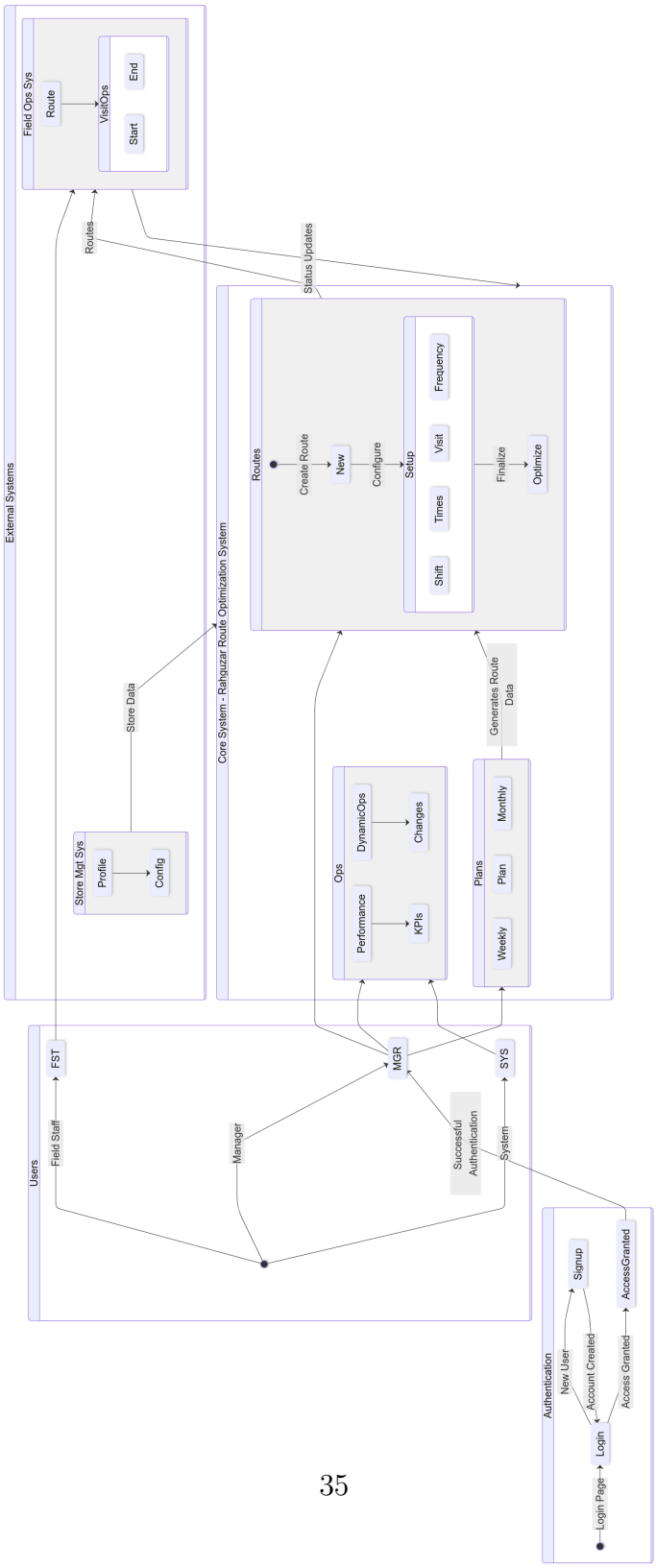

Figure 3.8: Rahguzar's High-Level Use Case Diagram for the Entire System
(Click here to view a high-quality version of this image)

