FYP – 2 Final Report

Route Optima

Reinventing Delivery

Group Members:

Hammad Habib i200864

Manahil Faisal i200683

Muhammad Abdullah Cheema i200468

Supervisor – Dr Arshad Islam

Fast School of Computing

National University of Computing and Emerging Sciences

Islamabad, Pakistan

2023

Anti-Plagiarism Declaration

This is to declare that the above FYP report produced under the:

Title: Route Optima

is the sole contribution of the author(s) and no part hereof has been reproduced on as its basis (cut and paste) which can be considered as Plagiarism. All referenced parts have been used to argue the idea and have been cited properly. I/We will be responsible and liable for any consequence if violation of this declaration is determined.

Date: July 6, 2024

Student 1 Name:Hammad Habib Signature: Hammad Habib

Student 2 Name:Manahil Faisal Signature: Manahil Faisal

Student 3 Name:Muhammad Abdullah Cheema Signature: Abdullah

Supervisor Name:Mr Arshad Islam Signature: _____

Authors' Declaration

This states the Authors' declaration that the work presented in the report is their own and has not been submitted/presented previously to any other institution or organization.

Abstract

Our project introduces a comprehensive last mile delivery optimization system for courier services in Pakistan, addressing critical challenges. The system incorporates three key components: an advanced algorithm, an intuitive admin portal, and a user-friendly rider app.

The algorithm optimizes parcel assignment by considering time windows, weight constraints, and proximity to delivery locations. The admin portal empowers administrators with rider management, allocation, and an analytics dashboard. Riders utilize the app for trip details, navigation, real-time updates, and performance analytics.

This integrated solution enhances operational efficiency, improves customer satisfaction, and reduces costs. Our project contributes to the enhancement of courier services in Pakistan, making last mile delivery more effective and adaptable to evolving constraints.

Executive Summary

Our project presents a groundbreaking Last Mile Delivery Optimization System tailored to the unique challenges faced by courier services in Pakistan. The system comprises three core elements: a sophisticated algorithm, an intuitive admin portal, and a user-friendly rider app.

The algorithm revolutionizes parcel assignment, factoring in time windows, weight limits, and proximity to delivery locations, resulting in efficient routing to minimize travel time and cost. The admin portal empowers administrators with rider management, allocation, and real-time analytics for data-driven decision-making. Meanwhile, the rider app equips delivery personnel with tools for trip management, navigation, and performance tracking.

By implementing this integrated solution, we aim to significantly enhance operational efficiency, elevate customer satisfaction, and reduce delivery costs. Our project promises to be a pivotal step towards optimizing last mile delivery for courier services, leading to a more efficient and adaptable delivery ecosystem in Pakistan.

Table Of Contents

Introduction

In the intricate world of courier services, the world grapples with a unique challenge: the last mile delivery problem. It's a complex puzzle, where parcels are assigned haphazardly, time windows are often missed, capacity constraints ignored, and routing inefficiencies persist. This not only increases delivery costs but also leads to delays, frustrating customers, riders, and managers.

Our project introduces a transformative Last Mile Delivery Optimization System, comprising three core components: a sophisticated algorithm, an intuitive admin portal, and a user-friendly rider app. At its core, the algorithm reimagines parcel assignments by considering time windows, weight constraints, and delivery proximity. It ensures that parcels are distributed efficiently, while also optimizing routes for the shortest, most cost-effective journey.

The admin portal empowers administrators to manage riders, allocate parcels, and access real-time analytics for informed decision-making. It provides insights into performance metrics, on-time deliveries, and estimated fuel costs. Meanwhile, the rider app equips delivery personnel with tools for efficient trip management, navigation, and real-time performance tracking. This integrated solution promises to enhance operational efficiency, customer satisfaction, and cost-effectiveness in the courier service industry, making the last mile delivery journey smoother and more efficient.

Literature Reviews

1)

The paper titled "A bi-objective optimization model for the medical supplies' simultaneous pickup and delivery with drones" explores the application of drones in emergency scenarios, such as the COVID-19 pandemic, where simultaneous pickup and delivery of medical supplies is crucial. This multi-objective problem involves achieving various goals, including minimizing route distance and efficiently utilizing the weight capacity of the drones.

To address this complex optimization challenge, the paper talks about the effectiveness of NSGA-II (Non-dominated Sorting Genetic Algorithm II). NSGA-II is

chosen for its ability to handle multi-objective problems by maintaining a diverse population of solutions and preventing the algorithm from converging to local optima prematurely.

The results obtained from using NSGA-II are highly promising, surpassing the performance of other optimization algorithms. This underscores the value of employing NSGA-II in scenarios where the simultaneous pickup and delivery of medical supplies by drones is critical, ultimately contributing to more efficient and effective emergency response strategies.

Strengths:

Diverse Solution Space: The paper effectively maintains diversity within the solution space, preventing the optimization algorithm from converging to local optima. This diversity can help find better solutions and adapt to various real-world scenarios, which is especially valuable for VRPTW problems.

Speed: The paper's choice of NSGA-II is beneficial because it is a relatively fast optimization algorithm compared to other alternatives. In practical applications, speed is often critical, especially in time-sensitive tasks like VRPTW.

Weaknesses:

Fixed Number of Trips: The paper's use of NSGA-II requires knowing the number of trips in advance, which may not be realistic for some VRPTW problems.

Relation:

This paper is relevant to the design of a Vehicle Routing Problem with Time Windows (VRPTW) and weight constraints. It provides valuable insights into multi-objective optimization using NSGA-II, which can help address the complexities of optimizing routes with both time windows and weight restrictions in VRPTW scenarios. While it excels in maintaining diverse solutions and offers speed advantages

2)

The Max-Min Ant Colony System (MM) is a variant of the Ant Colony Optimization (ACO) algorithm, specifically designed to enhance its performance. This research compares the traditional Ant System (AS) and the Max-Min Ant System in both single and multi-colonial structures. The key findings and attributes of MM are as follows:

Strengths:

The study shows that the Max-Min Ant System outperforms the original Ant System. It converges to a smaller area of the search space, closer to the optimal solution, while the Ant System stagnates in a larger search space area. Max-Min Ant System maintains a low probability of exploring solutions outside the concentrated area of the search space. This controlled exploration increases the chances of finding better solutions. MM achieves convergence, but not stagnation, resulting in a narrower range of solutions that are closer to the optimal. This, in contrast to the Ant System, which produces a wider variety of solutions but confines them to the same area of the search space.

Weaknesses

The research suggests that the Max-Min Ant System may not be effective at finding the absolute optimal solution if it is located too far from the concentrated area of the search space. The algorithm's success depends on the proximity of the optimal solution to the concentrated area.

Project Vision

1) Problem Statement

The courier service industry in Pakistan faces significant challenges, marked by inefficient parcel assignments, disregard for time windows, capacity constraints, and suboptimal routing. This results in increased operational costs, delivery delays, and dissatisfaction among riders, customers, and management.

Our project aims to resolve these issues by developing a Last Mile Delivery Optimization System, comprising an advanced algorithm that optimizes parcel assignments, time window adherence, weight capacity, and routing. This system seeks to enhance operational efficiency, reduce costs, and improve overall customer satisfaction in the courier service industry.

2) Business Opportunity

a) Operational Excellence

By meticulously optimizing parcel assignments, adhering to time windows, and ensuring capacity constraints, courier services can experience substantial

cost reductions. This heightened efficiency translates into a competitive advantage and elevated profitability.

b) Customer Loyalty and Market Expansion

With improved on-time deliveries and enhanced customer satisfaction, businesses are likely to cultivate stronger brand loyalty. This opens avenues for expanding market reach and increasing market share.

c) Data-Driven Insights

The system's analytics dashboard offers a treasure trove of data insights. This data can be leveraged for making informed strategic decisions, potentially leading to the development of new revenue streams or ancillary services.

d) Scalability and Geographic Expansion

Enhanced operational efficiency enables courier services to readily scale their operations and consider expansion into new territories, thereby seizing opportunities in untapped markets.

e) Cost-Effective Sustainability

The system contributes to sustainability efforts by optimizing routing, which in turn reduces fuel consumption and emissions. This not only aligns with environmental concerns but can also lead to cost savings, a critical aspect of any business.

f) Resource Efficiency

By preventing wasteful resource utilization, the system fosters a culture of resource optimization. This frugal approach not only conserves resources but also reduces costs.

g) Market Differentiation

Embracing cutting-edge technology for last mile delivery optimization can serve as a powerful differentiator in the competitive courier services landscape, attracting businesses and customers looking for innovative and reliable solutions.

3) Objectives

a) Optimize Last Mile Delivery Efficiency

Develop an advanced algorithm that intricately manages parcel assignments, ensuring parcels are allocated based on time windows, weight constraints, and proximity to delivery locations. The primary objective is to streamline the last mile delivery process, minimizing operational costs and enhancing efficiency.

b) Enhance Customer Satisfaction

The system will prioritize on-time deliveries and improved service quality. The goal is to significantly enhance customer satisfaction, thus fostering loyalty and positive word-of-mouth within the clientele.

c) Reduce Operational Costs

Through precise routing and capacity management, the project aims to significantly reduce operational costs associated with fuel, labor, and resources, ensuring that courier services operate more cost-effectively.

d) Data-Driven Decision-Making

Create an analytics dashboard that provides real-time data insights for administrators, enabling informed decision-making and performance monitoring. The objective is to promote data-driven strategies for improved operational outcomes.

e) Scalability and Adaptability

The system will be designed with scalability in mind, ensuring that courier services can efficiently handle varying parcel volumes and operational complexities. It should be adaptable to the dynamic needs of the industry.

f) Environmental Responsibility

By optimizing routing, the project will strive to reduce carbon emissions and environmental impact, promoting a more sustainable approach to last mile delivery.

g) Market Differentiation

Position courier services that implement the system as industry leaders, distinguishing them through cutting-edge technology, operational excellence, and a reputation for reliability.

h) Resource Optimization

Encourage a culture of prudent resource allocation, reducing waste and conserving valuable resources.

i) Revenue Diversification

Leverage data generated by the system for potential monetization opportunities, creating additional revenue streams for courier services.

j) Geographic Expansion

Enable businesses to expand into new regions by providing a scalable system capable of accommodating growth and geographic diversity.

4) Project Scope

a) Algorithm Module:

1) Algorithm Research and Implementation:Conduct research to identify suitable algorithms for parcel assignment, considering factors such as time windows, weight limits, and delivery locations.
2) Algorithm Performance Analysis and Comparison:Analyze the performance of different algorithms to determine their efficiency in optimizing parcel assignments. Compare their results, execution times, and other relevant metrics.
3) Implementing the Final Algorithm:Based on the research and analysis, implement the final algorithm that best meets the project's requirements for optimizing parcel assignments.
4) Testing and Integrating the Algorithm with the Web App:Test the implemented algorithm to ensure that it functions correctly and effectively. Integrate the algorithm with the Web App, enabling real-time parcel assignment.

b) Web App Module:

1) User Management and Authentication:Implement user registration and authentication for admin users . Develop user management features, allowing admins to add, modify, or deactivate user accounts. Ensure secure login and authentication mechanisms for access control.
2) Parcel and Route Management: Create interfaces for admins to add parcel details, including weight, time windows, and delivery locations. Implement route optimization features to generate efficient delivery routes by running the algorithm in the background Develop functionalities to allocate parcels to riders based on algorithm recommendations.
3) Analytics and Reporting: Design and develop an analytics dashboard for tracking key performance metrics, such as kilometers traveled, on-time deliveries, and rider performance.
4) Communication and Monitoring: Enable admins to communicate expected delivery times to customers via email or messages. Provide a mechanism for handling emergency requests, such as rider emergencies, and enabling admin responses. Implement a monitoring system for tracking the delivery status of each rider, including delivered and remaining parcels. Enable live tracking of rider locations and real-time monitoring of their progress

c) Mobile App Module:

1) Enabling Riders to View Trip Assignments: Riders can see trip details (route, start time, end time etc) in the app.
2) In-App Navigation: Real-time mapping for optimal delivery routes.
3) Providing Riders Ability to Update Delivery Status and Report Emergencies: Riders can update delivery status and report emergencies within the app for prompt response

5) Constraints

a) Project Constraints

i) Technological Constraints

The project's success heavily relies on the efficiency and reliability of the underlying technology, which may be subject to technical glitches, limitations, or compatibility issues.

ii) Budget Constraints

Adequate funding is essential for the project's development, testing, and deployment. Budget constraints can limit the project's ability to access necessary resources and talent.

iii) Time Constraints

Meeting project deadlines is vital, but unforeseen delays in development, testing, or implementation can be restrictive. Straying from the timeline may affect the project's overall success and efficiency.

iv) Resource Constraints

Availability of skilled developers, data scientists, and other personnel can be a limiting factor. The lack of necessary human resources may affect the project's execution.

b) Designing Constraints

i) Scalability

Ensuring that the algorithm scales effectively as parcel volumes and delivery complexities increase is a significant constraint. Inefficient scaling may lead to performance degradation.

ii) Complexity

The algorithm's complexity should be carefully managed. Excessively complex algorithms may lead to longer processing times and increased resource requirements.

iii) Time Windows and Delivery Constraints

Adhering to time windows for deliveries is a critical constraint. Deviating from specified time frames can lead to customer dissatisfaction and may be subject to regulatory constraints.

iv) Routing Constraints

Ensuring that the algorithm generates feasible routes considering road conditions, traffic, and delivery locations can be technically challenging. Unpredictable variables can impact routing accuracy.

v) Adaptability

The algorithm must be adaptable to varying parcel, rider, and delivery constraints, which can be complex to manage and implement.

c) Development Constraints

i) Integration Challenges

Integrating the new system with existing courier service infrastructure and databases may present technical challenges and constraints, particularly if legacy systems are not well-documented or compatible.

ii) Technology Stack

The project may have constraints in terms of the technology stack chosen for development. Compatibility issues, software limitations, or constraints in using specific programming languages could affect the development.

iii) Data Privacy and Security

The handling of customer and operational data necessitates stringent security and privacy measures. Constraints related to data security and privacy may impact the project's implementation.

iv) Data Availability

Availability of high-quality, relevant, and up-to-date data for algorithm training and testing may be constrained. Data acquisition and cleansing may pose challenges.

v) User Acceptance and Feedback

Developing a user-friendly interface and incorporating user feedback may pose constraints, as the system needs to meet the

expectations and preferences of various user groups, including riders and administrators.

6) Stakeholders

a) Stakeholder 1: Courier Service Administrators

i) Stakeholder's Description

Administrators are responsible for overseeing and managing courier service operations. They ensure the efficient allocation of parcels to riders, monitor delivery performance, and make strategic decisions to enhance service quality.

ii) High Key Level Goals

- Improve operational efficiency and cost-effectiveness.
- Enhance customer satisfaction by meeting delivery time windows.
- Monitor and analyze key performance metrics.
- Streamline and automate parcel assignment and routing.

iii) Stakeholder's Problems

- Inefficient parcel assignments leading to high operational costs.
- Inconsistent adherence to time windows and customer dissatisfaction.
- Limited real-time visibility into delivery performance.
- Manual and error-prone parcel allocation processes.

iv) Stakeholder's Summary

Administrators aim to optimize courier service operations, reduce costs, and elevate customer satisfaction. They seek streamlined, data-driven tools to efficiently manage parcel assignments, monitor performance, and make informed decisions.

b) Stakeholder 2: Delivery Riders

i) Stakeholder's Description

Delivery riders are the frontline workforce responsible for carrying out parcel deliveries efficiently. They navigate routes, adhere to time windows, and ensure customer satisfaction.

ii) High Key Level Goals

- Efficiently complete deliveries within specified time windows.
- Optimize routes to minimize travel time and fuel costs.
- Receive clear instructions for parcel assignments.
- Access tools for navigation and real-time updates.

iii) Stakeholder's Problems

- Unoptimized routes leading to excessive travel time.
- Difficulty in adhering to time windows and customer complaints.
- Lack of clear, real-time communication and guidance.
- Manual, time-consuming processes for reporting deliveries.

iv) Stakeholder's Summary

Delivery riders seek tools and support to optimize their routes, meet delivery time windows, and enhance their overall efficiency, contributing to customer satisfaction.

c) Stakeholder 3: Customers

i) Stakeholder's Description

Customers are the end-users of courier services. They rely on timely and reliable parcel deliveries to meet their needs and expectations.

ii) High Key Level Goals

- Receive parcels within specified time windows.
- Ensure the safety and security of delivered items.
- Experience a hassle-free and convenient delivery process.

iii) Stakeholder's Problems

- Frequent delays in parcel deliveries.
- Uncertainty about delivery time frames.
- Lack of visibility into the delivery process.
- Inconvenient or uncoordinated deliveries.

iv) Stakeholder's Summary

Customers expect reliable, on-time deliveries and a transparent, hassle-free experience. They rely on the courier service to meet these expectations.

d) Stakeholder 4: Data Scientists and Developers

i) Stakeholder's Description

Data scientists and developers are responsible for designing and implementing the optimization algorithm, creating the administrative portal, and developing the rider app.

ii) High Key Level Goals

- Design and implement an efficient optimization algorithm.
- Develop user-friendly and functional administrative and rider applications.
- Ensure the system's scalability and adaptability.
- Address data security and privacy concerns.

iii) Stakeholder's Problems

- Algorithm complexity and efficiency challenges.
- Technical constraints in app development.
- Balancing various optimization criteria.
- Ensuring data privacy and regulatory compliance.

iv) Stakeholder's Summary

Data scientists and developers aim to create a robust, efficient, and user-friendly system that addresses the complex challenges of last mile delivery optimization while ensuring data security and privacy compliance.

Software Requirement Specifications

1) List of Features

Feature Name	Short Feature Description

Algorithm	
Parcel Assignment	Assign parcels to riders based on time windows, weight limits, and proximity to delivery locations.
Time Window Management	Ensure parcels are delivered within specified time frames to meet customer expectations.
Capacity Constraint Handling	Verify parcel weight against rider weight capacity to prevent overloading.
Optimized Routing	Plan the shortest, most efficient routes to minimize travel time and distance.
Scalability	Ensure adaptability to varying parcel, rider, and delivery constraints.
Admin Portal	
Register Riders	Admin can register new riders into the system.
Route Optimization	Automatically generate optimized delivery routes for riders.
Allocate Riders	Assign parcels to riders based on the algorithm's recommendations.
Analytics Dashboard	Track kilometers traveled, monitor on-time deliveries, analyze rider performance, and estimate fuel costs.
Customer Communication	Communicate expected delivery times to customers via email or messages.
Emergency Handling	Admin can handle emergency requests such as rider emergencies.
Delivery Status Monitoring	Admin can monitor the delivery status of each rider, including delivered and remaining parcels.

Live Rider Tracking	Admin can track the live location of each rider.
Rider App	
View Trip Assignments	Riders can access trip details, including time, distance, and estimated fuel costs.
Navigation Assistance	Utilize navigation tools to follow the assigned route.
Update Delivery Status	Riders can update the delivery status (Delivered or Client Unavailable) for each parcel.
Parcel Documentation	Riders can upload photos/signatures as proof of delivered parcels.
Emergency Reporting	Riders can report emergency situations through the app.
Performance Analytics	Review personal performance analytics to track their own performance.

