RSET
RAJAGIRI SCHOOL OF
ENGINEERING & TECHNOLOGY
(AUTONOMOUS)

Project report on

Automated Bus Scheduling and Route Management System for Delhi Transport Corporation

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

By

Elviin Thomas Eldho (U2103085)
Fabin Chandi (U2103088)
Gayathri Bijoy (U2103093)
Kuruvilla Jacob (U2103127)

Under the guidance of

Dr. Sminu Izudheen

Department of Computer Science and Engineering
Rajagiri School of Engineering & Technology (Autonomous)
(Parent University: APJ Abdul Kalam Technological University)
Rajagiri Valley, Kakkanad, Kochi, 682039
November 2024

CERTIFICATE

This is to certify that the project report entitled ***"Automated Bus Scheduling and Route Management System for Delhi Transport Corporation"*** *is a bonafide record of the work done by* ***Elviin Thomas Eldho (U2103085)***, ***Fabin Chandi (U2103088)***, ***Kuruvilla Jacob (U2103127)***,***Gayathri Bijoy (U2103093)***, *submitted to the Rajagiri School of Engineering & Technology (RSET) (Autonomous) in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (B. Tech.) in Computer Science and Engineering during the academic year 2024-2025.*

Dr. Sminu Izudheen
Seminar Guide
Associate Professor
Dept. of CSE
RSET

Ms. Anu Maria Joykutty
Project Coordinator
Associate Professor
Dept. of CSE
RSET

Dr. Preetha K G
Professor & HOD
Dept. of CSE
RSET

ACKNOWLEDGMENT

We wish to express our sincere gratitude towards **Rev. Dr. Jaison Paul Mulerikkal CMI**, Principal of RSET, and **Dr Preetha K G**, Head of the Department of Computer Science and Engineering for providing me with the opportunity to undertake our project, "Automated Bus Scheduling and Route Management System for Delhi Transport Corporation".

We are highly indebted to our project coordinators, **Dr. Jisha G.**, Associate Professor, Department of Computer Science and Engineering, and **Ms. Anu Maria Joykutty**, Associate Professor, Department of Computer Science and Engineering, for their valuable support.

It is indeed our pleasure and a moment of satisfaction for us to express our sincere gratitude to our project guide **Dr. Sminu Izudheen** for her patience and all the priceless advice and wisdom she has shared with us.

Last but not the least, We would like to express our sincere gratitude towards all other teachers and friends for their continuous support and constructive ideas.

Elviin Thomas Eldho

Fabin Chandi

Gayathri Bijoy

Kuruvilla Jacob

Abstract

The Delhi Transport Corporation (DTC) faces ongoing challenges in bus scheduling and route planning due to manual processes, leading to scheduling conflicts, route overlaps, and unreliable services. This project, the Automated Bus Scheduling and Route Management System, aims to streamline these processes by automating scheduling, optimizing routes, and integrating real-time data for improved operational efficiency.

The system is built using a MERN (MongoDB, Express.js, React.js, Node.js) stack in a simple client-server architecture. MongoDB is used for managing schedules, routes, and transit data, while Express.js and Node.js handle backend operations and API interactions. Instead of WebSockets, the system relies on the Open Transit Data Delhi API to fetch real-time bus locations and schedule updates. A Genetic Algorithm is employed to optimize bus and driver scheduling, ensuring efficient allocation of resources while minimizing scheduling conflicts and idle time. The React.js frontend provides an intuitive interface for transit planners and commuters, offering real-time updates and optimized scheduling insights.

By integrating real-time data and intelligent scheduling algorithms, this system enhances service reliability, reduces operational inefficiencies, and improves the overall commuter experience. Its scalable design allows for future enhancements, contributing to a more responsive and sustainable urban transit infrastructure in Delhi.

Contents

List of Abbreviations

DTC - Delhi Transport Corporation

GIS - Geographic Information System

GPS - Geographical Positioning System

RFID - Radio-Frequency IDentification

GA - Genetic Algorithm

VNS - Variable Neighborhood Search

API - Application Programming Interface

DFD - Data Flow Diagram

GTFS - General Transit Feed Specification

CSV - Comma Separated Values

SSD - Solid State Drive

GPU - Graphics Processing Unit

UI - User Interface

List of Figures

Chapter 1

Introduction

For the last couple of years, there's been a need for effective, trusted, real-time transportation management solutions. Buses and in particular public transportation have issues of congestion, delays and inefficient schedules resulting in decreased commuter satisfaction and overall efficiency. This will require technology solutions to solve these problems and modernize and simplify public transportation systems so they are more convenient and efficient. In this project report, we present a turnkey solution to manage bus schedules, location, and route optimization based on data analytics, GIS, and automated scheduling algorithms.