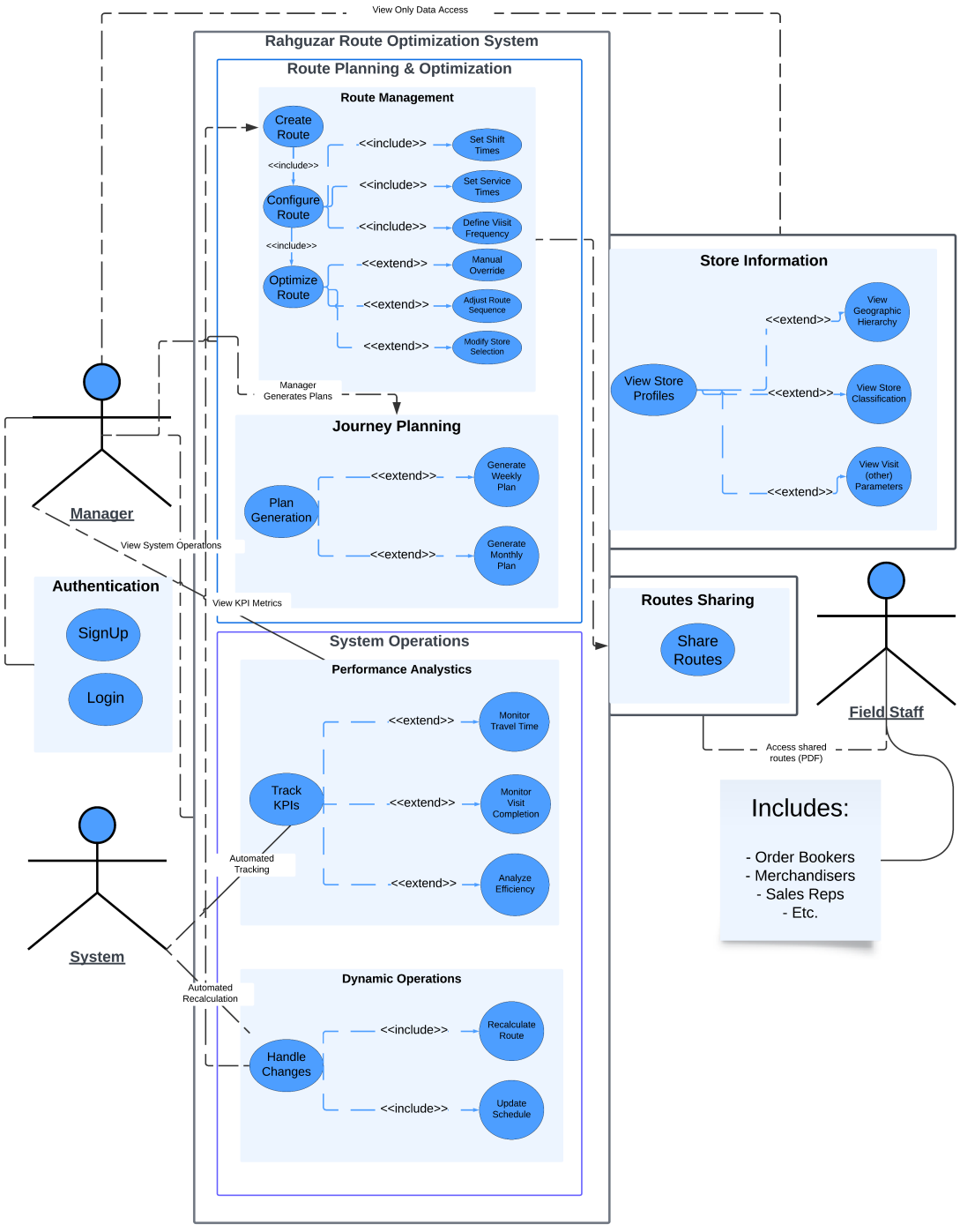

Figure 3.9: Rahguzar's Core System Use Case Diagram within the Main System

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. Further data and details are awaited from SalesFlo, and this section reflects the information we currently have.

Overview of Required 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.

Additional Datasets Needed

To enhance Rahguzar's functionality, the following additional datasets may be required:

1. **Order Booker Shifts Dataset**
   This dataset contains information on the availability and shifts of order bookers. Key fields include:

   - **bookerid:** Unique identifier for each order booker.
   - **shiftstarttime and shiftendtime:** Start and end times of each shift.
   - **availabilitystatus:** Indicates whether the booker is available or unavailable.

This dataset is essential for scheduling routes within the working hours of each order booker, optimizing route generation according to workforce availability.

2. **Geographic Data**
   Geographic and road network data provide detailed map and routing information, potentially sourced via the Google Maps API or GIS data services. Key components include:

   - **roadnetwork:** Map of roads, including distances and travel times.
   - **trafficpatterns:** Historical traffic data for congestion analysis.
   - **geofencingdata:** Predefined boundaries for specific zones, regions, or territories.

   This dataset enables precise route calculations, factoring in real-time or historical traffic conditions to minimize travel time.