2) Functional Requirements

a) Functional Requirements for Parcel Assignment Algorithm

i) FR-PA-01: Parcel Assignment Criteria

- The system shall automatically assign parcels to riders based on time windows, weight limits, and proximity to delivery locations.
- The assignment should consider the weight of the parcel to ensure it doesn't exceed the rider's capacity.

ii) FR-PA-02: Time Window Enforcement

- The system shall ensure that parcels are assigned only if they meet the specified time window for delivery.
- Parcels with time windows outside the specified range should not be assigned.
- Real-time monitoring should be in place to track and handle delays and reschedule deliveries if necessary.

iii) FR-PA-03: Weight Capacity Verification

- The system shall verify that the parcel weight is within the rider's weight capacity to prevent overloading.
- Overloaded parcels should not be assigned to riders, and the system should notify the admin of such cases.
- Weight capacity should be dynamically updated based on rider load and vehicle type.

iv) FR-PA-04: Route Optimization

- The system shall use an optimized routing algorithm to determine the shortest, most efficient delivery route.

v) FR-PA-05: Adaptability

- The system shall adapt to varying parcel, rider, and delivery constraints.
- It should accommodate changes in real-time, such as new parcel arrivals, rider availability, or delivery destination alterations.
- The system should reoptimize routes and assignments as needed.

b) Functional Requirements for Admin Portal:

i) FR-AP-01: Rider Registration

- The admin portal shall provide the capability to register new riders into the system.
- Admins should input rider details, such as name, contact information, vehicle type, and availability.

ii) FR-AP-02: Route Optimization Tool

- The admin shall have access to a route optimization feature that automatically generates optimized delivery routes for riders.
- The tool should consider all pending parcel assignments and rider locations when optimizing routes.
- Admins should have the option to override or fine-tune route assignments.

iii) FR-AP-03: Parcel Allocation

- The admin portal shall allow the admin to allocate parcels to riders based on the recommendations of the parcel assignment algorithm.
- Admins can review suggested assignments and make final decisions.

iv) FR-AP-04: Analytics Dashboard

- The admin portal shall include an analytics dashboard to track kilometers traveled, monitor on-time deliveries, analyze rider performance, and estimate fuel costs.
- Admins can access historical data and real-time statistics.
- The dashboard should offer data visualization and reporting capabilities.

v) FR-AP-05: Customer Communication

- The admin shall have the ability to communicate expected delivery times to customers via email or messages.
- Automated messages should be triggered based on parcel assignment and estimated delivery times.
- Admins can send customized messages in case of delays or issues.

vi) FR-AP-06: Emergency Handling

- The admin portal shall provide functionality to handle emergency requests, such as rider emergencies.
- Admins can receive and respond to emergency notifications from riders.
- Emergency procedures should be well-documented and accessible.

vii) FR-AP-07: Delivery Status Monitoring

- The admin portal shall enable the admin to monitor the delivery status of each rider, including delivered and remaining parcels.
- Admins can track the progress of each delivery and intervene if issues arise.
- Real-time tracking and status updates should be available.

viii) FR-AP-08: Real-Time Rider Tracking

- The admin portal shall allow real-time tracking of the live location of each rider.
- Admins can view the current positions of all riders on a map.
- Location data should be updated in real-time.

c) Functional Requirements for Rider App

i) FR-RA-01: Trip Assignment Information

- The rider app shall allow riders to view their trip assignments, including details such as time, distance, and estimated fuel costs.
- Riders can access a list of assigned parcels and view key delivery information.
- Estimated fuel costs should be calculated based on the route and vehicle.

ii) FR-RA-02: Navigation Assistance

- The rider app shall provide navigation assistance to help riders follow the assigned delivery route.
- Turn-by-turn navigation instructions should be available.
- Real-time traffic and road closure updates should be integrated.

iii) FR-RA-03: Delivery Status Updates

- Riders shall be able to update the delivery status for each parcel, indicating whether it has been delivered or if the client is unavailable.
- Riders can mark parcels as "Delivered" or "Client Unavailable."

iv) FR-RA-04: Parcel Documentation

- The rider app shall allow riders to upload photos and/or signatures as proof of delivered parcels.
- Riders can take photos of delivered parcels or capture client signatures electronically.
- Uploaded documentation should be securely stored and associated with the parcel.

v) FR-RA-05: Emergency Reporting

- Riders shall have the ability to report emergency situations directly through the app.
- The app should include an emergency button or feature for quick reporting.
- Emergency reports should be transmitted to admin and support teams.

vi) FR-RA-06: Performance Analytics

- Riders can review personal performance analytics to track their own performance.
- Riders can access data on completed deliveries, delivery times, and customer satisfaction.
- The app should provide performance reports and trends.

3) Non-Functional Requirements

a) Performance

- The system shall be responsive and provide quick feedback to users.

b) Reliability

- The system shall have high availability to ensure minimal downtime.
- It should be robust and resilient, recovering gracefully from failures.

c) Security

- The system shall protect user and rider data and maintain the privacy of sensitive information.
- It should have mechanisms for user authentication and authorization.

d) Scalability

- The system should be scalable to accommodate an increasing number of riders, and parcels.
- It should adapt to changes in load without significant performance degradation.

e) Usability

- The user interfaces (admin portal and rider app) should be intuitive and user-friendly.

f) Compatibility

- The system should be compatible with various web browsers and mobile devices.
- It should support multiple operating systems and platforms.

g) Maintainability

- The system should be modular and well-documented to facilitate future maintenance and updates.
- It should support version control and change tracking.

h) Compliance

- The system should adhere to local regulations and laws related to data privacy and delivery services.
- It should maintain records and logs for auditing and compliance purposes.

i) Response Time

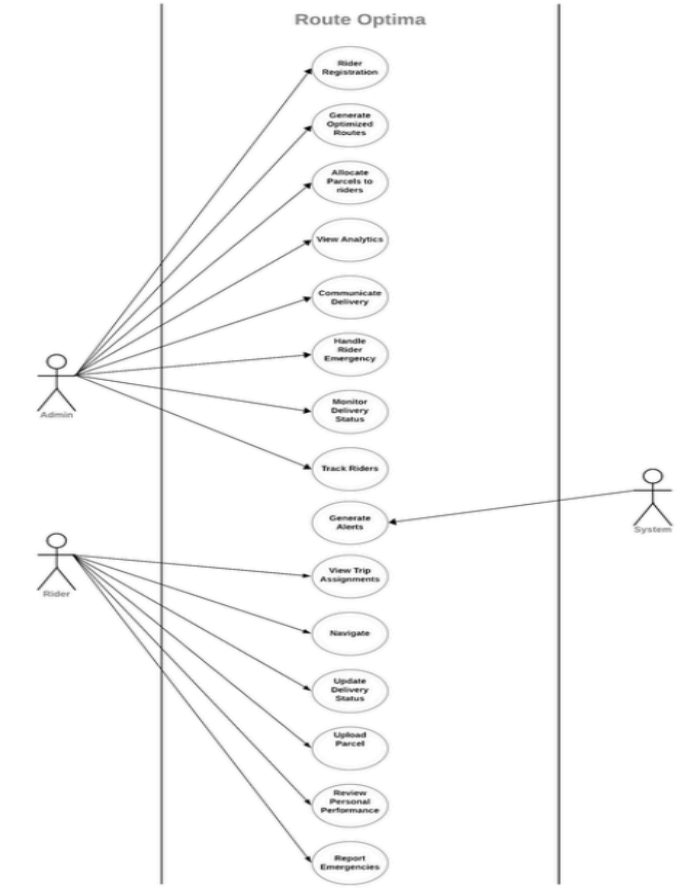
- The system should provide timely responses to user requests, ensuring a reasonable response time.
- Critical functions should have minimal response times.

j) Environmental Sustainability

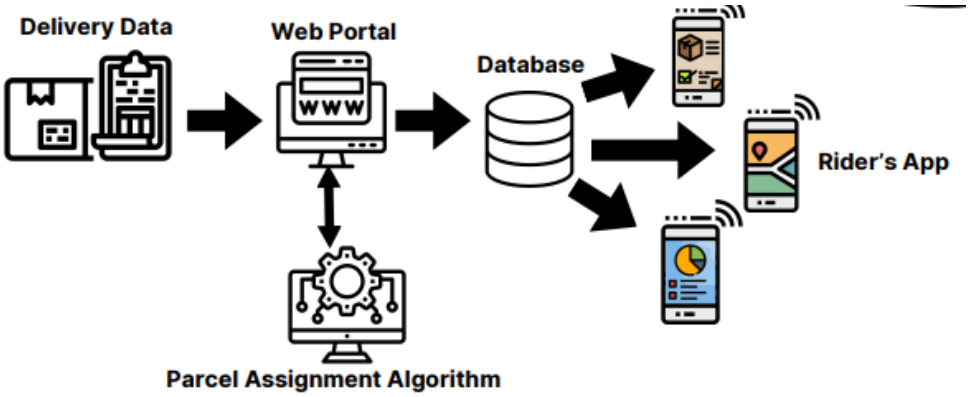
- The system should aim to minimize its environmental impact by optimizing routes and reducing unnecessary energy consumption.

Design Phase

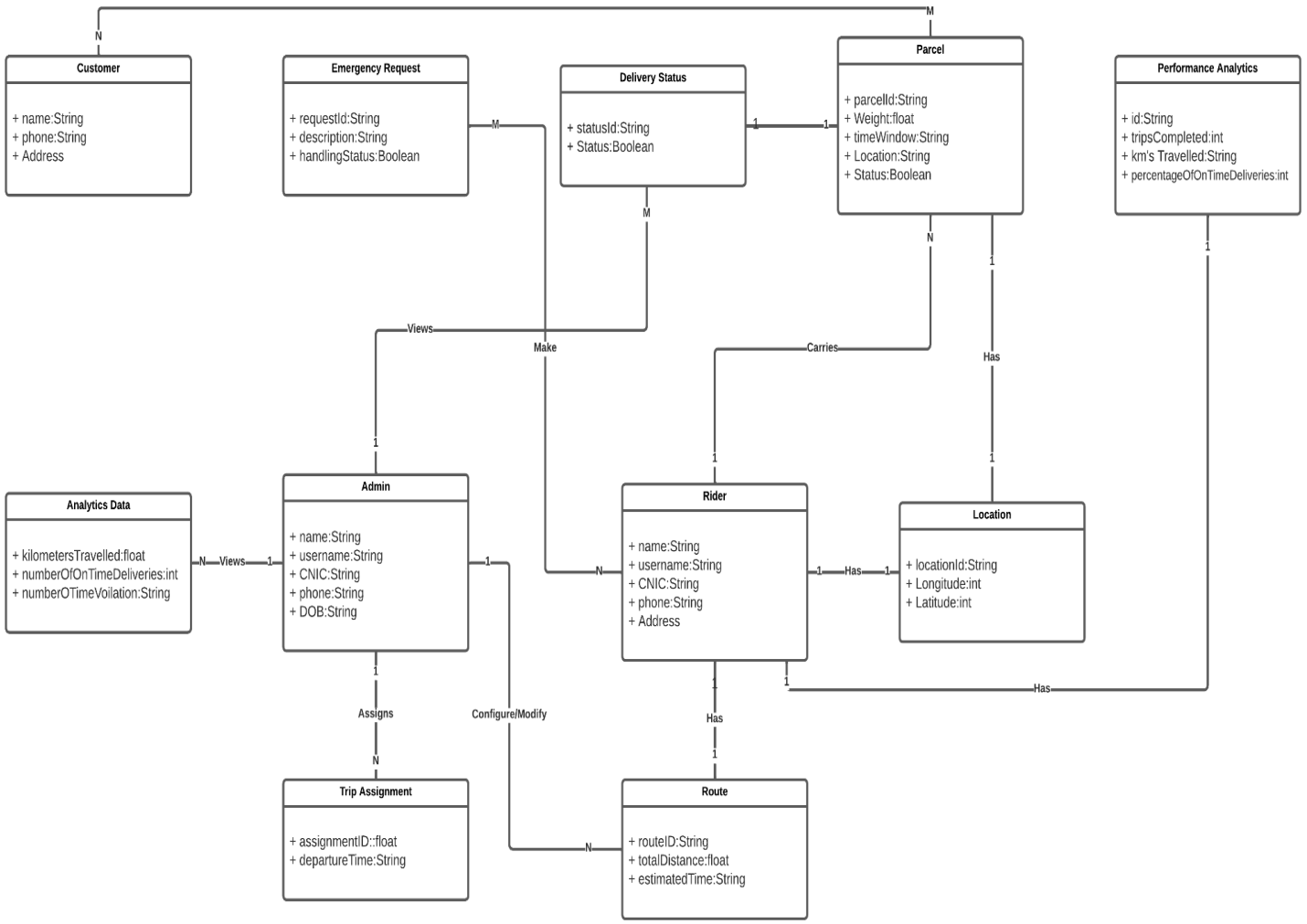
1) Use Case Diagram



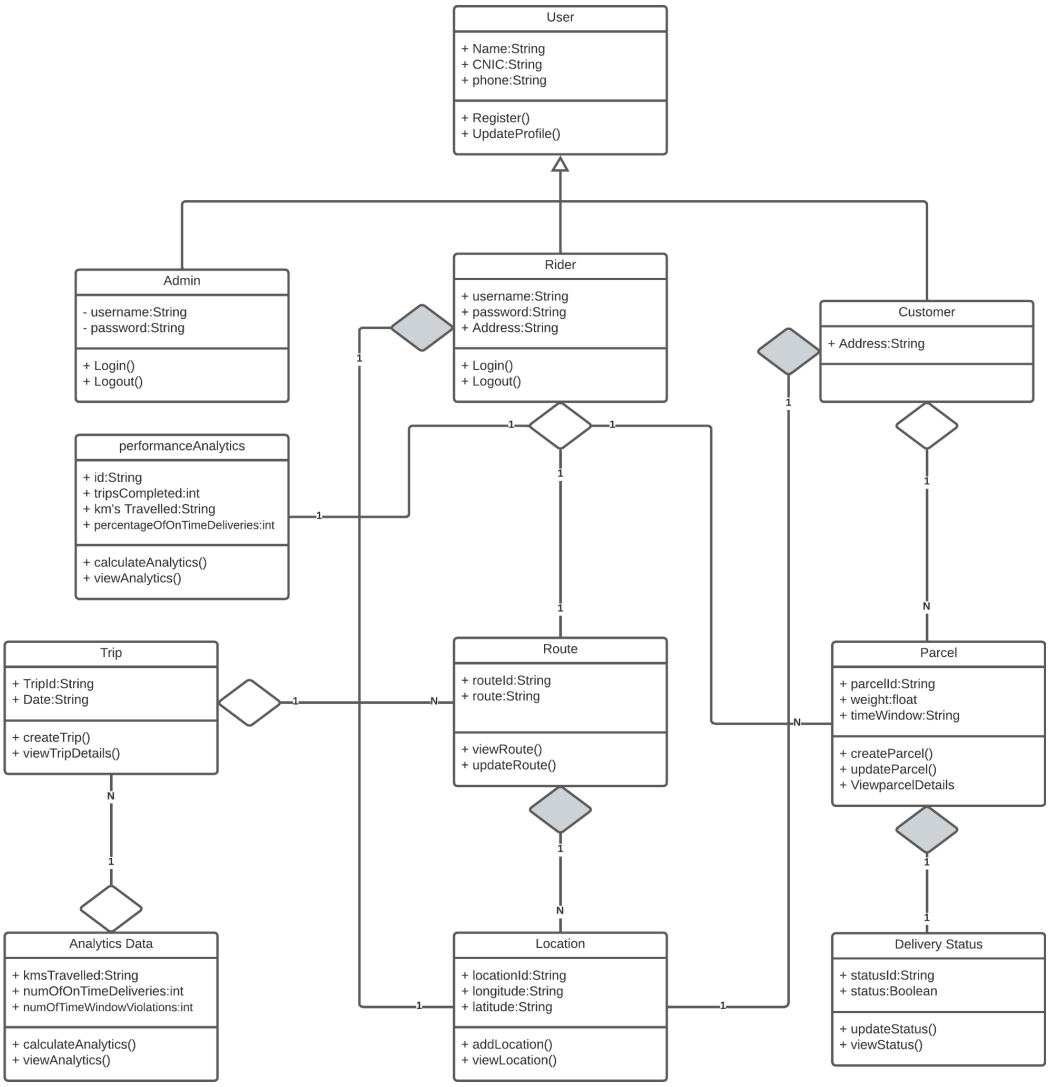
2) Architecture Diagram



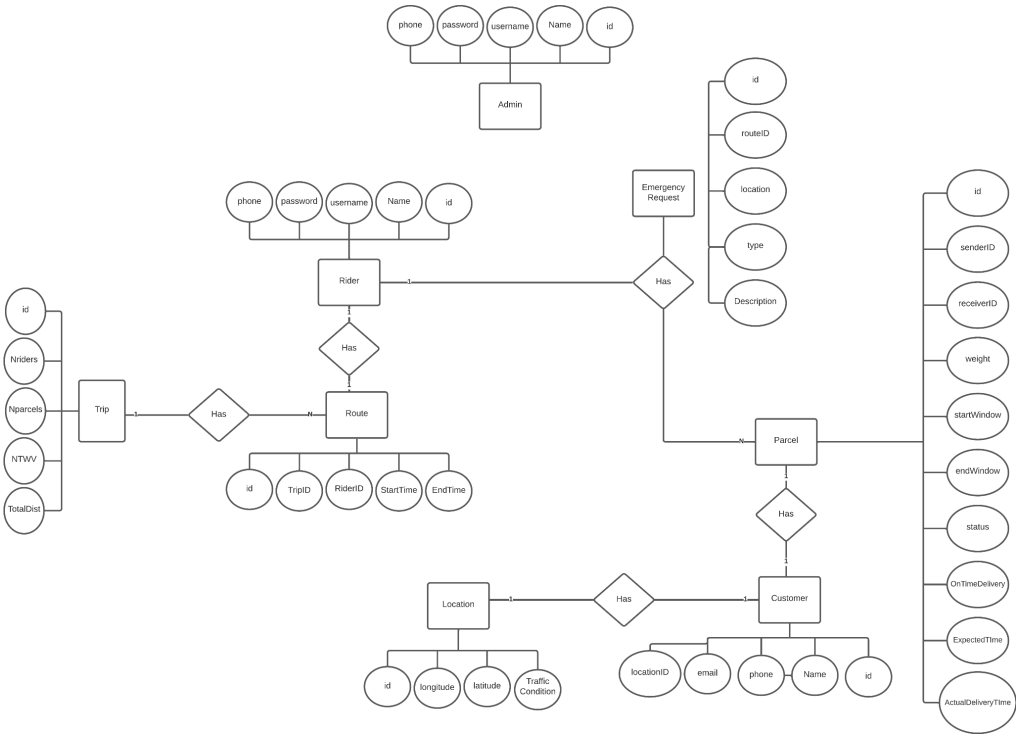
3) Domain Model



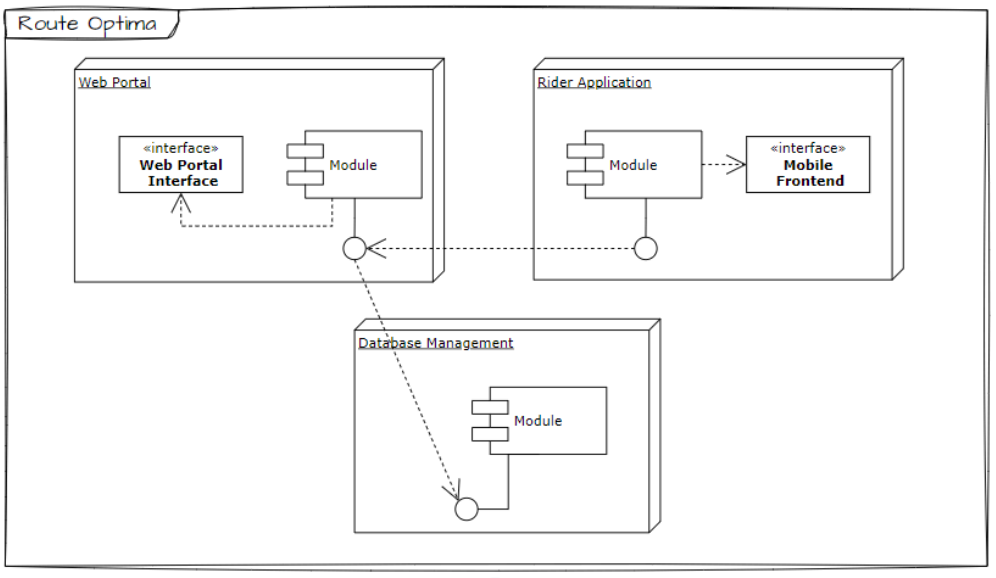
4) Class Diagram


User
+ Name:String
+ CNIC:String
+ phone:String
+ Register()
+ UpdateProfile()
Admin
- username:String
- password:String
+ Login()
+ Logout()
Rider
+ username:String
+ password:String
+ Address:String
+ Login()
+ Logout()
Customer
+ Address:String
performanceAnalytics
+ id:String
+ tripsCompleted:int
+ km's Travelled:String
+ percentageOfOnTimeDeliveries:int
+ calculateAnalytics()
+ viewAnalytics()
Trip
+ TripId:String
+ Date:String
+ createTrip()
+ viewTripDetails()
Route
+ routeId:String
+ route:String
+ viewRoute()
+ updateRoute()
Parcel
+ parcelId:String
+ weight:float
+ timeWindow:String
+ createParcel()
+ updateParcel()
+ ViewparcelDetails
Analytics Data
+ kmsTravelled:String
+ numOfOnTimeDeliveries:int
+ numOfTimeWindowViolations:int
+ calculateAnalytics()
+ viewAnalytics()
Location
+ locationId:String
+ longitude:String
+ latitude:String
+ addLocation()
+ viewLocation()
Delivery Status
+ statusId:String
+ status:Boolean
+ updateStatus()
+ viewStatus()