1.1 Background

This project has its origins in the growing need to make public transport more efficient and reliable. Bus networks are the mainstay of public transport, and in cities they're often under a lot of stress. They include long delays, high commuting rates, and frequent switching of routes due to traffic or other issues. Traditional methods of governing bus time and routes don't allow for enough flexibility and responsiveness to cope with changing situations, leading to poor service quality.

Data analytics, evolutionary algorithms and global information system (GIS) have the potential to transform public transport. With real-time data, transportation departments can dynamically adjust timetables, keep track of vehicle positions, and inform drivers about delays or schedule changes. This project aims to leverage these technologies to build a scalable automated bus scheduling and route management solution that improves service reliability and performance. This not only benefits transit operators but also helps everyday commuters by enabling a sustainable and flexible urban transit network.

1.2 Problem Definition

The goal of this project is to create an automated bus scheduling and real-time monitoring system for public transportation systems. The challenge, in particular, would be the establishment of a reliable and responsive system to dynamically plan bus schedules, optimize drivers, and offer real-time updates for increased service reliability and customer happiness.

1.3 Scope and Motivation

It involves the design of a combined platform to monitor buses in real-time, automate the scheduling, and optimize the routes. The platform will leverage GPS data, data analytics and web-based interfaces to bring seamlessness to transit operators and passengers. It can handle large data, manage complex scheduling, and provide an easy to use interface for dealing with real-time transit data. It will be a scalable solution, and it can be deployed for different transit networks and environments.

The idea behind this project was inspired by the common challenges encountered in public transportation systems in cities. Buses are prone to delays due to traffic, bus schedules and poor planning. These issues are not only undermining the service, but also driving up commuter frustration. With an infrastructure that tracks and adjusts in real time, this will enhance the reliability and productivity of public transportation while minimizing waiting times and improving service. This move is also driven by the potential to decrease the cost of running transit agencies and encourage public transit as an environmentally responsible alternative to private vehicles.

1.4 Objectives

This project is focused on establishing a unified solution for improving the performance and security of public transportation by tracking it in real-time, automated scheduling, and making smart decisions based on data. Using GPS, data analysis, and a user-friendly interface, the system aims to bring insight for transit providers and a better experience for commuters. The project objectives are as follows:

- Develop a real-time tracking system for monitoring bus locations using GPS and

data analytics, ensuring accurate location updates.

- Automate bus scheduling and route planning to optimize resource allocation and minimize delays.
- Design a user-friendly interface that allows operators to easily access and interact with real-time transit data.
- Analyze historical and real-time data to identify patterns and support data-driven planning and operational decisions.
- Implement a scalable solution that can be adapted for various transit systems and networks, supporting future expansions.

1.5 Challenges

The Delhi public bus network is vast, consisting of a large number of bus stops, routes, and trips, making efficient scheduling and real-time management highly complex. This project faces challenges in processing massive amounts of real-time data from multiple sources while ensuring seamless integration into a simple and accessible dashboard. Tracking bus locations with minimal latency across such a large network further adds to the complexity. Additionally, data privacy concerns and technical limitations, such as network connectivity issues, must be carefully addressed to maintain reliability and security.

1.6 Assumptions

- All buses in the transit network are equipped with GPS tracking devices capable of providing real-time location data.
- The transportation network has a stable infrastructure, with occasional variations in schedules due to traffic or maintenance.
- Transit operators and users have access to mobile devices or computers for interacting with the system.

1.7 Societal / Industrial Relevance

It is an ambitious project, with enormous importance for society and transportation. Socially, it improves the access and continuity of public transport, which makes it more appealing to commuters and less reliant on private cars. For transportation companies, it provides the platform to efficiently manage fleet, save costs, and better use resources. This type of infrastructure is vital to addressing the ever-rising demand for urban public transport, especially in rapidly developing cities, and also facilitates urban mobility that is sustainable.

1.8 Organization of the Report

This report is structured as follows:

This chapter has provided an overview of the project background, objectives, scope, and significance. The subsequent chapters explore the technical aspects in detail, covering the system's design, implementation, and evaluation.