3. **Store Activity Requirements**
   This dataset details the specific activities and time requirements for each store visit, allowing Rahguzar to optimize for different tasks performed at each store. Key fields include:

   - **storecode:** Identifier for each store.
   - **activitytype:** Type of activity required (e.g., stock check, merchandising).
   - **estimatedtime:** Average time required to complete each activity.

   This dataset supports the system's ability to tailor visit durations based on store requirements, leading to more accurate time estimates for each route.

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.
- **Dynamic Scheduling:** Using order booker shifts and activity requirements to generate flexible schedules aligned with store needs and workforce availability.
- **Traffic and Geographic Optimization:** Utilizing geographic data to plan routes that avoid congestion and optimize for specific zones or territories.

- **Geographical Filtering:** Geographical and channel-based categorization from the Store Hierarchy dataset will help managers plan routes tailored to specific regions and sales priorities.
- **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.

Further details and data will be incorporated as they become available from SalesFlo.

3.6 System Diagram

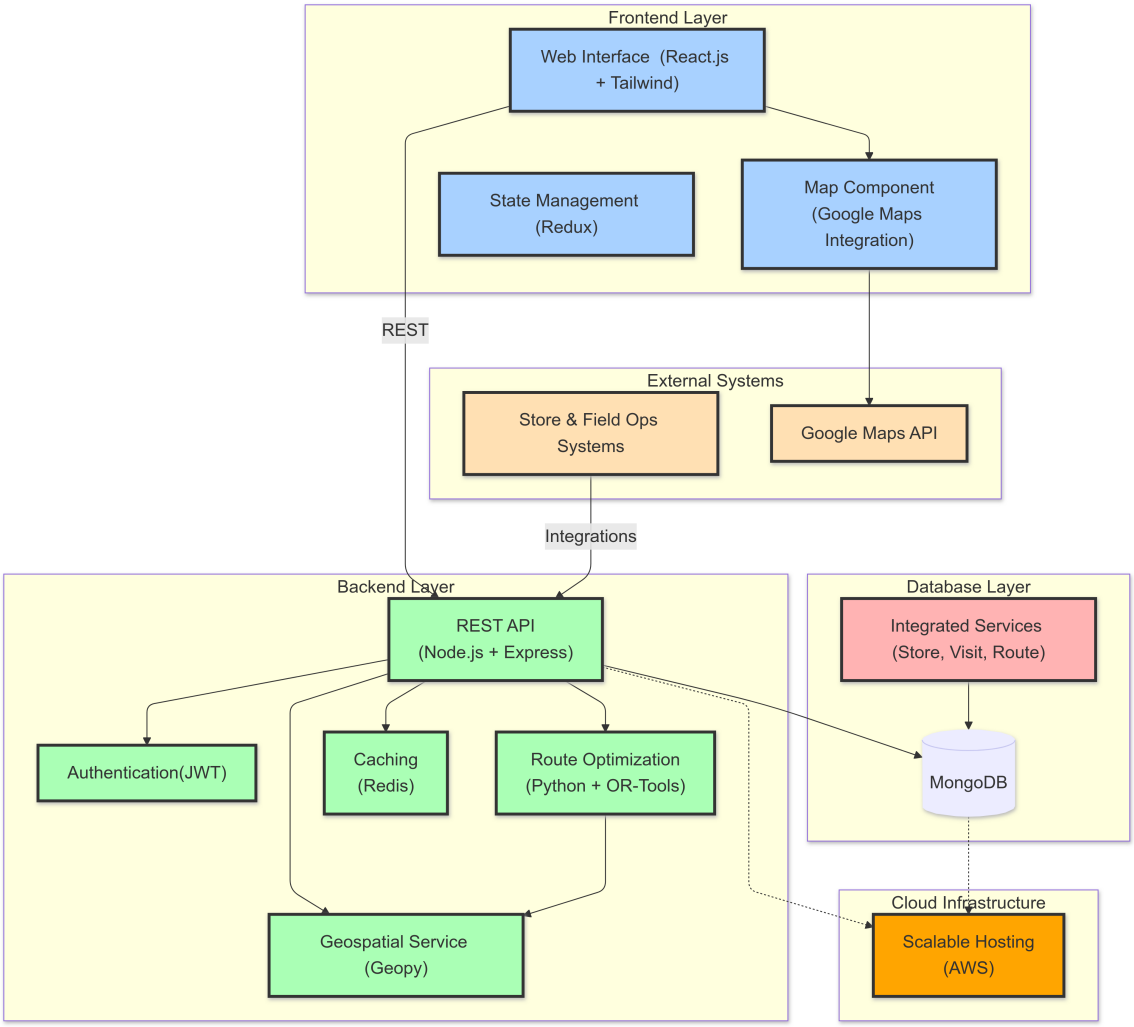

Figure 3.10: Rahguzar's System Architecture Diagram

4. Software Design Specification (SDS)

This chapter provides important artifacts related to design of our project. It includes the Software and Data design of our project.

4.1 Software Design

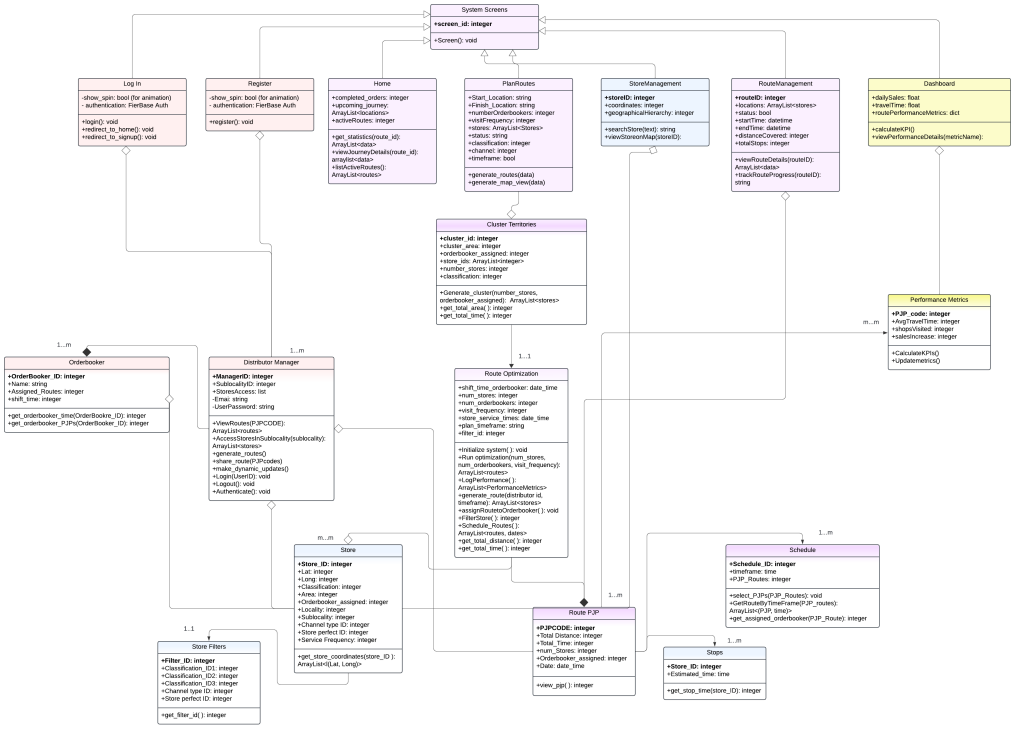

Figure 4.1: UML Class Diagram

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.