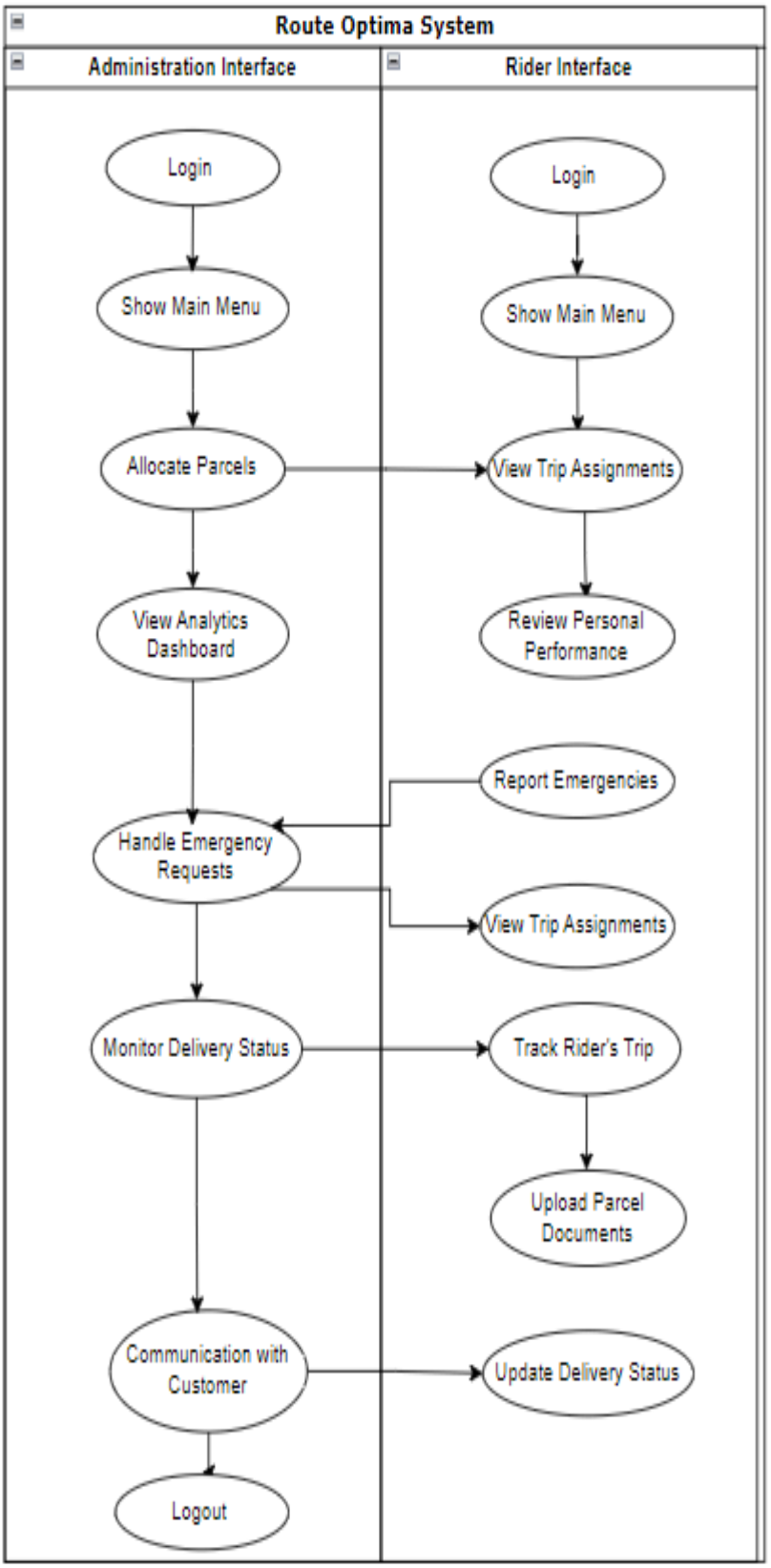
5) Entity Relationship Diagram



6) Component Diagram



7) Activity Diagram



8) Software Development Plan

a) Step 1: Project Initiation

Objective: Define the project's goals and team structure.

- Conduct a project kickoff meeting with key stakeholders to discuss the project's objectives, scope, and expectations.
- Develop a project charter specifying the high-level project scope, objectives, and initial requirements.
- Assemble the project team, including project managers, system architects, database developers, front-end and mobile app developers, algorithm developers, quality assurance team, and technical writers.
- Establish clear communication channels and choose collaboration tools for efficient team coordination.

b) Step 2: Requirement Analysis and Planning

Objective: Gather and document detailed project requirements and create a comprehensive project plan.

- Conduct detailed requirement gathering sessions with key stakeholders to understand the system's functionality and performance expectations.
- Document functional and non-functional requirements, including parcel assignment rules, time windows, weight constraints, and user interface specifications.
- Develop a project schedule, breaking it down into phases and milestones, considering an Agile development approach.
- Allocate resources based on project requirements and team availability.
- Create a risk management plan to identify potential project risks, assess their impact, and define mitigation strategies.

c) Step 3: System Design

Objective: Create the system's architecture and detailed designs.

- Design the system architecture, outlining a client-server model with web and mobile interfaces, centralized database management, and optimization algorithms.
- Define the database schema, identifying tables for parcels, riders, administrators, routes, assignments, and status tracking.
- Create user interface designs for administrators and riders, focusing on user-friendly and intuitive interfaces.
- Develop data flow diagrams and specify data storage and retrieval requirements.
- Select and document the development tools, frameworks, and technologies to be used.

d) Step 4: Development

Objective: Implement the system based on the approved design and requirements.

- Set up a version control system (e.g., Git) to facilitate collaborative development and code management.
- Develop the system components, including front-end interfaces, back-end services, database management, optimization algorithms, and mobile applications.
- Implement unit tests to ensure the functionality and integrity of individual components.
- Continuously integrate code changes and conduct code reviews to maintain code quality.
- Monitor project progress against the schedule, adjusting timelines as necessary.

e) Step 5: Quality Assurance

Objective: Ensure the system is reliable, secure, and meets quality standards.

- Develop a comprehensive testing strategy, including unit testing, integration testing, user acceptance testing, and performance testing.
- Create detailed test plans and execute test cases to verify the functionality of all system components.

- Implement a bug tracking system to log and prioritize issues, coordinating with developers for timely resolutions.
- Perform rigorous testing to validate the system's stability, scalability, and security.

f) Step 6: Deployment

Objective: Prepare the system for deployment and a smooth rollout.

- Develop a deployment plan, including staging, testing, and production release phases.
- Gradually roll out the system to ensure a seamless transition and user adoption.
- Monitor the deployment process closely, addressing any deployment-related issues promptly.
- Execute data migration if needed to ensure accurate data transfer.

g) Step 7: Maintenance and Support

Objective: Provide ongoing support, monitoring, and plan for future enhancements.

- Establish a dedicated support team to address user concerns and system issues post-deployment.
- Continuously monitor system performance, address potential bottlenecks, and optimize system responsiveness.
- Plan for future system enhancements and updates to meet evolving business requirements and user needs.

h) Step 8: Project Closure

Objective: Close the project and evaluate its success.

- Seek client acceptance, ensuring the system meets client satisfaction and requirements.
- Conduct a post-implementation review to evaluate the project's overall success and identify lessons learned.
- Deliver final documentation, including source code repositories, to the client for future reference and system maintenance.

9) Iteration Plan

Iteration-1:

a) Iteration-1 Overview

Objective: This iteration focuses on designing the core algorithms for parcel assignment, time window management, capacity constraints, and optimized routing. Additionally, it involves creating basic wireframes for the web portal and rider app interfaces to visualize the user experience.

Duration: 4 weeks

b) Tasks and Milestones

i) Week 1

Algorithm Design: Parcel Assignment

- Define the algorithm for parcel assignment based on time windows, weight limits, and proximity.
- Conduct a technical review of the algorithm with the development team.
- Document the algorithm's flowchart and logic.

Web Portal Wireframes: Initial Concepts

- Create initial wireframes for the admin portal, focusing on layout and key features.
- Review and validate wireframes with the project manager and UX/UX designers.

ii) Week 2

Algorithm Design: Time Window Management

- Develop the algorithm for managing time windows to ensure on-time deliveries.
- Test the algorithm using sample data to validate its accuracy.
- Document the time window management logic and rules.

Rider App Wireframes: Initial Concepts

- Create wireframes for the rider app, emphasizing essential trip information and navigation.

- Review wireframes with the project manager, UI/UX designers, and rider app developers.

iii) Week 3

Algorithm Design: Capacity Constraint Handling

- Design the algorithm to verify parcel weight against rider weight capacity to prevent overloading.
- Conduct testing with various weight scenarios to validate the algorithm's functionality.
- Document the capacity constraint handling logic.

Web Portal Wireframes: Detailed Design

- Refine and expand web portal wireframes, including user interaction flows.
- Ensure wireframes accurately represent user tasks and system features.
- Review wireframes with the entire development team.

iv) Week 4

Algorithm Design: Optimized Routing

- Plan the algorithm for generating the shortest, most efficient routes for riders.
- Perform testing with sample routes and geographical data to validate efficiency.z
- Document the routing algorithm's logic and integration points.

Rider App Wireframes: Detailed Design

- Extend and refine rider app wireframes to include detailed trip information, delivery status updates, and reporting features.
- Review wireframes with the project manager, UI/UX designers, and rider app developers.
- Iteration Review and Feedback

Iteration-2:

a) Iteration-2 Overview

Objective: Over the past four weeks, our main goals were to set up the basics of our delivery management system. In the first week, we focused on creating the web portal with features like letting riders sign up and connecting the route generation algorithm. At the same time, we worked on making the rider app easy to use. In the second week, we made things more efficient by allowing the web portal to assign parcels to riders. On the rider app, riders could now see their trip assignments. In the third week, we improved communication by making the web portal tell customers when to expect their deliveries, and riders could report emergencies using the app. Finally, in the fourth week, the system was ready to handle emergency requests on the web portal, and the rider app allowed real-time updates on delivery statuses. These goals helped us build a strong and practical system to make parcel delivery smoother. We tested things regularly, got feedback, and kept documents updated to make sure everything worked well.

Duration: 4 weeks

b) Tasks and Milestones

i) Week 1

***Web Portal**:*

- Rider Registration
- Route Generation (Integration of algorithm with the web portal)

***Rider App**:*

- UI for the mobile app

***Algorithm Module**:*

- Designed brute force algorithm for comparison with heuristics

ii) Week 2

***Web Portal**:*

- Allocate Parcels to the riders

***Rider App**:*

- View Trip Assignments

iii) Week 3

***Web Portal**:*

- Communicate expected delivery times to the customers

***Rider App**:*

- Report Emergency

iv) Week 4

***Web Portal**:*

- Handle Emergency requests

***Rider App**:*

- Dashboard for riders

***Algorithm Module**:*

- Comparison of brute force algorithm's results with heuristics'

Iteration-3:

a) Iteration-3 Overview

Over the next four weeks, we'll be enhancing our delivery system. In the first week, we'll set up an Admin Portal with a dashboard for quick insights and start monitoring rider performance. The Rider App will get a better look, and we'll begin adding navigation features. In the second week, we'll improve the Admin Portal with advanced rider tracking and visual tools. The Rider App will focus on smoother trips and the ability to upload photos and signatures for delivered parcels. Week three is about finalizing navigation and enabling proof of delivery uploads for the Rider App. The last week will be all about testing and refining everything.

b) Tasks and MileStones

i) Week 1

***Admin Portal**:*

- Implement Analytics Dashboard for real-time insights.

***Rider App**:*

- Begin integration of navigation features for efficient trip planning.
- Update Delivery Status of parcels.

ii) Week 2

***Admin Portal**:*

- Rider monitoring, ensuring accurate tracking and status updates.

***Rider App**:*

- Continue refining navigation features for seamless trip execution.
- Initiate the implementation of photo and signature uploads for delivered parcels.

iii) Week 3

Admin Portal:

- Review and refine the Analytics Dashboard based on user feedback.

***Rider App**:*

- Finalize navigation features for optimal route planning and execution.
- Enable riders to upload photos and digital signatures for parcels successfully delivered.

iv) Week 4

***Admin Portal**:*

- Conduct comprehensive testing, addressing any issues with analytics or monitoring features.

***Rider App**:*

- Conduct thorough testing of all features and address any outstanding issues.

10) Use Cases

Use Case	Actor	Description

Register Riders	Admin	Admins can register new riders into the system.
Generate Optimized Routes	Admin	Admins can generate optimized delivery routes for riders.
Allocate Parcels	Admin	Admins assign parcels to riders based on the recommendations of the parcel assignment algorithm.
View Analytics Dashboard	Admin	Admins can access an analytics dashboard to track kilometers traveled, monitor on-time deliveries, analyze rider performance, and estimate fuel costs.
Communicate Expected Delivery Times	Admin	Admins can communicate expected delivery times to customers via email or messages.
Handle Emergency Requests	Admin	Admins can handle emergency requests, such as rider emergencies.
Monitor Delivery Status	Admin	Admins can monitor the delivery status of each rider, including delivered and remaining parcels.
Track Live Location	Admin	Admins can track the live location of each rider.
View Trip Assignments	Rider	Riders can view their trip assignments, including details such as time, distance, and estimated fuel costs.
Utilize Navigation	Rider	Riders utilize the navigation feature to follow the assigned delivery route.
Update Delivery Status	Rider	Riders update the delivery status for each parcel, marking them as delivered or client unavailable.
Upload Parcel Documentation	Rider	Riders upload photos and/or signatures as proof of delivered parcels.

Report Emergencies	Rider	Riders can report emergency situations directly through the app.
Review Personal Performance	Rider	Riders can review personal performance analytics, tracking their own performance.

a) High Level Use Cases

i) Register Riders

Use Case ID	UC-1
Actor	Admin
Type	Primary
Description	Admins can register new riders into the system.

ii) Generate Optimized Routes

Use Case ID	UC-2
Actor	Admin
Type	Primary
Description	Admins can generate optimized delivery routes for riders.

iii) Allocate Parcels

Use Case ID	UC-3
Actor	Admin

Type	Primary
Description	Admins assign parcels to riders based on the recommendations of the parcel assignment algorithm.

iv) View Analytics Dashboard

Use Case ID	UC-4
Actor	Admin
Type	Primary
Description	Admins can access an analytics dashboard to track kilometers traveled, monitor on-time deliveries, analyze rider performance, and estimate fuel costs.

v) Communicate Expected Delivery Times

Use Case ID	UC-5
Actor	Admin
Type	Primary
Description	Admins can communicate expected delivery times to customers via email or messages.

vi) Handle Emergency Requests

Use Case ID	UC-6
Actor	Admin

Type	Primary
Description	Admins can handle emergency requests, such as rider emergencies.

vii) Monitor Delivery Status

Use Case ID	UC-7
Actor	Admin
Type	Primary
Description	Admins can monitor the delivery status of each rider, including delivered and remaining parcels.

viii) Track Live Location

Use Case ID	UC-8
Actor	Admin
Type	Primary
Description	Admins can track the live location of each rider.

ix) View Trip Assignments

Use Case ID	UC-9
Actor	Rider

Type	Primary
Description	Riders can view their trip assignments, including details such as time, distance, and estimated fuel costs.

x) Utilize Navigation

Use Case ID	UC-10
Actor	Rider
Type	Primary
Description	Riders utilize the navigation feature to follow the assigned delivery route.

xi) Update Delivery Status

Use Case ID	UC-11
Actor	Rider
Type	Primary
Description	Riders update the delivery status for each parcel, marking them as delivered or client unavailable.

xii) Upload Parcel Documentation

Use Case ID	UC-12
Actor	Rider
Type	Primary

Description	Riders upload photos and/or signatures as proof of delivered parcels.

xiii) Report Emergencies

Use Case ID	UC-13
Actor	Rider
Type	Primary
Description	Riders can report emergency situations directly through the app.

xiv) Review Personal Performance

Use Case ID	UC-14
Actor	Rider
Type	Primary
Description	Riders can review personal performance analytics, tracking their own performance.

b) Extended Use Cases

i) Register Riders

Use Case ID	**UC-1**
Actor	Admin
Type	Primary

Description	Admins can register new riders into the system.
Precondition	The admin has the necessary information to register a new rider.
Postcondition	The new rider is successfully registered in the system.
Normal Flow	
	1. The admin accesses the rider registration functionality.
	2. The admin enters the rider's personal and contact information.
	3. The system validates the information provided by the admin.
	4. The system registers the new rider in the system.
Alternative Flow	If the provided information is incomplete or invalid, the system displays an error message to the admin.
Exception Flow	If there are technical issues during the registration process, the system logs the error and notifies the admin

ii) Generate Optimized Routes

Use Case ID	UC-2
Actor	Admin
Type	Primary
Description	Admins can generate optimized delivery routes for riders.

Precondition	The admin has access to the route optimization functionality.
Postcondition	Optimized delivery routes are generated for riders.
Normal Flow	
	1. The admin accesses the route optimization tool.
	2. The admin selects the parcels and riders for which routes need to be generated.
	3. The system calculates the shortest and most efficient routes for each rider.
	4. The system displays the optimized routes to the admin.
Alternative Flow	If there are no parcels or riders available for route optimization, the system notifies the admin.
Exception Flow	If there are errors in route calculations, the system logs the error and notifies the admin.

iii) Allocate Parcels

Use Case ID	UC-3
Actor	Admin
Type	Primary
Description	Admins assign parcels to riders based on the recommendations of the parcel assignment algorithm.

Precondition	The admin has access to the parcel allocation functionality.
Postcondition	Parcels are successfully assigned to riders.
Normal Flow	
	1. The admin accesses the parcel allocation tool.
	2. The admin selects the parcels that need to be assigned.
	3. The system recommends the assignment of parcels to available riders.
	4. The admin approves the parcel assignments.
Alternative Flow	. If there are no available riders for parcel assignment, the system notifies the admin.
Exception Flow	If there are technical issues during the assignment process, the system logs the error and notifies the admin.

iv) View Analytics Dashboard

Use Case ID	UC-4
Actor	Admin
Type	Primary
Description	Admins can access an analytics dashboard to track kilometers traveled, monitor on-time deliveries, analyze rider performance, and estimate fuel costs.

Precondition	The admin has access to the analytics dashboard.
Postcondition	The admin has viewed the analytics data.
Normal Flow	
	1. The admin accesses the analytics dashboard.
	2. The system displays analytics data, including kilometers traveled, on-time deliveries, rider performance, and fuel costs.
	3. The admin reviews the analytics data.
Alternative Flow	If there is no relevant analytics data available, the system notifies the admin.
Exception Flow	If there are technical issues while accessing analytics data, the system logs the error and notifies the admin.

v) Communicate Expected Delivery Times

Use Case ID	UC-5
Actor	Admin
Type	Primary
Description	Admins can communicate expected delivery times to customers via email or messages.
Precondition	The admin has access to the communication functionality.
Postcondition	Expected delivery times are communicated to customers.

Normal Flow	1. The admin selects parcels with specified delivery times.
	2. The admin chooses the communication method (email or messages).
	3. The system sends expected delivery times to customers.
Alternative Flow	If there are no parcels with specified delivery times, the system notifies the admin.
Exception Flow	If there are technical issues during the communication process, the system logs the error and notifies the admin.

vi) Handle Emergency Requests

Use Case ID	UC-6
Actor	Admin
Type	Primary
Description	Admins can handle emergency requests, such as rider emergencies.
Precondition	The admin has access to the emergency request functionality.
Postcondition	Emergency requests are successfully handled.
Normal Flow	
	1. The admin receives an emergency request from a rider.
	2. The admin assesses the nature of the emergency and takes appropriate action.