Chapter 2 presents a comprehensive literature review, analyzing existing solutions, identifying their limitations, and highlighting the gaps that this project seeks to address.

Chapter 3 details the system design, covering the architecture, component design, algorithms, data flow diagrams, tools and technologies used, dataset, module division, key deliverables, and Gantt chart.

Chapter 4 discusses the results and findings, analyzing the system's performance through testing and validation, and evaluating its effectiveness and accuracy.

Chapter 5 concludes the report by summarizing key insights, discussing limitations, and outlining potential future enhancements to improve the system.

Chapter 2

Literature Survey

This chapter reviews the latest research findings related to real-time public transit information systems, GIS and GPS route optimization, passenger traffic pattern analysis and multimodal transportation network optimization. These papers provide an overview of the strengths and weaknesses of each strategy, and they serve as a starting point for defining gaps in current research.

2.1 Literature Review

2.1.1 MOBANA: Public Transit Real-Time Information Framework

MOBANA [1] is a real-time public transit information platform, leveraging various data sources such as social media to identify event triggers to foresee delays or interruptions in real time. It is an scalable system that supports huge amounts of real-time data, an important requirement for large transit systems. The key features of MOBANA include fast real-time data processing and scalability. But integration with diverse data sources can be challenging, and a distributed infrastructure to handle streams of data and provide system stability is essential.

2.1.2 Research on Logistics Route Optimization Based on GPS and GIS Technology

This study focuses on optimizing logistics [2] routes by leveraging GPS and GIS technologies, combined with RFID-enabled smart machines for real-time tracking. The approach demonstrates significant improvements in delivery efficiency and cost reduction, making it ideal for logistics applications. The primary advantage of this system is its ability to optimize resource allocation, reduce delivery times, and lower operational costs. However, the system heavily depends on the accuracy of GPS and GIS data, which may not be con-

sistent in densely populated areas or regions with poor satellite coverage. Additionally, the implementation costs associated with RFID and GIS infrastructure can be a barrier for some organizations.

2.1.3 Data-Driven Approach for Passenger Mobility Pattern Recognition Using Spatiotemporal Embedding

The data-driven method used in this work is to map passenger movement in urban transit networks using spatiotemporal embeddings [3]. The algorithm can identify habitual movements and discover spatial and temporal information that can guide the optimization of routes and scheduling. Its key feature is that it can predict bus frequencies and adjust schedules according to how the passenger moves. However, the technique requires high-powered computation, which is expensive, particularly for large datasets. A second drawback is that it handles only spatiotemporal information without real-time analytics, which may prevent it from responding well to short-term transit requirements.

2.1.4 Memetic Algorithm for Computing Shortest Paths in Multimodal Transportation Networks

This paper utilizes a hybrid Genetic Algorithm (GA) and Variable Neighborhood Search (VNS) to compute shortest path in multimodal transportation systems [4] as a potential substitute to Dijkstra and other similar algorithms. This memetic algorithm performs large-scale, real-time routing planning efficiently, providing nearly optimal solutions in short order. This hybrid approach has the benefit of being scaleable and giving us concrete solutions that are less out of the optimal way than standalone algorithms. However, the performance of the hybrid algorithm depends strongly on the parameter tuning and no default settings exist. In addition, although it saves time compared to classic algorithms, the hybrid method is computationally intensive on large networks or with large populations.

2.1.5 Opportunistic Sensing for Crowd Detection in Public Transportation Systems

This study leverages smartphone-based opportunistic sensing to assess bus crowdedness and enhance route scheduling efficiency without the need for extensive hardware infras-

tructure [5]. By utilizing accelerometer and GPS data from commuters' smartphones, the system detects events such as bus boarding and whether a passenger was standing or seated during their journey. This innovative method provides real-time, energy-efficient crowd detection by minimizing GPS usage, which is typically power-intensive. Evaluations conducted on arterial roads in Ahmedabad and Gandhinagar revealed that with just 8–12% commuter participation, more than 80% of route segments could be accurately assessed for crowd levels. The historical crowdedness data are then used to optimize bus scheduling, deploying feeder buses to alleviate congestion in heavily traveled segments. Key advantages include scalability, reduced dependence on additional infrastructure, and adaptability for periodic data-driven adjustments. However, the system's effectiveness declines with lower commuter participation, and its reliance on smartphone sensors limits accuracy in areas with reduced smartphone penetration. Despite these challenges, the approach demonstrates a viable path toward intelligent transportation systems that integrate seamlessly with existing resources, enhancing commuter experiences and operational efficiency.