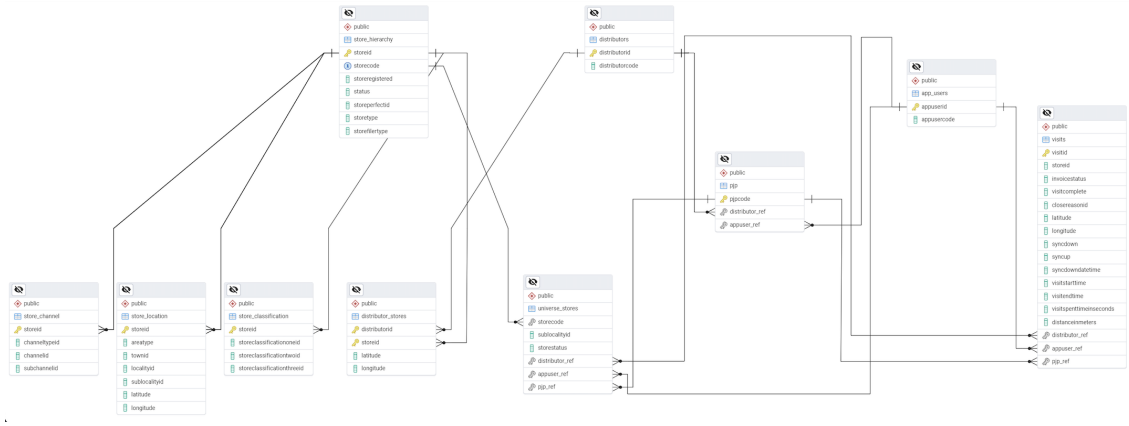

Figure 4.2: ERD Diagram

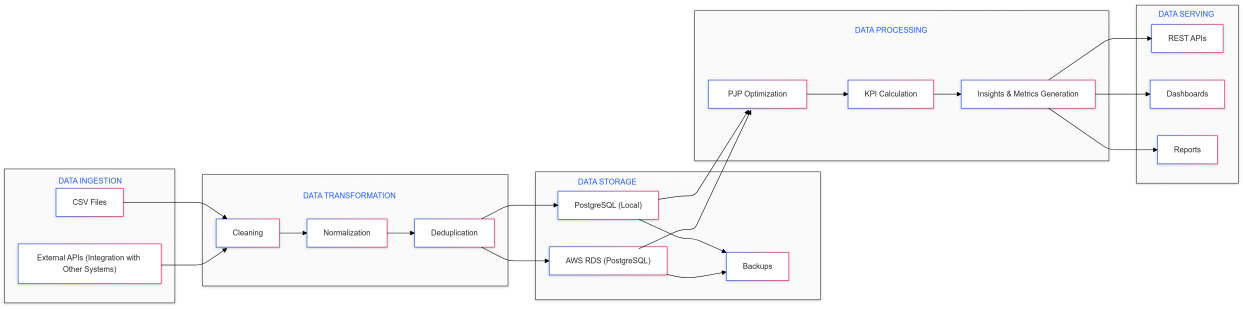

Figure 4.3: End-to-end data pipeline showcasing ingestion, transformation, storage, processing, and serving

4.3 Technical Details

As shown above, our project includes an ERD created based on the cleaned data. Additionally, following are the technical aspects of the algorithm (Evolutionary Algorithm) utilized so far.
The primary objectives of our algorithm are:

- Maximizing the number of shops visited within the constraints.
- Minimizing the total distance traveled to improve route efficiency.
- Minimizing overall time spent, which includes travel and time spent at each store.

The constraints of our algorithm are:

- **Number of order bookers**: Determines the workload distribution and affects route planning.
- **Shift times**: Incorporates break times, holidays, and working hours.
- **Duration at each store**: This is a fixed value for each visit and must be accounted for in total route time.
- **Store priority**: A priority metric for stores, which could be calculated based on sales data.

These objectives and constraints guide the design and functionality of our route optimization algorithm. The algorithm dynamically adapts to these parameters to give optimal route plans.

5. Experiments and Results

We did many experiments and got the best results.

6. Conclusion and Future Work

Our work is awesome. We would write more but we need to catch the flight to collect our Turing Award.

Appendix A. More Math

Here, we describe the background math for the techniques used in the text.

Appendix B. Data

Here is a dump of our 2TB data set. Enjoy!

Appendix C. Code

Here is our code.

```
print('Hello World!')
print('Computing true random number.')
print('Capturing interstellar radiation.')
print('This will take time!')
import random
import time
time.sleep(3600*random.randint(1,10))
print(4)
```

Our code can be found at `https://github.com/habib-university/Kaavish-Template`.

References

[1] Albert Einstein. "Zur Elektrodynamik bewegter Körper. (German) [On the electrodynamics of moving bodies]". In: *Annalen der Physik* 322.10 (1905), pp. 891–921. DOI: http://dx.doi.org/10.1002/andp.19053221004.

[2] Michel Goossens, Frank Mittelbach, and Alexander Samarin. *The LaTeX Companion*. Reading, Massachusetts: Addison-Wesley, 1993.

[3] Donald Knuth. *Knuth: Computers and Typesetting*. 1984. URL: http://www-cs-faculty.stanford.edu/~uno/abcde.html.