	3. The system records the handling of the emergency request.
Alternative Flow	If there are no emergency requests, the system notifies the admin.
Exception Flow	If there are technical issues during the handling of an emergency request, the system logs the error and notifies the admin.

vii) Monitor Delivery Status

Use Case ID	UC-7
Actor	Admin
Type	Primary
Description	Admins can monitor the delivery status of each rider, including delivered and remaining parcels.
Precondition	The admin has access to the delivery status monitoring functionality.
Postcondition	The admin has monitored the delivery status.
Normal Flow	
	1. The admin accesses the delivery status monitoring tool.
	2. The system displays the status of each rider, including delivered parcels and remaining parcels.
	3. The admin reviews the delivery status.
Alternative Flow	If there are no riders with assigned parcels, the system notifies the admin.
Exception Flow	If there are technical issues while monitoring the delivery status, the system logs the error and notifies the admin.

viii) Track Live Location

Use Case ID	UC-8
Actor	Admin
Type	Primary
Description	Admins can track the live location of each rider.
Precondition	The admin has access to the live location tracking functionality.
Postcondition	The admin has tracked the live location of riders.
Normal Flow	
	1. The admin accesses the live location tracking tool.
	2. The system displays the real-time location of each rider on a map.
	3. The admin tracks the live location of riders.
Alternative Flow	If there are no riders with active trips, the system notifies the admin.
Exception Flow	If there are technical issues with location tracking, the system logs the error and notifies the admin.

ix) View Trip Assignments

Use Case ID	UC-9

Actor	Rider
Type	Primary
Description	Riders can view their trip assignments, including details such as time, distance, and estimated fuel costs.
Precondition	The rider has access to the trip assignment view.
Postcondition	The rider has viewed trip assignments.
Normal Flow	
	1. The rider accesses the trip assignment view in the app.
	2. The system displays the details of assigned trips, including time, distance, and estimated fuel costs.
	3. The rider reviews trip assignments.
Alternative Flow	If there are no assigned trips, the system notifies the rider.
Exception Flow	If there are technical issues with displaying trip assignments, the system logs the error and notifies the rider.

x) Utilize Navigation

Use Case ID	UC-10
Actor	Rider
Type	Primary

Description	Riders utilize the navigation feature to follow the assigned delivery route.
Precondition	The rider has access to the navigation feature.
Postcondition	The rider has followed the delivery route.
Normal Flow	
	1. The rider selects a trip assignment and chooses to navigate.
	2. The system provides turn-by-turn navigation instructions.
	3. The rider follows the navigation to the delivery location.
Alternative Flow	If the rider chooses not to use navigation, they proceed with the delivery based on their knowledge.
Exception Flow	If there are technical issues with the navigation feature, the system logs the error and notifies the rider.

xi) Update Delivery Status

Use Case ID	UC-11
Actor	Rider
Type	Primary
Description	Riders update the delivery status for each parcel, marking them as delivered or client unavailable.
Precondition	The rider has successfully delivered a parcel.
Postcondition	The delivery status is updated in the system.

Normal Flow	
	1. The rider selects a delivered parcel.
	2. The system allows the rider to update the status as "delivered" or "client unavailable."
	3. The rider updates the delivery status.
Alternative Flow	If there are parcels that were not delivered, the rider leaves them as "pending."
Exception Flow	If there are technical issues with updating the delivery status, the system logs the error and notifies the rider.

xii) Upload Parcel Documentation

Use Case ID	UC-12
Actor	Rider
Type	Primary
Description	Riders upload photos and/or signatures as proof of delivered parcels.
Precondition	The rider has successfully delivered a parcel.
Postcondition	The documentation is uploaded as proof of delivery.
Normal Flow	
	1. The rider selects a delivered parcel.

	2. The system allows the rider to upload photos or signatures as proof of delivery.
	3. The rider uploads the documentation.
Alternative Flow	. If no documentation is available or required, the rider proceeds without uploading.
Exception Flow	If there are technical issues with uploading documentation, the system logs the error and notifies the rider.

xiii) Report Emergencies

Use Case ID	UC-13
Actor	Rider
Type	Primary
Description	Riders can report emergency situations directly through the app.
Precondition	The rider encounters an emergency situation during a trip.
Postcondition	The emergency report is received and addressed.
Normal Flow	
	1. The rider accesses the emergency reporting feature in the app.
	2. The rider provides details of the emergency situation.
	3. The system receives and forwards the report to the appropriate authorities.

Alternative Flow	If there are no emergency situations, the rider does not use the reporting feature.
Exception Flow	If there are technical issues with reporting emergencies, the system logs the error and notifies the rider and authorities.

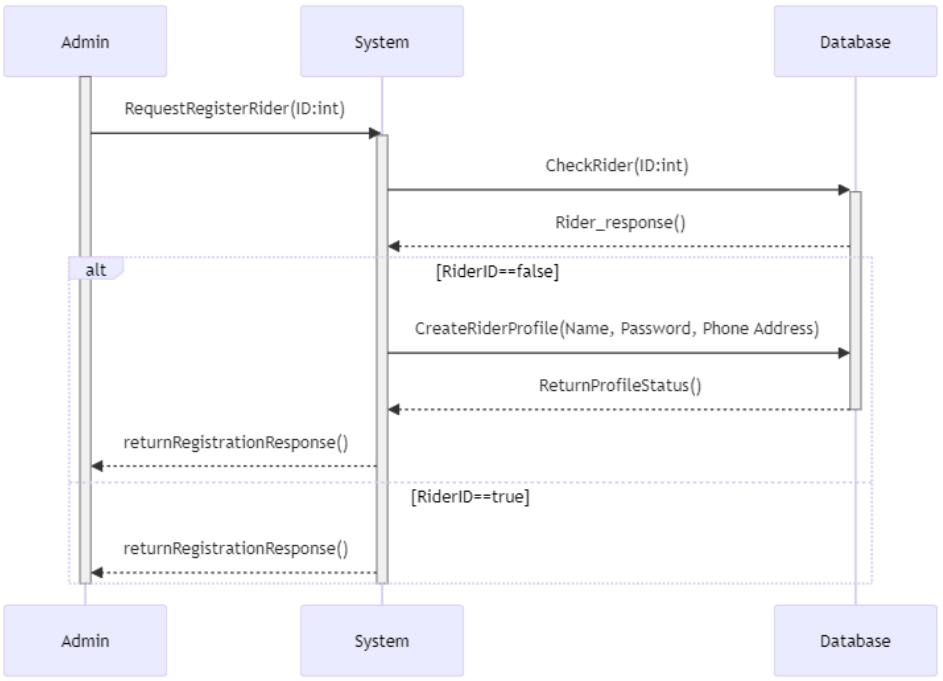
xiv) Review Personal Performance

Use Case ID	UC-14
Actor	Rider
Type	Primary
Description	Riders can review personal performance analytics, tracking their own performance.
Precondition	The rider has access to personal performance analytics.
Postcondition	The rider has reviewed their performance data.
Normal Flow	
	1. The rider accesses the personal performance analytics in the app.
	2. The system displays performance data, including trips completed, on-time deliveries, and other relevant metrics.
	3. The rider reviews their personal performance.
Alternative Flow	If there is no performance data available, the system notifies the rider.

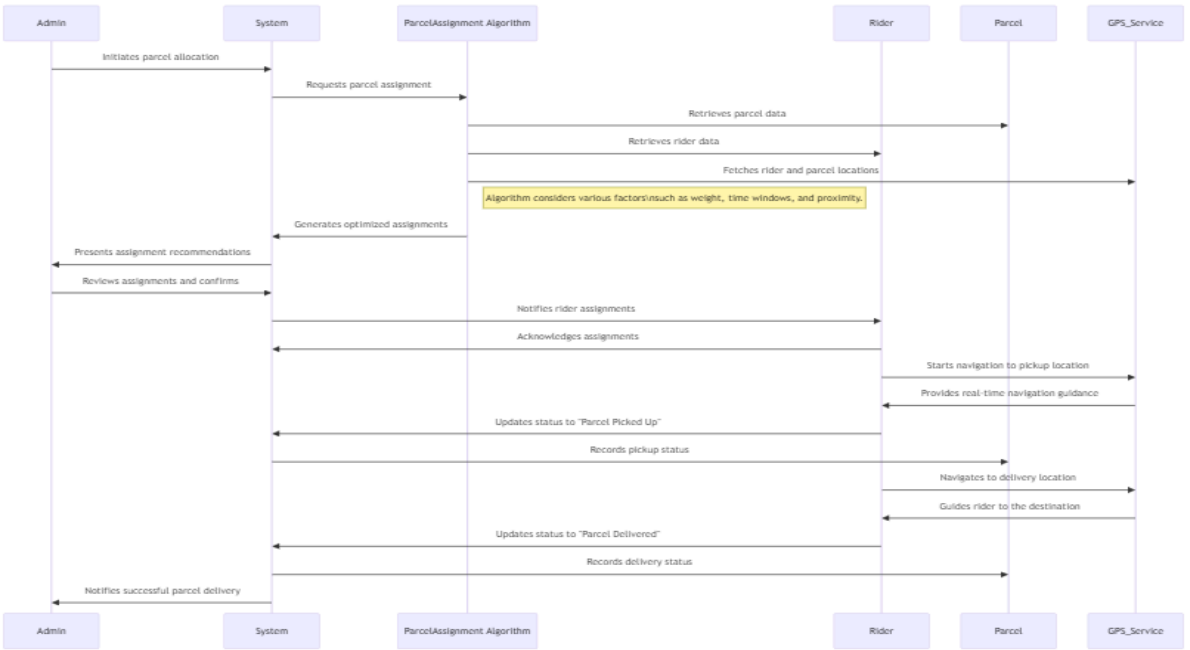
Exception Flow	If there are technical issues with accessing performance data, the system logs the error and notifies the rider.

11) Sequence Diagram

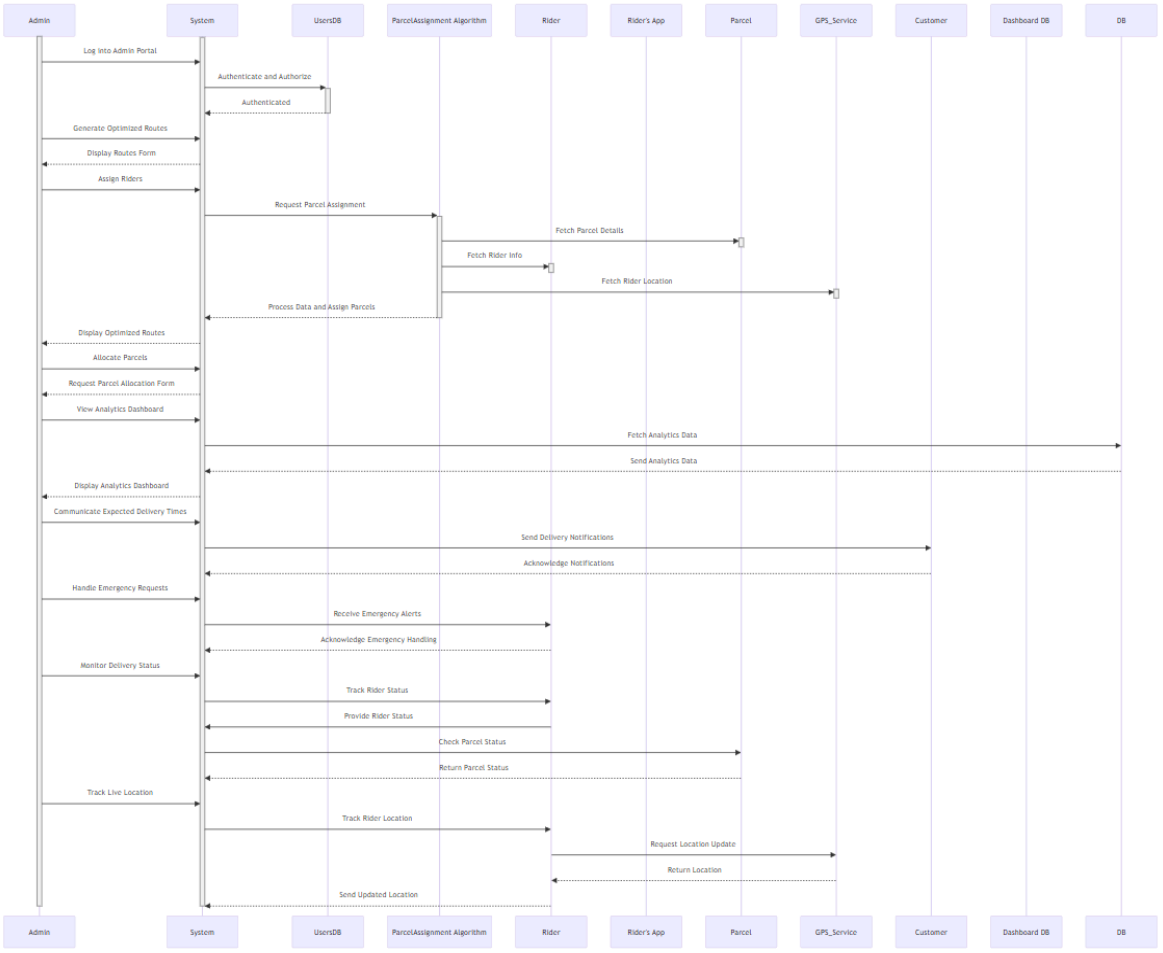
a) Register Riders



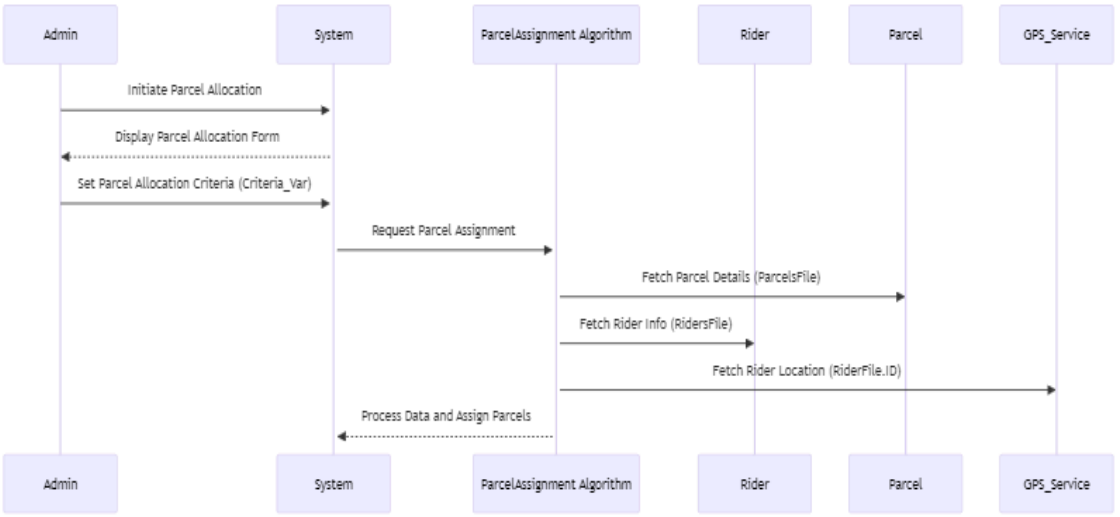
b) Assign Riders



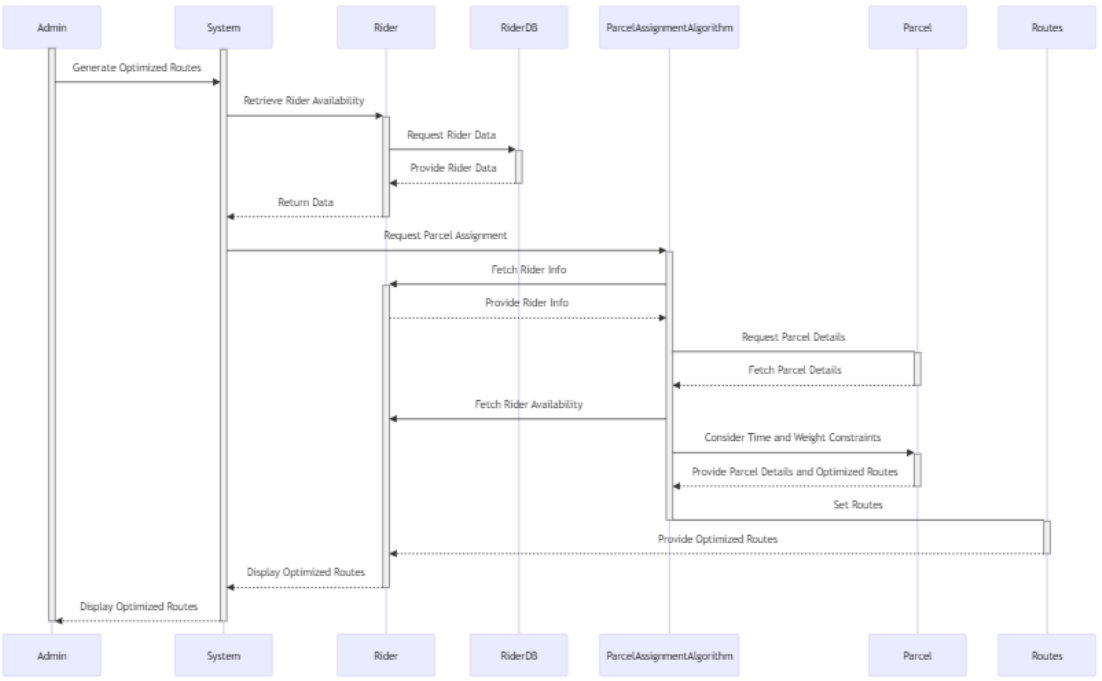
c) Login Portal and Major Navigation Bar Functions



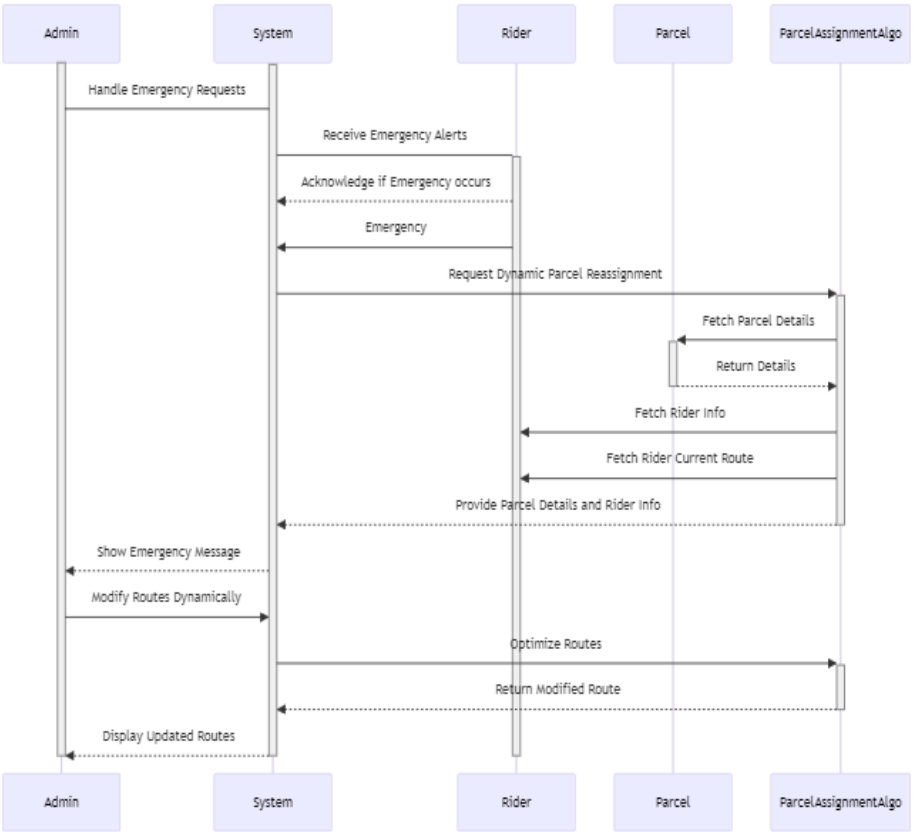
d) Allocate Parcels



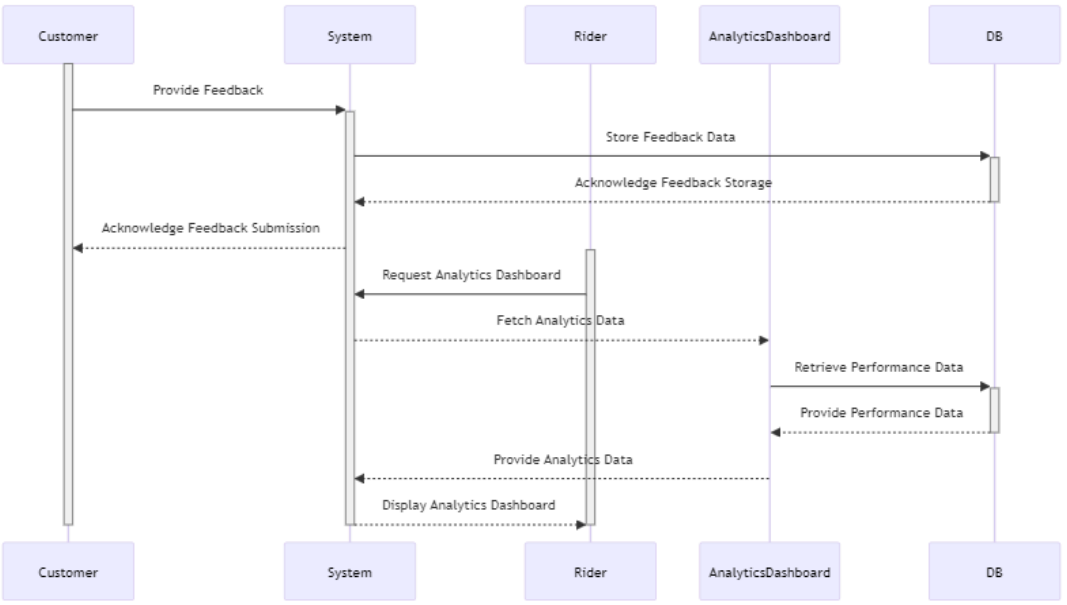
e) Rider Availability and Route Generation