2.2 Summary and Gaps Identified

The studies reviewed cover various facets of transit optimization, including real-time information processing, route planning, and passenger behavior analysis. MOBANA contributes a robust framework for real-time information processing, beneficial for large-scale public transit systems. The logistics optimization study highlights GPS and GIS technology's value in reducing costs and improving route efficiency, while the spatiotemporal embedding approach for mobility pattern recognition provides insights into passenger flow that can enhance schedule planning. The memetic algorithm study demonstrates an efficient solution for complex route optimization, especially in multimodal transit networks.

Despite these advancements, several research gaps remain:

1. Limited focus on real-time adaptation to sudden changes (e.g., traffic disruptions or unexpected demand fluctuations).

2. High computational requirements for processing spatiotemporal data in real-time applications.

3. Lack of a comprehensive framework that can seamlessly integrate multiple data sources (e.g., GPS, social media, passenger data) for holistic transit insights.

4. Inadequate attention to data security and privacy in public transit data sharing systems.

5. Limited research on optimizing parameter tuning in hybrid algorithms for consistent performance across varying transit environments.

2.3 Conclusion

This paper explores the latest research on the role and limitations of real-time public transit information systems, logistics route optimization, and passenger mobility pattern recognition. Although these studies provide novel solutions, some of the missing aspects including real-time flexibility, compute efficiency and central data management remain. These constraints will need to be addressed in order to create a more adaptive, flexible and safe transit management system that adapts to the changing requirements of urban transportation.

Chapter 3

System Design

This chapter outlines the technical framework and organizational structure underlying our project. The system architecture displays top-level components such as front-end, back-end, database and third-party APIs and shows detailed module designs for user authentication data processing and real-time tracking. The process of creating algorithms for bus scheduling and delay prediction receives emphasis through illustrations provided by data flow diagrams (DFDs) and use case diagrams which demonstrate data interactions. The documentation specifies both the tools, technologies, and datasets utilized as well as the modular work breakdown structure. The main outputs of this project consist of a functioning prototype and optimized algorithms besides a live monitoring dashboard and a Gantt chart demonstrating the project timeline for systematic execution.

3.1 System Architecture

The Automated Bus Scheduling and Route Management System system architecture processes both real-time and static data effectively and maintains a responsive user interface. The architecture comprises the following components:

1. **Frontend:**
    - Developed using React.js for an intuitive and responsive user interface.
    - Leverages Leaflet for dynamic geospatial visualization of bus locations and routes.
2. **Backend Services:**
    - **Static Data Service:** Processes and manages static GTFS (General Transit Feed Specification) data (e.g., routes, stops, and schedules).

- **Real-Time Data Service:** Handles incoming real-time updates from GPS devices via the Open Transit Data Delhi API.
- **Data Optimization Service:** Implements algorithms for route optimization, and scheduling.
- **Notification Service:** Facilitates communication for system updates, alerts, and notifications to stakeholders.

3. **Database:**
   - Utilizes MongoDB to store static GTFS data and real-time logs, ensuring efficient storage and retrieval.
   - Maintains a historical dataset for analytics and reporting purposes.

3.1.1 System Architecture Diagram

The high-level architecture is illustrated in Figure 3.1, showcasing the interaction between the services, third-party API, database, and frontend.