f) Emergency Routes Handling

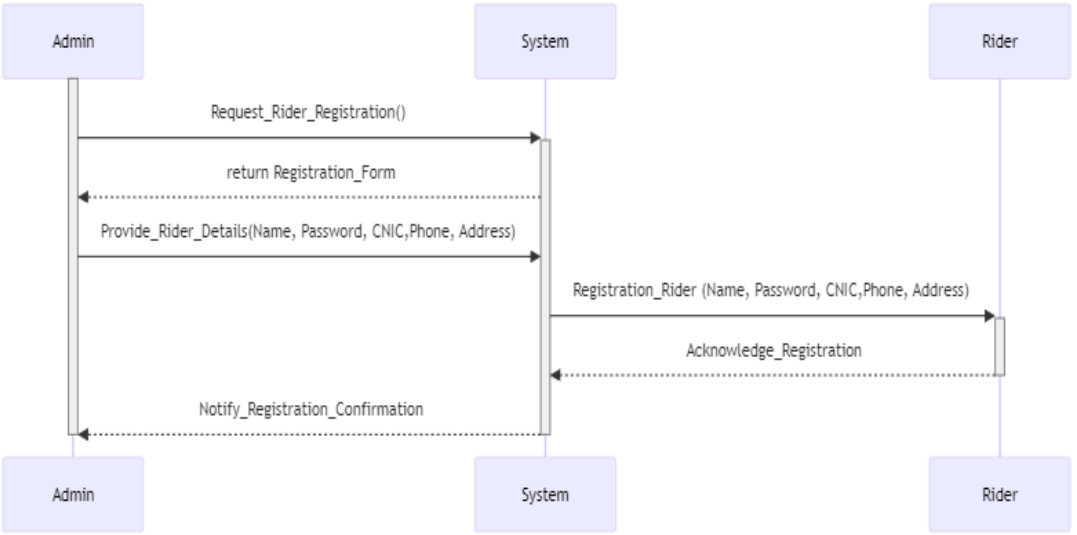


g) Feedback System and Analytics

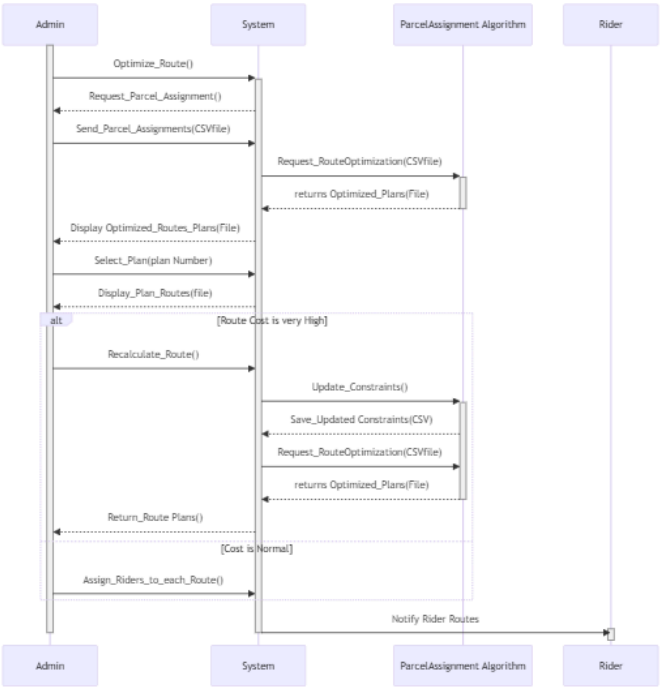


12) System Sequence Diagram

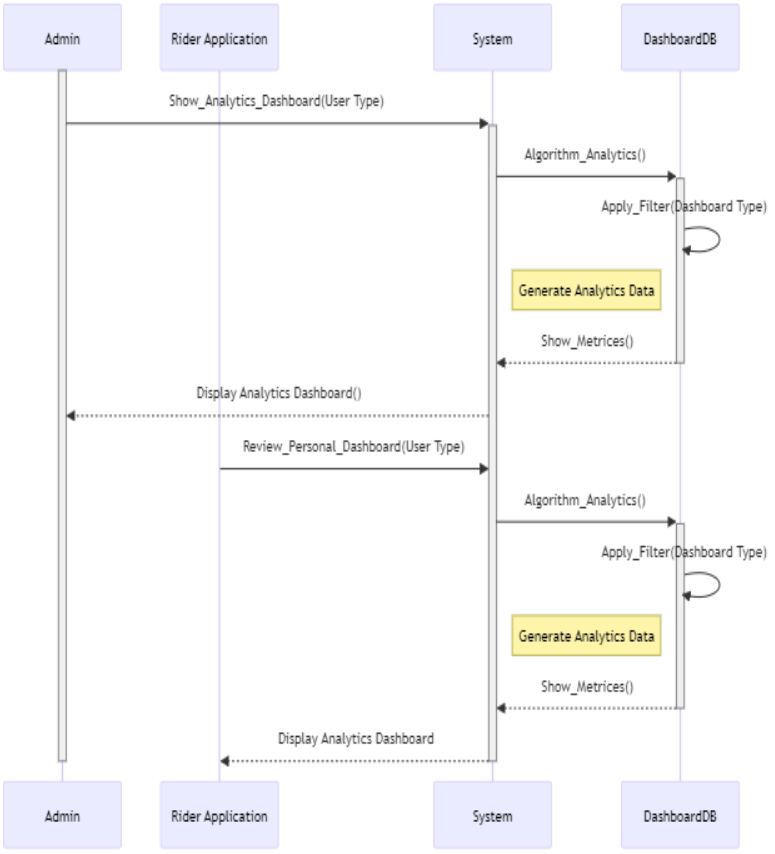
a) Register Riders



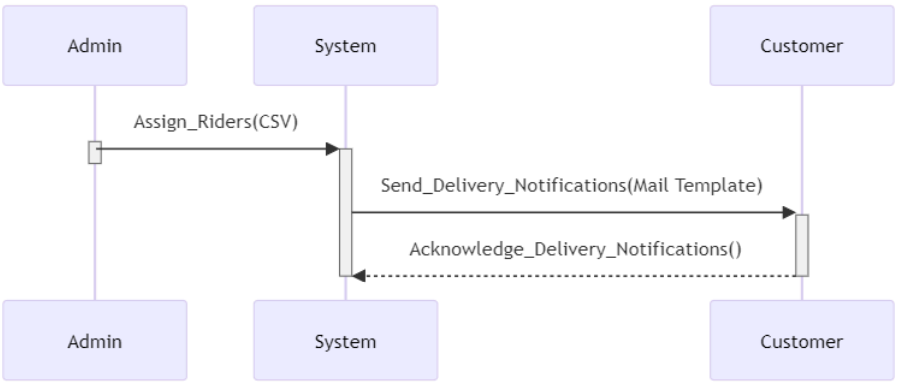
b) Generate Optimized Route



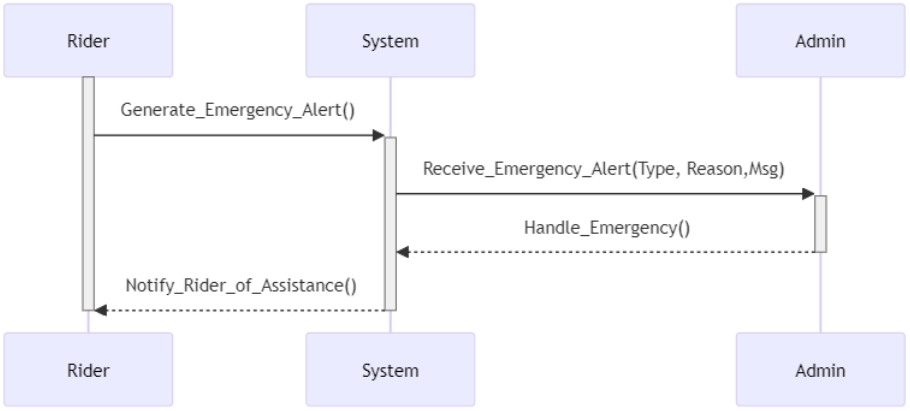
c) View Analytics Dashboard



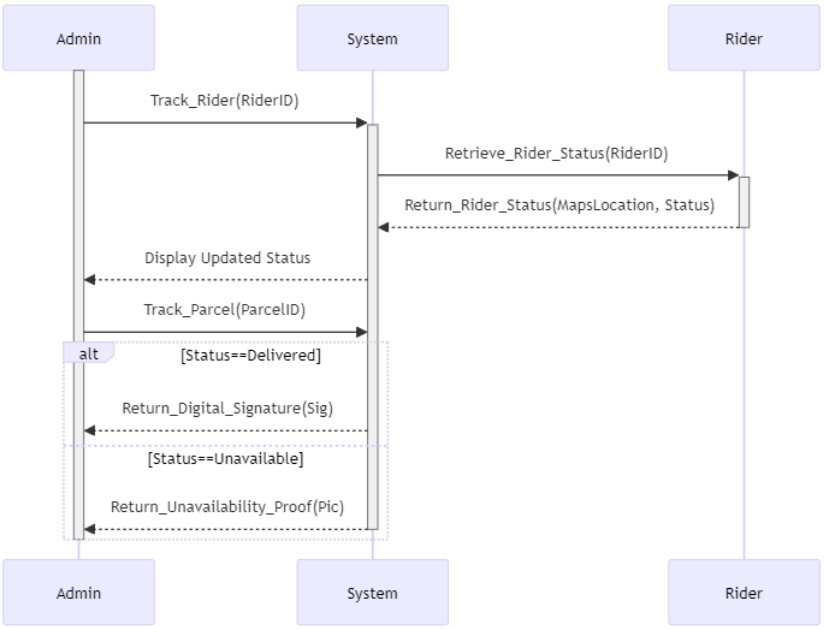
d) Communicate Expected Delivery Times



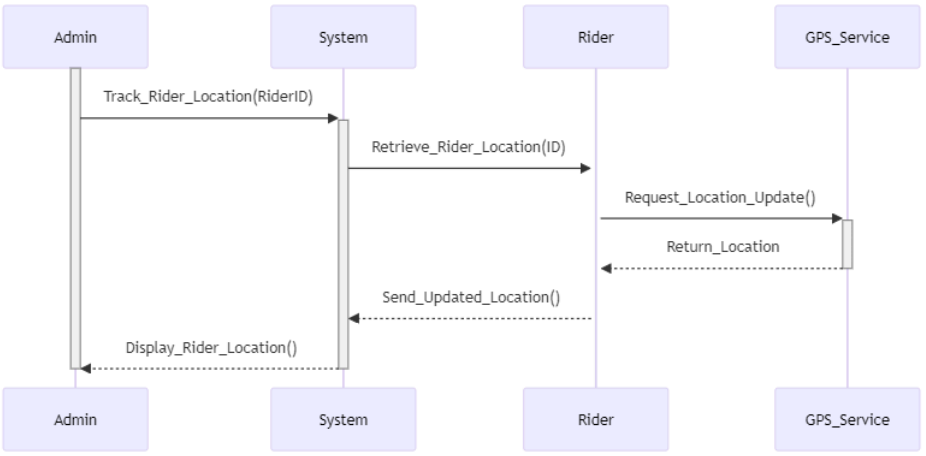
e) Handle Emergency Requests



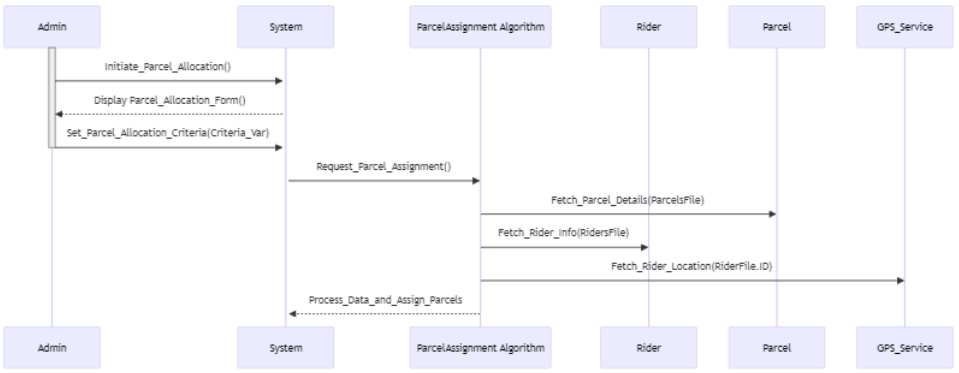
f) Monitor Delivery Status



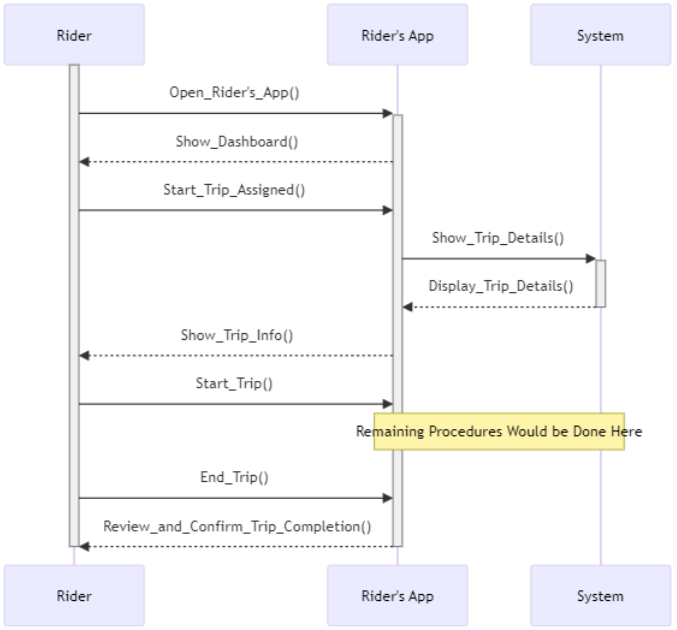
g) Track Live Location



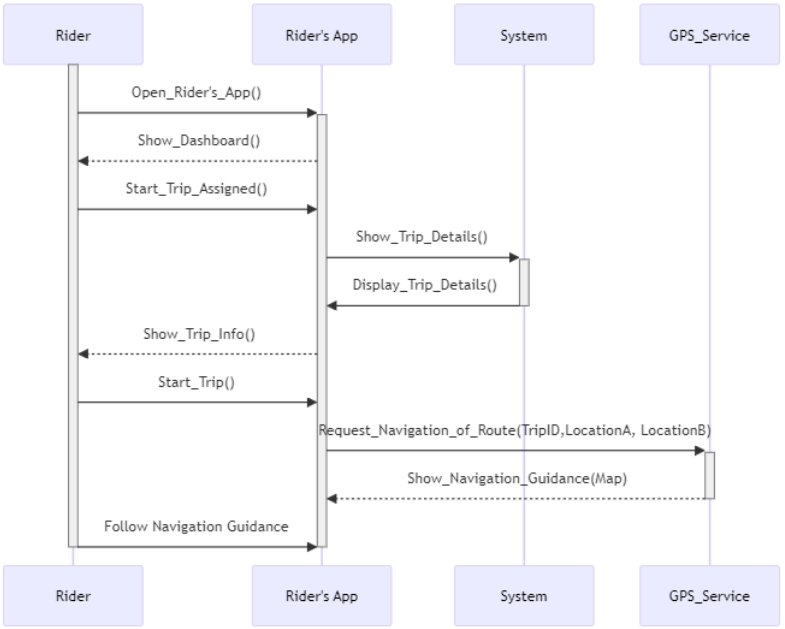
h) Allocate Parcels



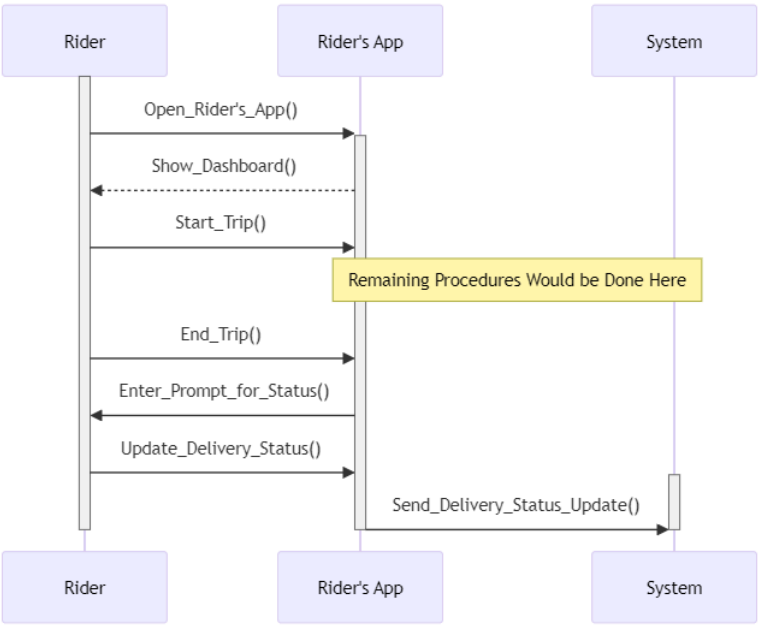
i) View Trip Assignments



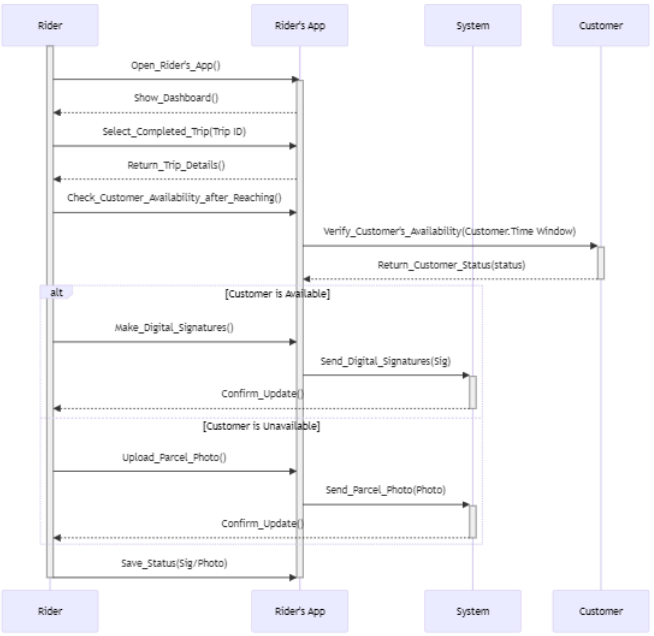
j) Utilize Navigations



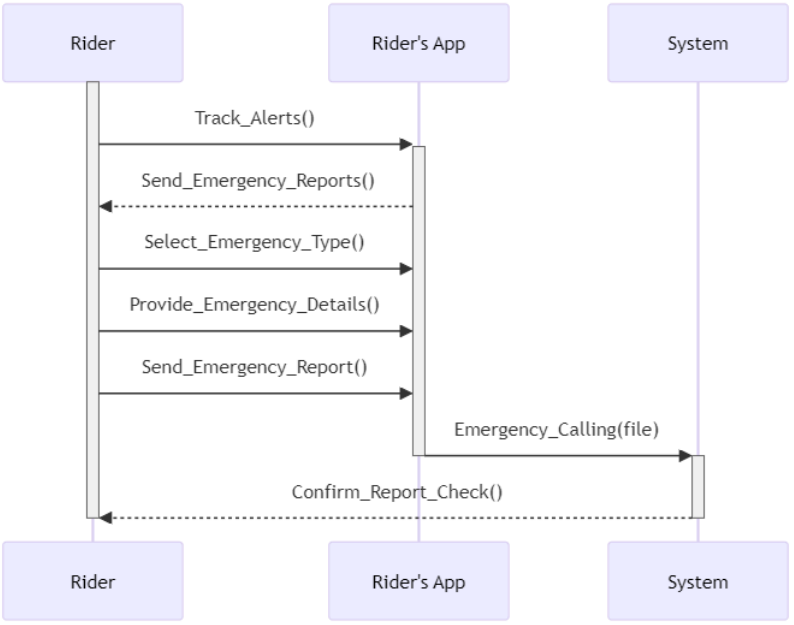
k) Update Delivery Status



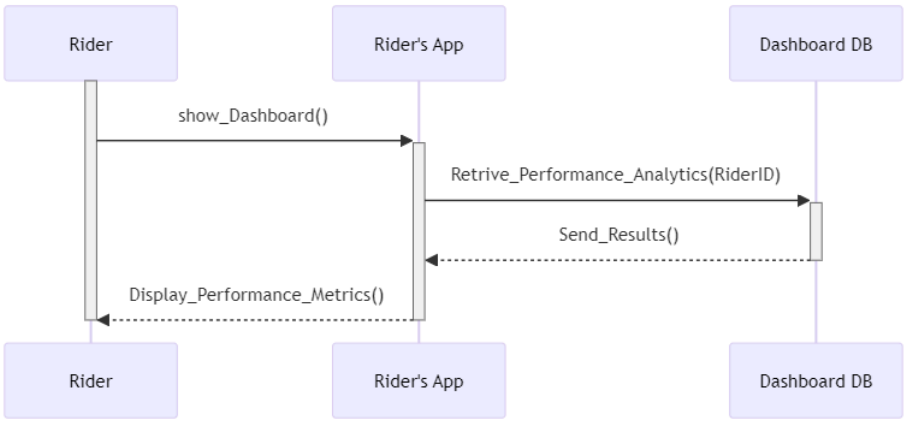
l) Upload Parcel Documentation



m) Report Emergency

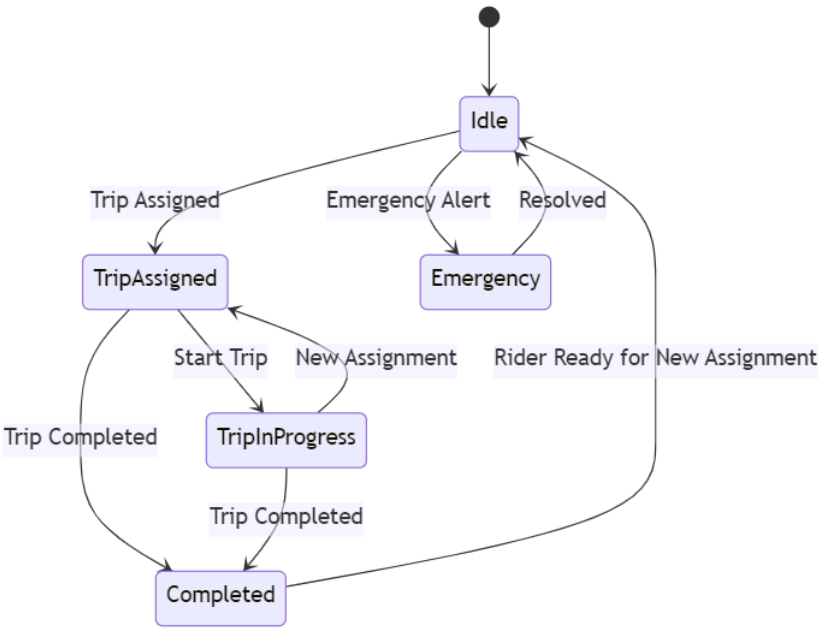


n) Review Personal Performance

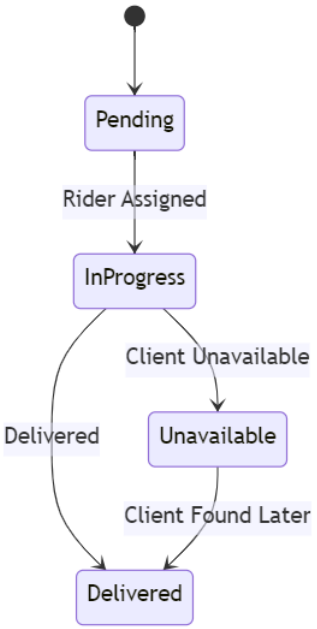


13) State Diagrams

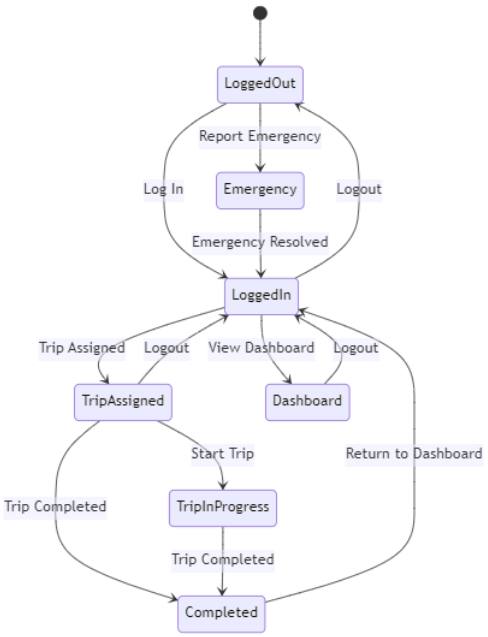
a) Rider Status



b) Parcel Status



c) Mobile App



Development Phase

1) Back-end Development

a) Comments / Functions Description

i) Importance of Comments and Descriptive Function Names

a) Comments and descriptive function names play a vital role in code comprehension and maintainability.

b) We utilized third-party algorithm implementations, but ensured that the code was well-documented.

c) Clear comments helped us understand the code's structure and its alignment with our problem.

ii) Commenting and Function Naming Conventions

a) Followed our commenting conventions to make the code more accessible.

b) Provided comments that clarified the purpose and functionality of various functions, variables, and code sections.

iii) Examples of Well-Commented Code

Python

```
def calculate_distance(point1, point2):
    """
    Calculate the Euclidean distance between two points.
    Args:
        point1: A tuple (x, y) representing the first point.
        point2: A tuple (x, y) representing the second point.
    Returns:
        The Euclidean distance between the two points.
    """
    x1, y1 = point1
    x2, y2 = point2
    distance = ((x1 - x2) ** 2 + (y1 - y2) ** 2) ** 0.5
```

```
    return distance
```

Even though this code was not developed by us, we maintained it with clear comments for improved code readability.

b) Imports and Naming Conventions

i) Significance of Proper Naming Conventions

a) Proper naming conventions are essential for code readability and collaboration.

b) While we did not develop the original code, we ensured the naming conventions adhered to common practices.

ii) Specific Naming Conventions Followed

Maintained consistent naming conventions for variables and functions in the code, ensuring readability and a common coding style.

iii) Choice of Import Statements and Libraries

a) Third-party implementations were chosen for their relevance to our project objectives.

b) Import statements were retained or modified as necessary to ensure compatibility with our project environment.

c) Analysis of Code

During Iteration 1, we faced several challenges while utilizing third-party implementations for algorithm performance evaluation:

i) Understanding Code and Problem Compatibility

a) Evaluating and understanding third-party code was a crucial task.

b) Ensured that the chosen code was capable of addressing our specific problem of the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW).

ii) Incorporation of Constraints

a) Verified that the code incorporated all the constraints relevant to our problem.

b) Addressed any deviations to ensure that the code aligned with our project requirements.

iii) **Code Quality and Reliability**

a) While using GitHub implementations, we encountered variations in code quality and documentation.

b) Carefully reviewed and tested the code to identify and rectify bugs or inefficiencies, ensuring reliable results.

iv) **Compatibility**

a) Modified the third-party code to meet our project's specific environment, including programming language, libraries, and dependencies.

v) **Reproducibility**

a) Reproducibility of results is paramount in academic research.

b) Confirmed that the selected implementations consistently produced results to facilitate further validation by others.

vi) **Algorithm Parameters**

a) Understanding and setting the algorithm parameters correctly was essential.

b) Adjusted the default parameters to suit our VRPTW problem and dataset, ensuring optimal performance.

vii) **Benchmark Dataset Compatibility**

a) Verified that the chosen implementations could handle the SOLOMON benchmark dataset format and constraints.

b) Made necessary modifications and preprocessing to adapt the data to the code's expected format.

viii) **Results Evaluation**

a) Established a rigorous evaluation method, considering criteria such as execution time, minimum distance, and the number of vehicles.

b) This method was designed to align with our research objectives.

ix) **Benchmarking Overhead**

a) Aware of potential overhead introduced by third-party code structure and I/O operations.

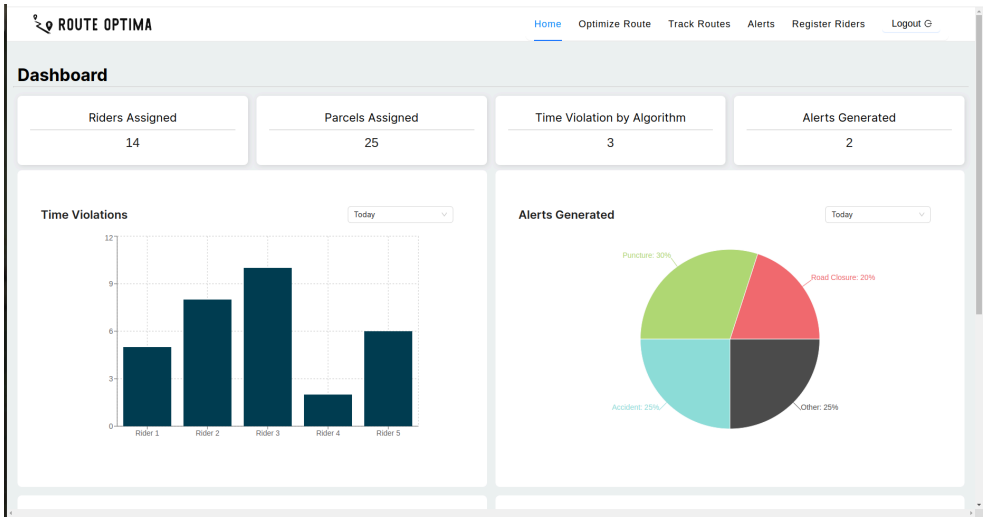
b) This consideration was crucial for interpreting execution times and result accuracy.

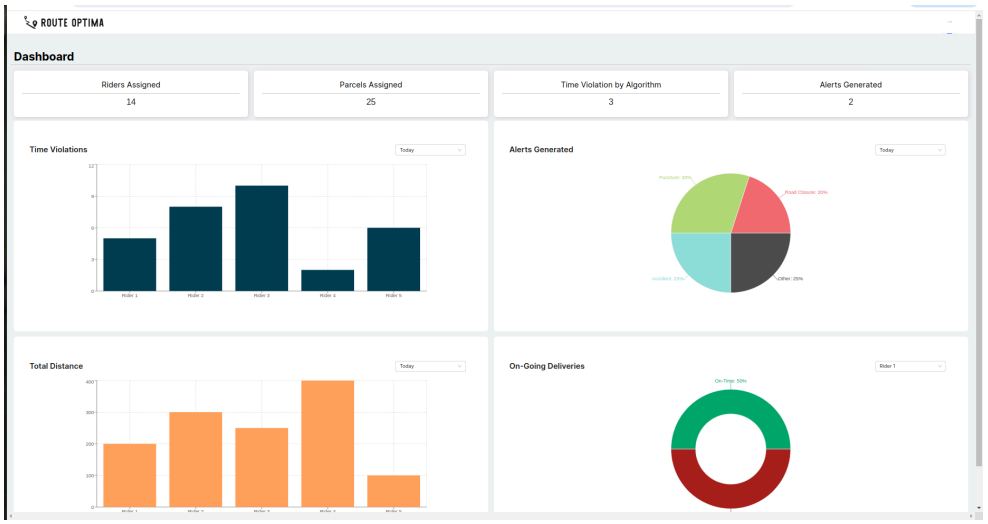
2) Front-end Development

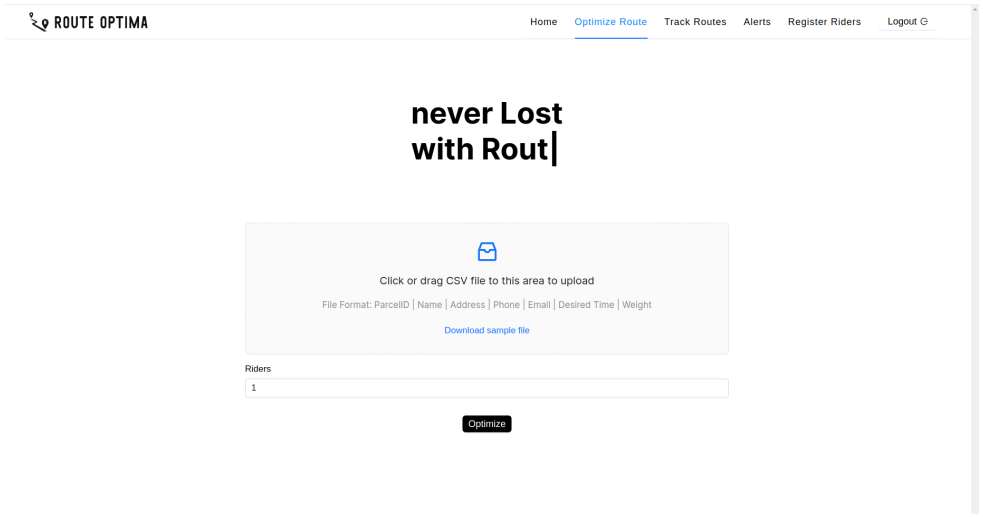
a) Wireframes/ User Interfaces

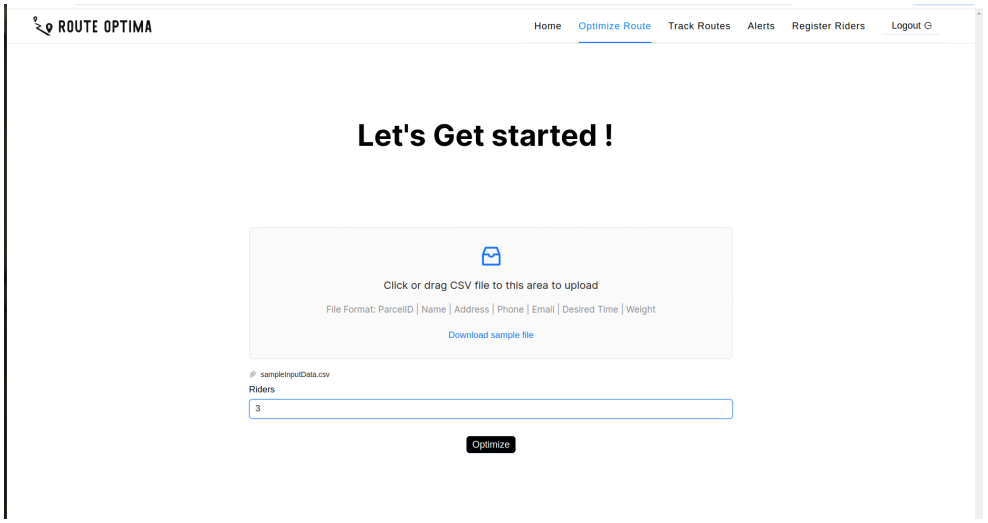
Link to all UIs: [Wireframes of Mobile and Web Portal]

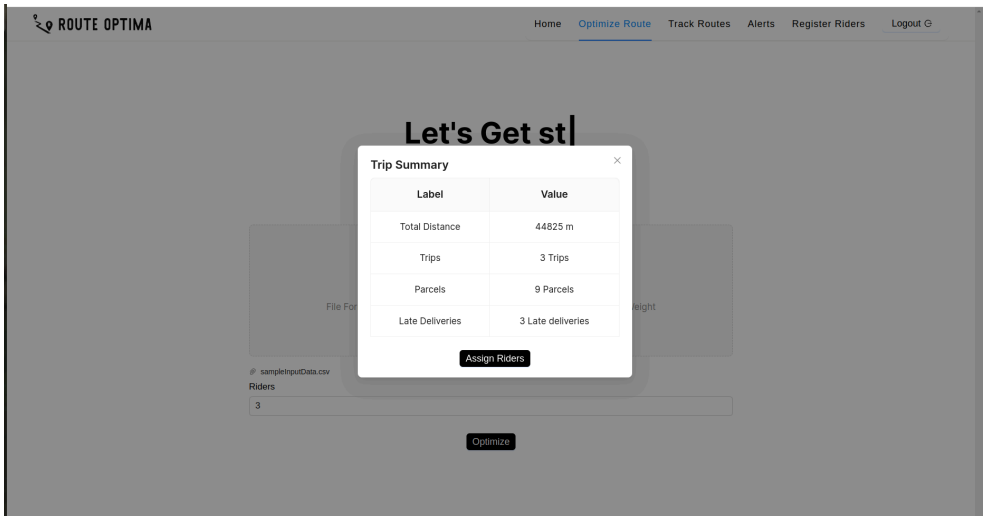
b) Frontend Development

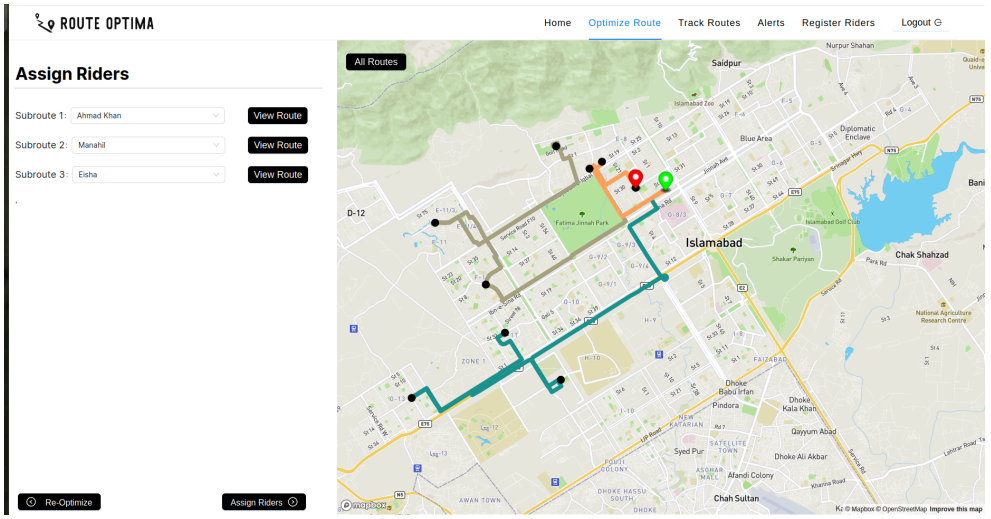


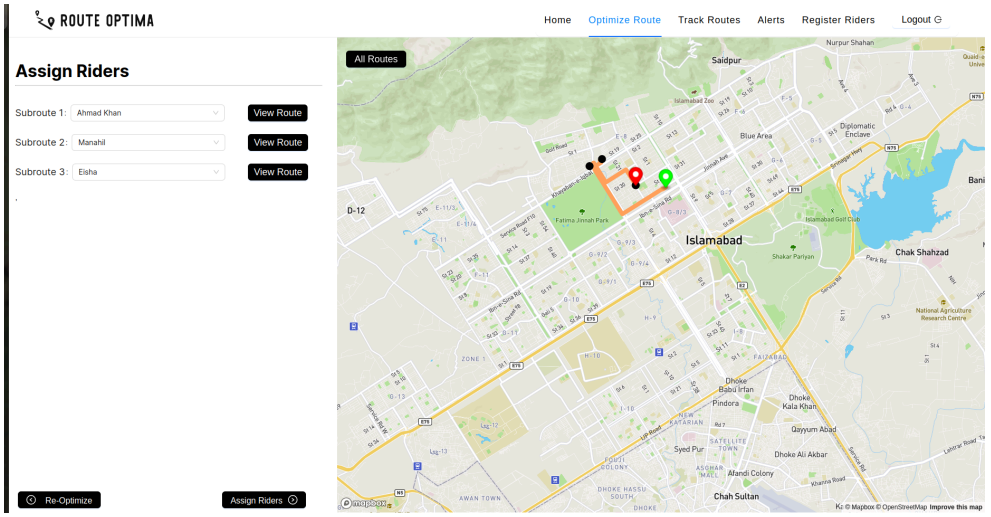


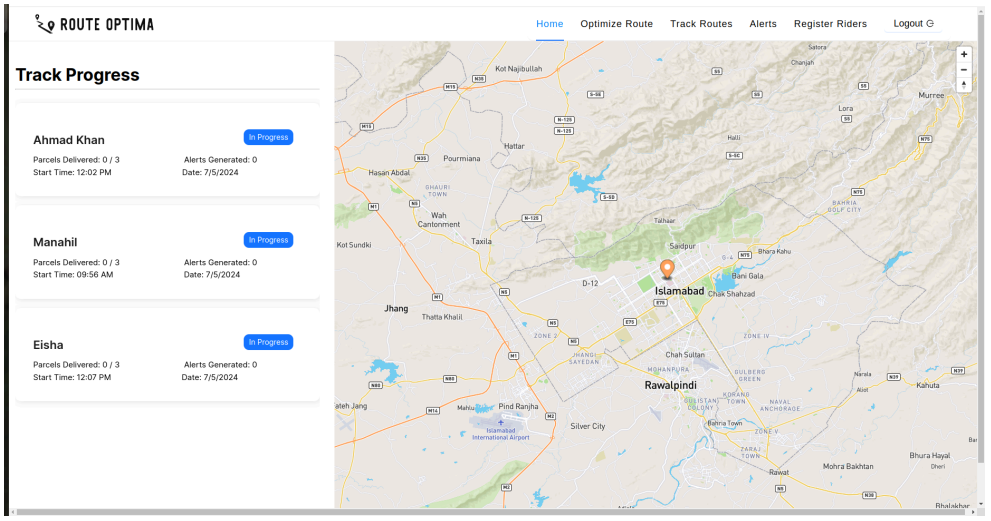

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
never Lost
with Rout
Click or drag CSV file to this area to upload
File Format: ParcelID | Name | Address | Phone | Email | Desired Time | Weight
Download sample file
Riders
1
Optimize