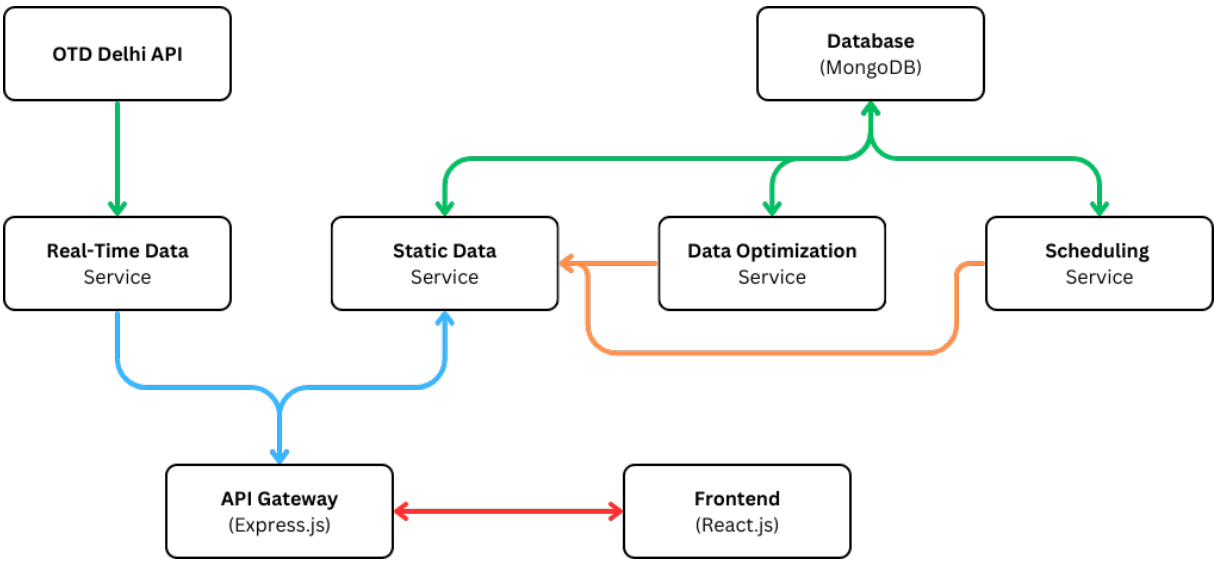


Figure 3.1: High-Level System Architecture

This architecture ensures scalability, seamless real-time data integration through the Open Transit Data API, and efficient resource utilization, meeting the operational demands of the DTC.

3.2 Component Design

This section examines how individual modules of the Automated Bus Scheduling and Route Management System were designed and put into operation. The documentation of each module includes details about its architectural structure while also explaining its operational workflows and how it connects with other system elements.

3.2.1 Route Management Module

- **Design Overview:**
  - Utilizes GIS tools and Open Transit Data Delhi API for creating and modifying routes.
  - Implements an adjacency matrix to analyze passenger flow between stops.
  - Implements a time matrix to estimate travel times between any two stops.
  - Generates alternative routes based on the adjacency and time matrices.
  - Normalizes passenger flow data to allocate trips proportionally based on demand.
- **Implementation Details:**
  - Input data includes stops, stop sequences, trips and routes from GTFS.
  - Passenger flow calculated by tracking consecutive stops in trip sequences.
  - Preprocessing involves calculating a crowding factor from the adjacency matrix and combining it with travel times to compute weighted edge costs.
  - Key stops are identified using demand analysis, focusing on high-demand stops and underutilized stops near congested routes.
  - Route optimization is performed by applying shortest-path algorithms (e.g., Dijkstra) with weighted edge costs to generate alternative paths. Intermediate underutilized stops are included in these paths.

3.2.2 Bus and Crew Scheduling Module

- **Design Overview**:

- Utilizes a Genetic Algorithm (GA) to efficiently allocate buses to multiple trips (bus scheduling) and drivers to buses (crew scheduling).
- Optimizes bus schedules to ensure efficient fleet utilization, minimizing idle time, meeting passenger demand, and reducing overall travel time.
- Generates driver schedules that align with assigned buses while ensuring balanced workload, minimized fatigue, and compliance with work-hour regulations and break periods.
- Dynamically adjusts both bus and crew schedules based on operational constraints and real-time updates, such as vehicle availability and driver shift limitations.

- **Implementation Details**:
  - Chromosome Representation:
    - Each chromosome encodes bus assignments to multiple trips, forming a complete bus schedule.
    - Driver assignments are mapped to buses, ensuring shifts comply with work regulations.
  - Fitness Function:
    - Evaluates solutions based on factors like demand satisfaction, minimized idle time, reduced travel time, and fair driver workload distribution.
    - Penalizes infeasible schedules, such as driver overwork or excessive layover time.
  - Optimization Process:
    - Selection, crossover, and mutation operators iteratively refine bus and driver assignments.
    - Ensures optimized schedules while maintaining adaptability to real-time changes.

3.2.3 Real-Time Tracking Module

- **Design Overview:**

- Integrates real-time data from the Open Transit Data API to track buses.
- Displays live locations and trip progress metrics for each bus on an interactive dashboard.
- Combines real-time and static data to provide insights on trip completion and adherence to schedules.

- **Implementation Details:**
  - Uses GTFS bindings to decrypt and process real-time data from the API.
  - Leaflet.js is employed to plot live bus locations on an interactive map.
  - A dashboard was developed to display information about each bus, including its current location, route, and schedule status.
  - The dashboard includes a search bar for quick retrieval of bus-specific details.
  - Static GTFS data on trips and routes is integrated to calculate and display metrics such as percentage of trip progress and estimated time to destination.