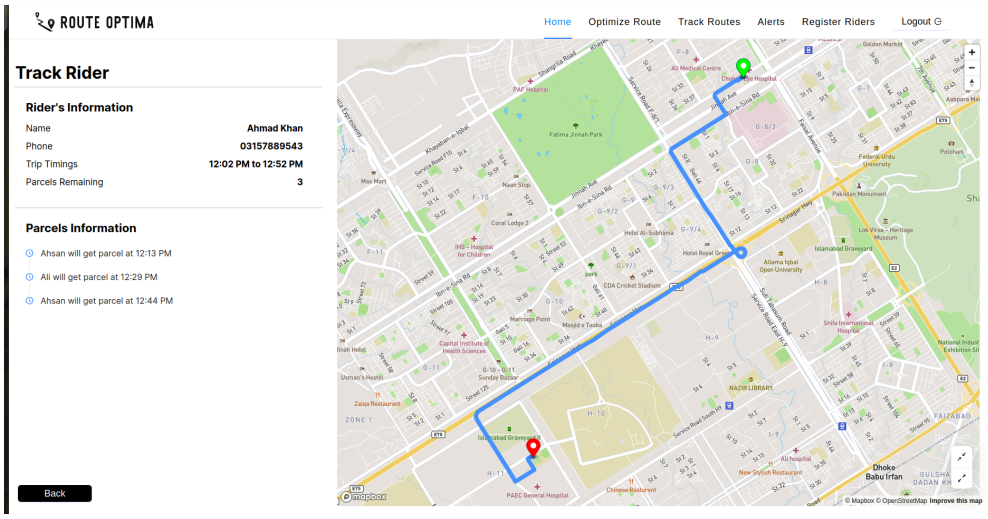

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Let's Get started !
Click or drag CSV file to this area to upload
File Format: ParcelID | Name | Address | Phone | Email | Desired Time | Weight
Download sample file
sampleInputData.csv
Riders
3
Optimize

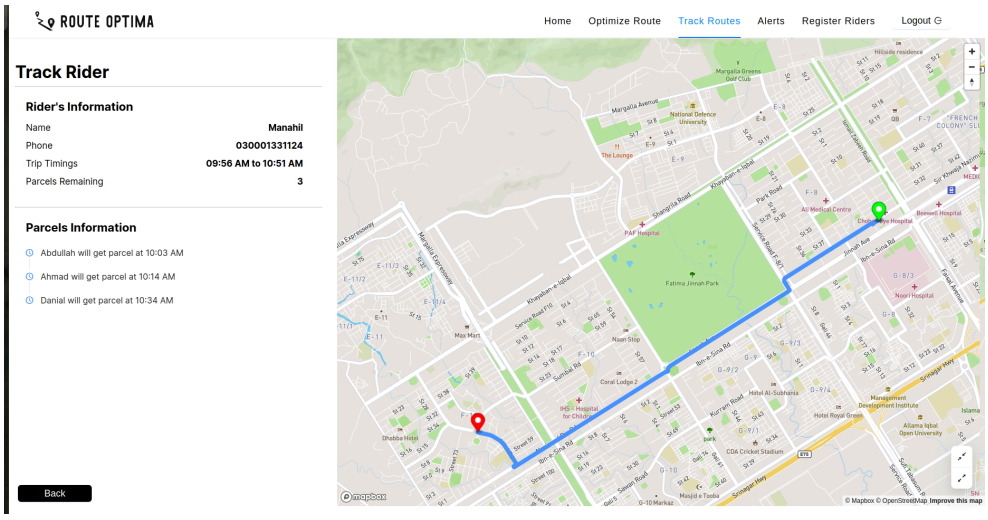

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Let's Get st
Trip Summary
Label
Value
Total Distance
44825 m
Trips
3 Trips
Parcels
9 Parcels
Late Deliveries
3 Late deliveries
Assign Riders
sampleInputData.csv
Riders
3
Optimize

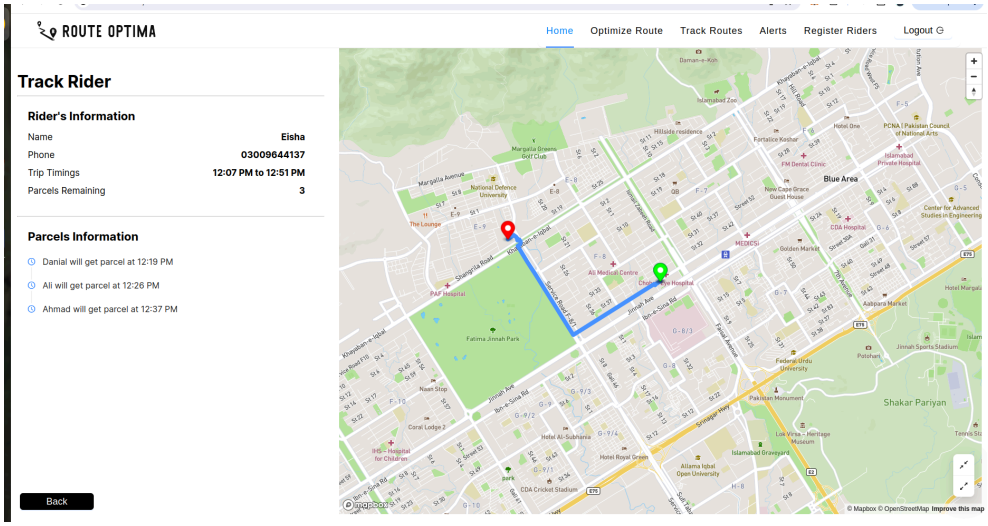

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Assign Riders
Subroute 1: Ahmad Khan
Subroute 2: Manahil
Subroute 3: Eisha
View Route
View Route
View Route
Re-Optimize
Assign Riders
All Routes
Islamabad
Saidpur
Chak Shahzad
Chah Sultan
D-12

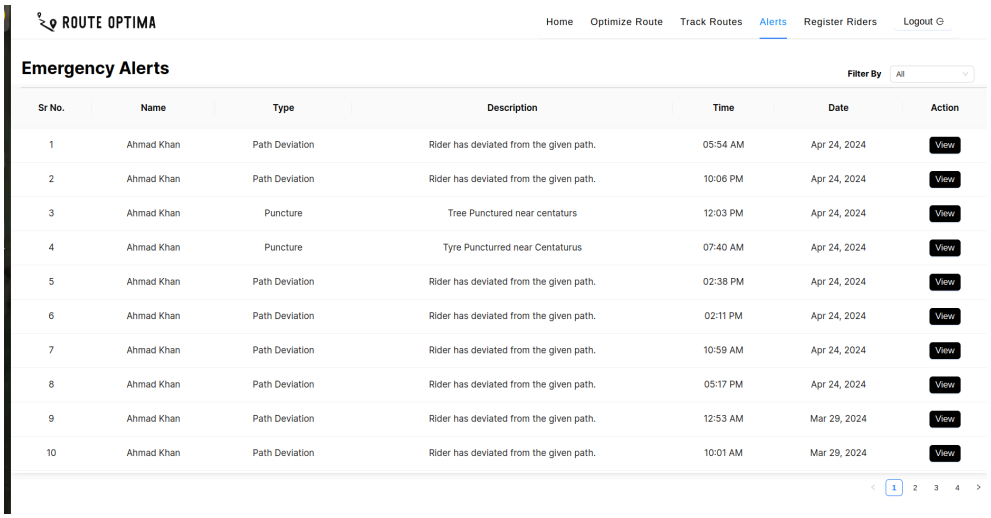

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Assign Riders
Subroute 1: Ahmad Khan
Subroute 2: Manahil
Subroute 3: Eisha
View Route
View Route
View Route
Re-Optimize
Assign Riders
All Routes
Islamabad
Saidpur
Chak Shahzad
Chah Sultan
D-12

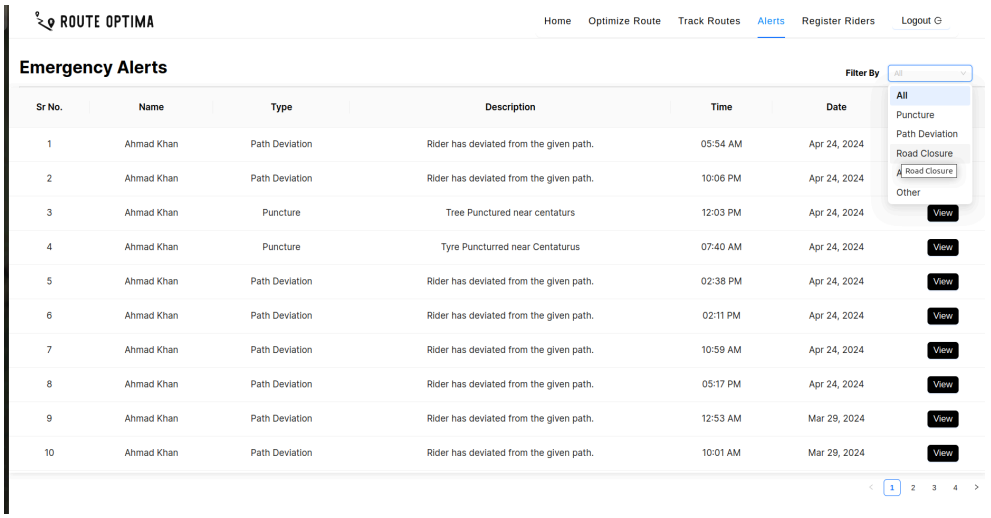

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Track Progress
Ahmad Khan
In Progress
Parcels Delivered: 0 / 3
Start Time: 12:02 PM
Alerts Generated: 0
Date: 7/5/2024
Manahil
In Progress
Parcels Delivered: 0 / 3
Start Time: 09:56 AM
Alerts Generated: 0
Date: 7/5/2024
Eisha
In Progress
Parcels Delivered: 0 / 3
Start Time: 12:07 PM
Alerts Generated: 0
Date: 7/5/2024
Islamabad
Rawalpindi
Chak Shahzad
Saidpur
D-12

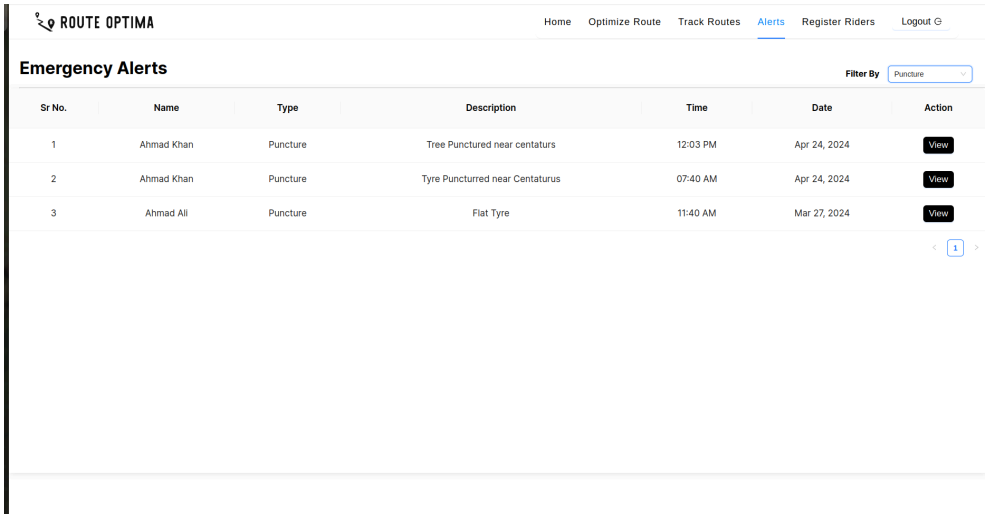

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Track Rider
Rider's Information
Name
Ahmad Khan
Phone
03157889543
Trip Timings
12:02 PM to 12:52 PM
Parcels Remaining
3
Parcels Information
Ahsan will get parcel at 12:13 PM
Ali will get parcel at 12:29 PM
Ahsan will get parcel at 12:44 PM
Back

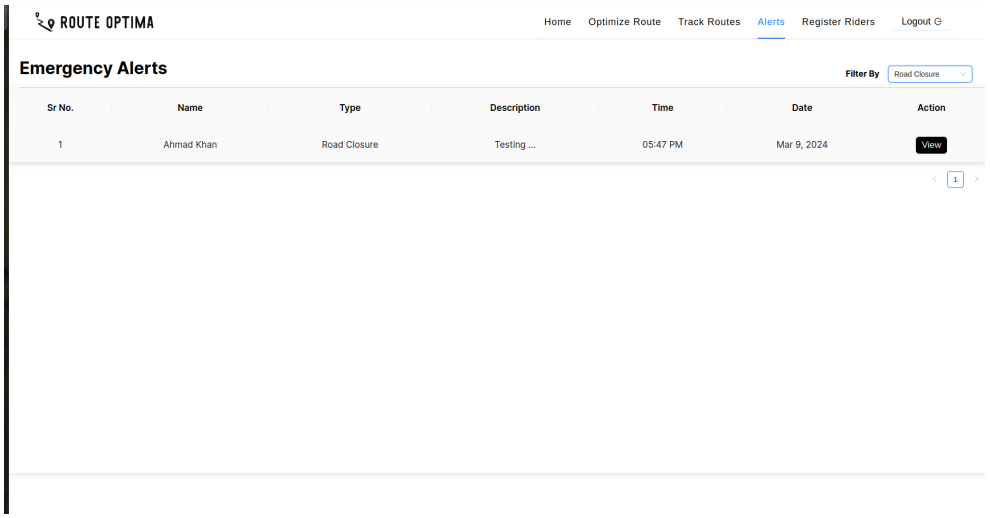

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Track Rider
Rider's Information
Name
Manahil
Phone
030001331124
Trip Timings
09:56 AM to 10:51 AM
Parcels Remaining
3
Parcels Information
Abdullah will get parcel at 10:03 AM
Ahmad will get parcel at 10:14 AM
Danial will get parcel at 10:34 AM
Back

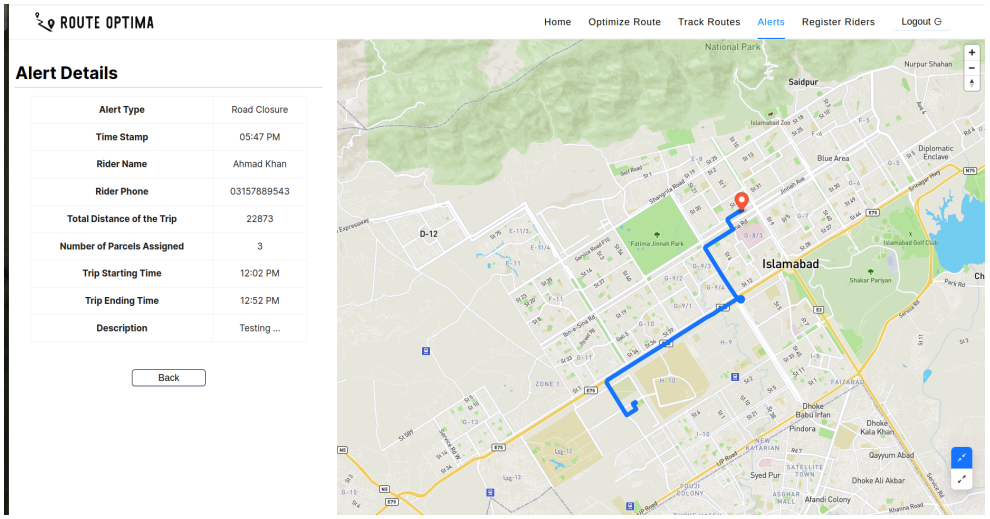

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Track Rider
Rider's Information
Name
Eisha
Phone
03009644137
Trip Timings
12:07 PM to 12:51 PM
Parcels Remaining
3
Parcels Information
Danial will get parcel at 12:19 PM
Ali will get parcel at 12:26 PM
Ahmad will get parcel at 12:37 PM
Back

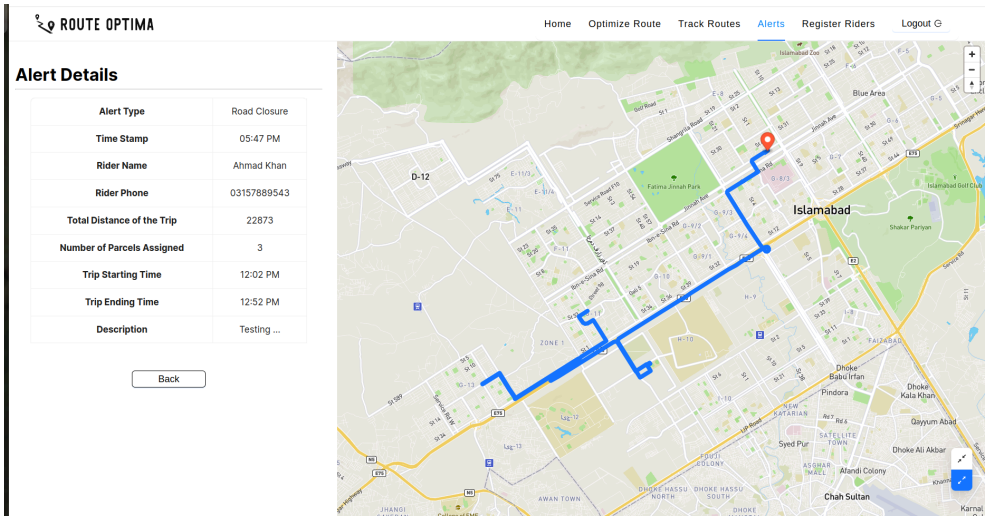

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Emergency Alerts
Filter By
All
Sr No. | Name | Type | Description | Time | Date | Action
1 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 05:54 AM | Apr 24, 2024 | View
2 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:06 PM | Apr 24, 2024 | View
3 | Ahmad Khan | Puncture | Tree Punctured near centaturs | 12:03 PM | Apr 24, 2024 | View
4 | Ahmad Khan | Puncture | Tyre Puncturred near Centaturus | 07:40 AM | Apr 24, 2024 | View
5 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 02:38 PM | Apr 24, 2024 | View
6 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 02:11 PM | Apr 24, 2024 | View
7 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:59 AM | Apr 24, 2024 | View
8 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 05:17 PM | Apr 24, 2024 | View
9 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 12:53 AM | Mar 29, 2024 | View
10 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:01 AM | Mar 29, 2024 | View
1 2 3 4

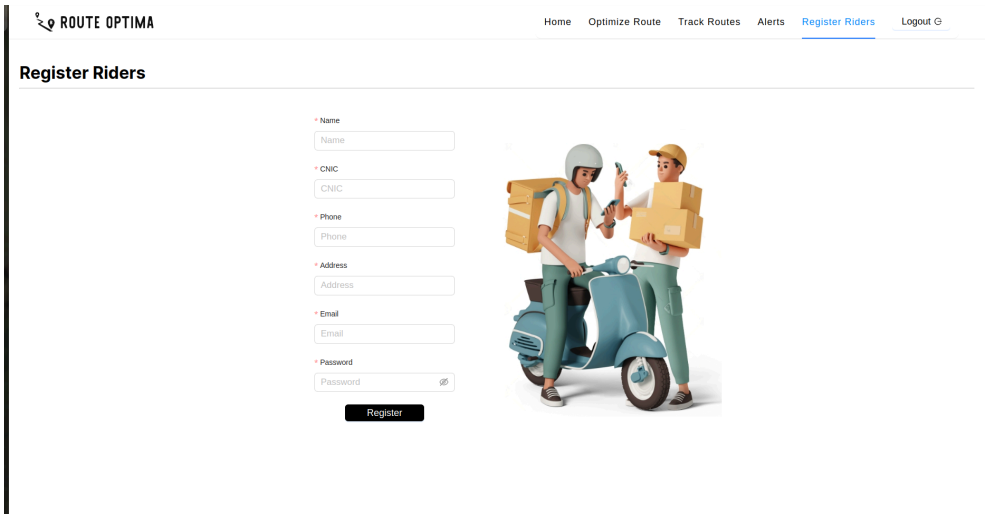

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Emergency Alerts
Filter By
All
All
Puncture
Path Deviation
Road Closure
Road Closure
Other
Sr No. | Name | Type | Description | Time | Date | Action
1 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 05:54 AM | Apr 24, 2024 | View
2 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:06 PM | Apr 24, 2024 | View
3 | Ahmad Khan | Puncture | Tree Punctured near centaturs | 12:03 PM | Apr 24, 2024 | View
4 | Ahmad Khan | Puncture | Tyre Puncturred near Centaturus | 07:40 AM | Apr 24, 2024 | View
5 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 02:38 PM | Apr 24, 2024 | View
6 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 02:11 PM | Apr 24, 2024 | View
7 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:59 AM | Apr 24, 2024 | View
8 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 05:17 PM | Apr 24, 2024 | View
9 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 12:53 AM | Mar 29, 2024 | View
10 | Ahmad Khan | Path Deviation | Rider has deviated from the given path. | 10:01 AM | Mar 29, 2024 | View
1 2 3 4

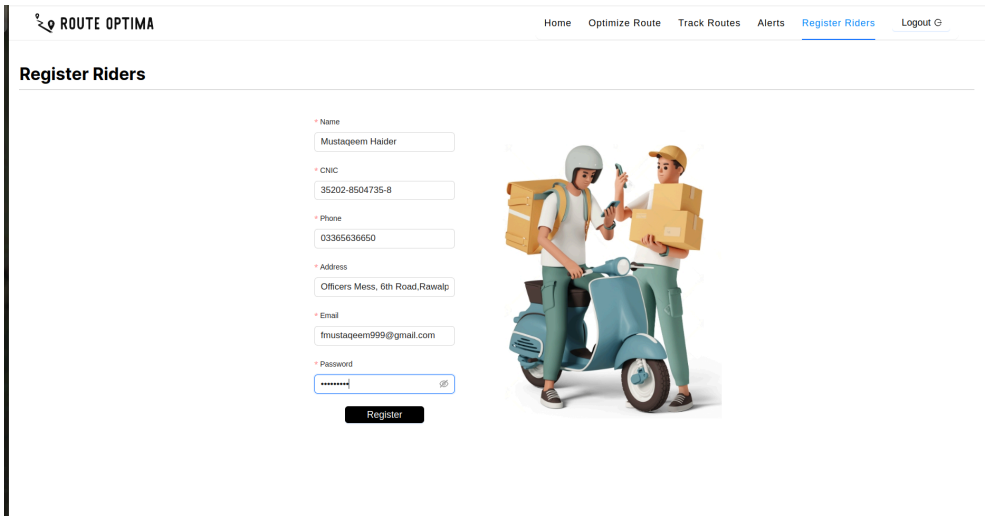

ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Emergency Alerts
Filter By
Puncture
Sr No. | Name | Type | Description | Time | Date | Action
1 | Ahmad Khan | Puncture | Tree Punctured near centaturs | 12:03 PM | Apr 24, 2024 | View
2 | Ahmad Khan | Puncture | Tyre Puncturred near Centaturus | 07:40 AM | Apr 24, 2024 | View
3 | Ahmad Ali | Puncture | Flat Tyre | 11:40 AM | Mar 27, 2024 | View
1


ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Emergency Alerts
Filter By
Road Closure
Sr No.
Name
Type
Description
Time
Date
Action
1
Ahmad Khan
Road Closure
Testing ...
05:47 PM
Mar 9, 2024
View
1


ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Alert Details
Alert Type
Road Closure
Time Stamp
05:47 PM
Rider Name
Ahmad Khan
Rider Phone
03157889543
Total Distance of the Trip
22873
Number of Parcels Assigned
3
Trip Starting Time
12:02 PM
Trip Ending Time
12:52 PM
Description
Testing ...
Back
National Park
Saidpur
Nurpur Shahan
Blue Area
D-12
Islamabad
Dhoke Babu Irfan
Pindora
Dhoke Kala Khan
Qayyum Abad
Syed Pur
Dhoke Ali Akbar
Afandi Colony


ROUTE OPTIMA
Home
Optimize Route
Track Routes
Alerts
Register Riders
Logout
Alert Details
Alert Type
Road Closure
Time Stamp
05:47 PM
Rider Name
Ahmad Khan
Rider Phone
03157889543
Total Distance of the Trip
22873
Number of Parcels Assigned
3
Trip Starting Time
12:02 PM
Trip Ending Time
12:52 PM
Description
Testing ...
Back
Blue Area
D-12
Islamabad
Dhoke Babu Irfan
Pindora
Dhoke Kala Khan
Qayyum Abad
Syed Pur
Dhoke Ali Akbar
Afandi Colony
Chah Sultan





Testing Phase

1) Test Plan

a) Overall Approach to Testing

Purpose: Ensure the robustness of third-party algorithm implementations by comparing different metaheuristic algorithms (ACO and NSGA) on the SOLOMON benchmark dataset.

Approach: Conduct thorough testing to verify whether these algorithms accurately considered critical constraints, such as weight, number of vehicles, and time windows.

b) Types of Testing Conducted

i) **Unit Testing:** Verify the correctness of the algorithms and uncover how they handled constraints.

ii) **Functional Testing:** Ensure that the algorithms executed as expected, producing valid solutions for different datasets and constraints.

iii) **Performance Testing:** Evaluate the algorithms' performance in terms of execution time, minimum distance of solutions, and the number of vehicles required.

iv) **Compatibility Testing**: Ensure that results produced by different authors' implementations were consistent and allowed for meaningful comparisons.

c) Testing Tools and Frameworks

Unit Testing: Python's unittest framework.

2) Test Cases

a) Feasible Time Windows and Weight Constraints:

- **Purpose:** Verify that the algorithm accurately handles datasets with feasible time windows and weight constraints.
- **Steps:**
    1. Input datasets with feasible constraints.
    2. Assess the algorithm's solutions.
- **Expected Result:** Valid solutions reflecting adherence to constraints.

b) Infeasible Time Windows and Weight Constraints:

- **Purpose:** Test how the algorithm responds when faced with datasets with infeasible/conflicting time windows and weight constraints.
- **Steps:**
    1. Input datasets with infeasible constraints.
    2. Assess the algorithm's response.
- **Expected Result:** Observing how the algorithm addresses the infeasible constraints (e.g., potentially adding more vehicles).

c) Tight Time Windows:

- **Purpose:** Assess how the algorithm handles datasets with tight time windows.
- **Steps:**
    1. Input datasets with tight constraints.
    2. Evaluate the algorithm's solutions.
- **Expected Result:** Solutions that adhere to the tight constraints.

d) Variable Weight Constraints:

e) **Purpose:** Test the algorithm's performance under different weight constraints.

f) Steps:

a) Input datasets with varying weight constraints.

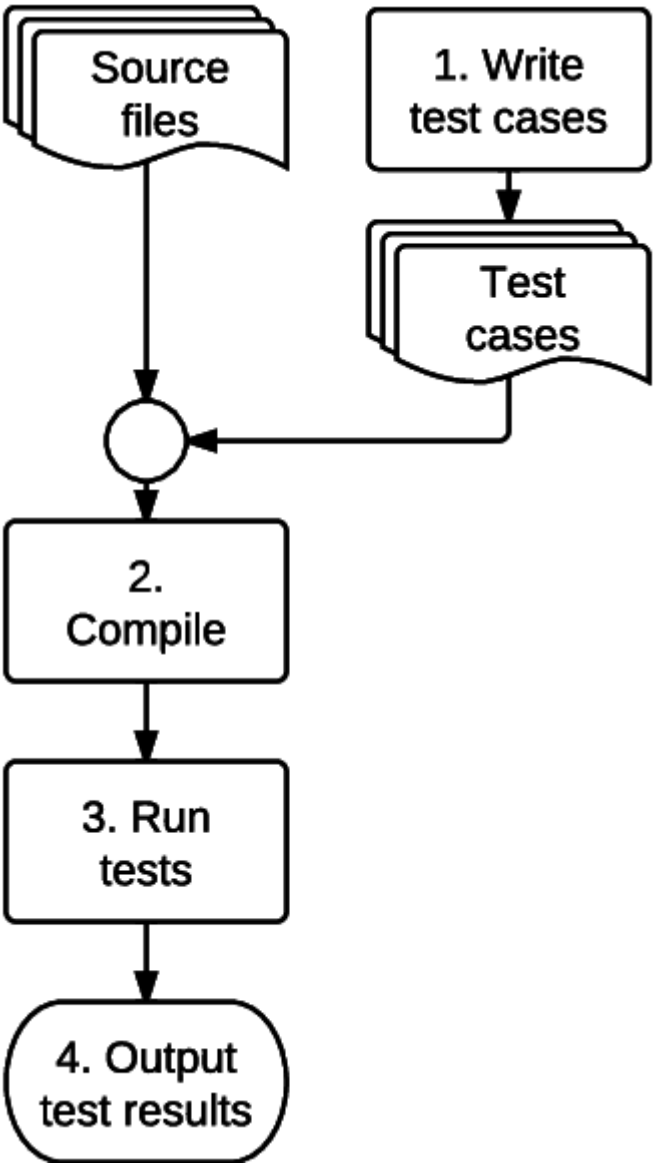
b) Analyze the algorithm's behavior.

g) **Expected Result:** Evaluating how the algorithm optimizes routes when dealing with varying weight constraints.

3) Unit Testing

Unit testing was a pivotal component of the testing phase, revealing and rectifying several bugs and discrepancies.

Unit Testing Flowchart:



Unit Testing Results:

During unit testing, the following example bug was encountered and addressed:

CODE SNIPPET:

```
if self.target_time_limits[0][i] > k_time:  # The
rider reached earlier than the time-window
specified

    k_time = self.target_time_limits[0][i]       #
Since the time window is not yet active, the rider
will have to wait

    penalty += PE * (self.target_time_limits[0][i]
- k_time) # Add penalty cost for arriving earlier
according to minutes left for time window to be
open
```

In the code snippet above, the penalty is added after the rider's waiting time is set. This logic error results in penalties not being applied to riders who arrive earlier than the time window.

The corrected code snippet is as follows:

CODE SNIPPET:

```
if self.target_time_limits[0][i] > k_time:  # The
rider reached earlier than the time-window
specified

     penalty += PE *
(self.target_time_limits[0][i] - k_time) # Add
penalty cost for arriving earlier according to
minutes left for the time window to be open

     k_time = self.target_time_limits[0][i]
# Set the waiting time to the opening of the time
window
```

In the corrected code snippet, the penalty is added before the waiting time is set. This ensures that penalties are appropriately applied to riders who arrive earlier than the specified time window. The waiting time is set after applying the penalty cost, as it should be in the logical flow.

Impact of Unit Testing:

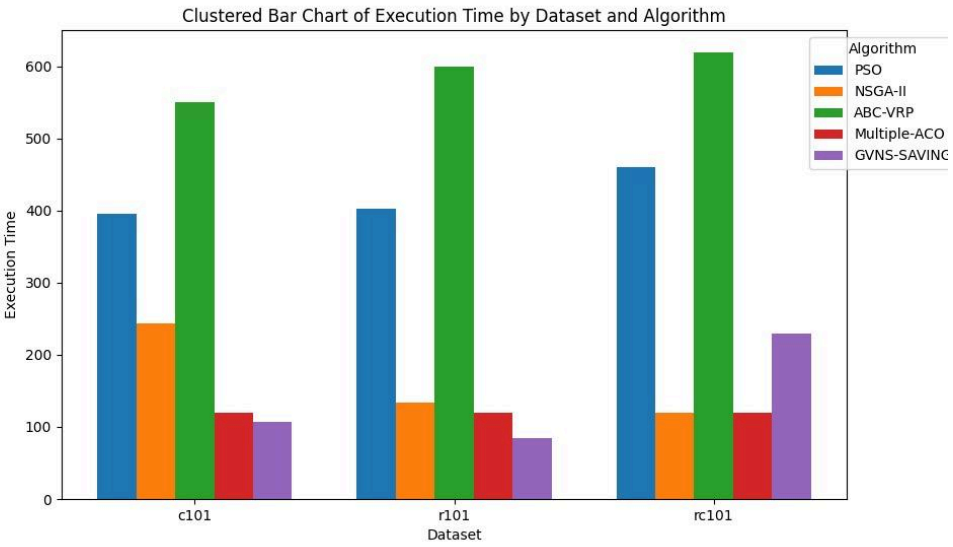
Unit testing played a pivotal role in enhancing the reliability and correctness of the selected algorithms:

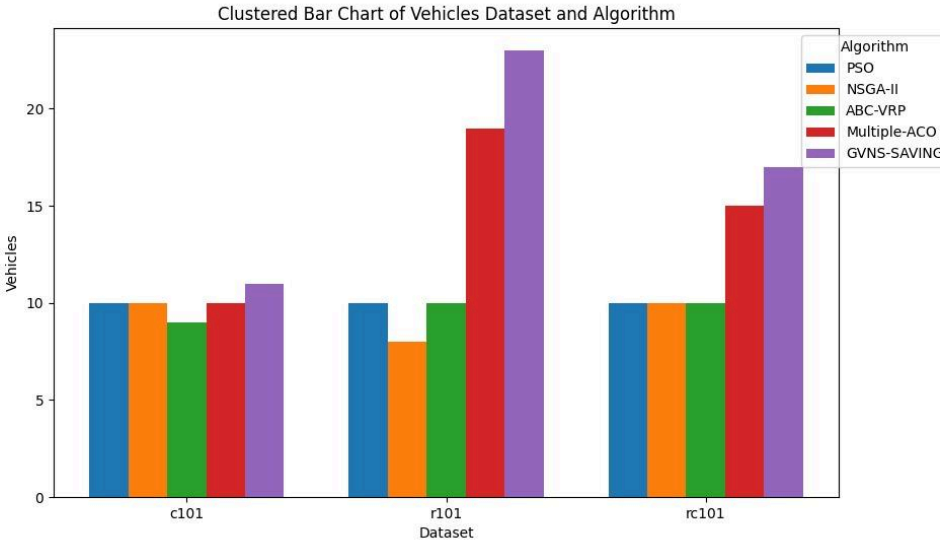
- **Bug Identification and Resolution:** The systematic approach of unit testing facilitated the identification and rectification of logical errors, improving the correctness of the code.
- **Enhanced Functionality:** Correcting the identified bug in the penalty calculation logic improved the functionality and accuracy of the algorithm.
- **Quality Assurance:** Unit testing helped ensure that the algorithms produced consistent and reliable results, enhancing the overall quality and trustworthiness of the codebase.

Algorithm Results' Comparison:

a) Heuristic Results' Comparison:

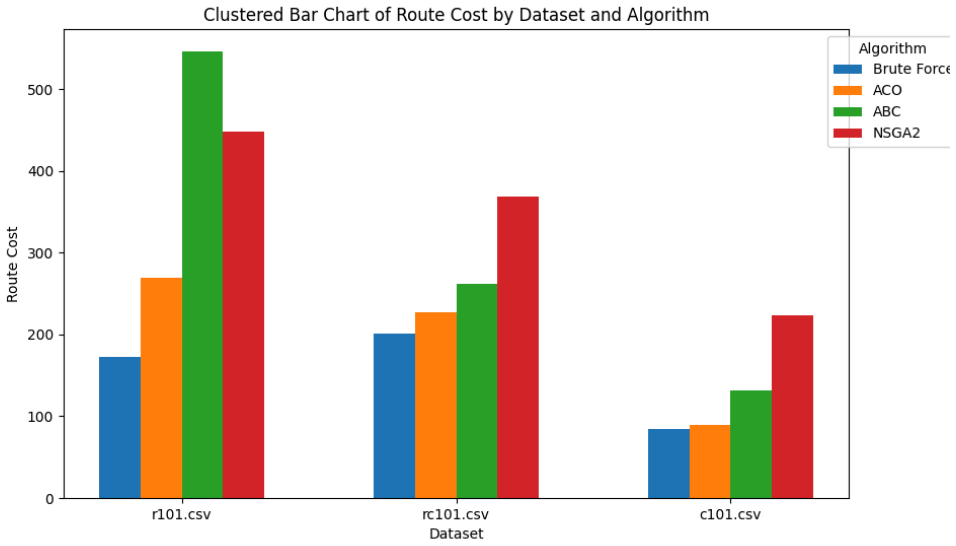
Following are the results obtained from evaluation of different heuristic algorithm





b) Exact-Heuristic Results' Comparison:

Following is the comparison of results obtained from exact algorithm and chosen heuristic algorithms:



Conclusion

In conclusion, this report comprehensively outlines the Software Requirements Specification, Analysis and Design Artifacts, Development Phase, and Testing Phase of our project. It provides a clear understanding of the features, functional and non-functional requirements, data and functional perspectives, development practices, and testing strategies.

This document serves as a valuable reference for the project's stakeholders, offering insights into the design, development, and testing processes. It emphasizes the importance of well-commented code, naming conventions, and thorough testing, ensuring code quality, reliability, and compatibility. By following the guidelines and best practices outlined in this report, it is our hope that this report will be a helpful resource throughout the project's lifecycle, aiding in its successful completion.

References

1. A Comparison Between Ant System and Max-Min Ant System in Multi Colonial Systems
2. Ant colony optimization - Scholarpedia
3. Shi, Y., Lin, Y., Li, B., & Li, R. Y. M. (2022). A bi-objective optimization model for the medical supplies' simultaneous pickup and delivery with drones. Computers & Industrial Engineering, 171, 108389.
4. Wang, M., Ma, T., Li, G., Zhai, X., & Qiao, S. (2020). Ant colony optimization with an improved pheromone model for solving MTSP with capacity and time window constraint. IEEE Access, 8, 106872-106879.
5. GitHub - rwuilbercq/Hive: Artificial Bee Colony Algorithm in Python.
6. GitHub - krishna-praveen/Capacitated-Vehicle-Routing-Problem: Capacitated vehicle routing problem implemented in python using DEAP package. Non dominated sorting Genetic algorithm is used to solve Multiobjective problem of minimizing Total distance travelled by all vehicles and minimizing total number of vehicles at same time.
7. GitHub - jonzhaocn/VRPTW-ACO-python: A python implementation of a ant colony optimization based solution to Vehicle Routing Problem with Time Windows.
8. GitHub - dungtran209/Modelling-and-Analysis-of-a-Vehicle-Routing-Problem-with-Time-Windows-in-Freight-Delivery
9. Using improved PSO(Particle Swarm Optimization) algorithm resolve VRPTW question.
10. Benchmarking Problems
11. FYP (Final Year Project) Github Repository