3.2.4 Data Analytics and Reporting Module

- **Design Overview:**
  - Analyzes historical and real-time data to derive actionable insights.
  - Generates customizable reports for operational and strategic planning.
  - Provides visualizations of key metrics such as trip density, punctuality.

- **Implementation Details:**
  - Uses Matplotlib to plot the density of bus stops based on the number of trips associated with each stop.
  - Employs Folium to create interactive maps for visual representation of routes and stops.
  - Utilizes Seaborn to generate polylines for multiple routes, offering enhanced data visualization capabilities.
  - Implements predictive analytics for demand forecasting and resource allocation using historical trends.

– Exports reports in standard formats for stakeholders, including summary tables and graphical insights.

The modular architecture ensures scalability, robustness, and effective integration of all components, meeting the complex demands of a modern public transportation system.

3.3 Algorithims

3.3.1 Adjacency Matrix Algorithm

The algorithm follows these steps:

1. **Load and Preprocess Data:**
   - Load the `stop_times.csv` file into a DataFrame using pandas.
   - Sort the data by `trip_id` and `stop_sequence` to ensure sequential order within each trip.
2. **Generate All Reachable Stop Pairs:**
   - Group the data by `trip_id`.
   - For each group, compute all reachable stop pairs $(from\_stop, to\_stop)$ where $i < j$ (using combinations of stops in the same trip).
3. **Count Frequency of Stop Pairs:**
   - Count how many trips contain each stop pair using the `value_counts()` method.
4. **Map Stops to Indices:**
   - Create a mapping of unique `stop_id` values to indices.
   - This mapping ensures each stop is associated with a unique row and column in the adjacency matrix.
5. **Initialize and Populate Adjacency Matrix:**
   - Create a zero-initialized square matrix of size $n \times n$, where n is the number of unique stops.

- For each stop pair, update the corresponding entry in the matrix with the trip count.

6. **Apply Floyd-Warshall Algorithm:**

   - For each intermediate stop k, update the adjacency matrix to consider indirect connections between stops i and j via k.
   - The weight of the connection (i, j) is updated as:

$$adj[i][j] = \max(adj[i][j], \min(adj[i][k], adj[k][j]).$$

7. **Normalize the Matrix:**

   - Replace all zero entries in the adjacency matrix with 1 to ensure minimal connectivity between all stops.

8. **Convert and Save Matrix:**

   - Convert the adjacency matrix into a DataFrame for better readability, using `stop_id` as row and column labels.
   - Save the resulting DataFrame to a CSV file for future use.

3.3.2 Genetic Algorithm for Bus Scheduling

Inputs:

- List of buses and trips.
- Number of generations to run the algorithm.
- Population size (number of schedules to evaluate at each step).
- Mutation rate (probability of making random changes).

Steps:

1. **Create an initial population:** Start with a group of random schedules. Each schedule assigns trips to buses in a random way.

2. **Evaluate fitness**: For each schedule, calculate its fitness score, which measures the quality of the schedule based on the following factors:

   - Bus Utilization: The extent to which each bus is effectively used. Higher utilization improves the score.
   - Average Trips per Bus: The number of trips assigned to each bus. A higher average with minimal idle time is desirable.
   - Efficiency Score: A composite score reflecting overall efficiency, including minimizing idle time and distance traveled. The less idle time and travel distance, the higher the score.
   - Overlaps: Any overlapping trips between buses. This is penalized, with more overlaps leading to a lower fitness score.
   - Infeasibility: If the schedule violates constraints (such as driver work hour regulations or bus availability), it is penalized.

   The fitness value is calculated using the formula:

$$\mathrm{f} = 0.4 \times U + \frac{0.3}{T+1} + \frac{0.2}{E+1} + \frac{0.1 \times O}{X+1} \tag{3.1}$$

   Where:

   - f = Fitness Value
   - U = Bus Utilization
   - T = Average Trips per Bus
   - E = Efficiency Score
   - O = Overlaps
   - X = Total Trips

3. **Select parents:** Choose the best-performing schedules to become parents for the next generation.

4. **Create a new population:** Generate a new group of schedules using:

   - **Crossover:** Combine parts of two parent schedules to create new schedules.

- **Mutation:** Make random changes to some schedules to introduce variety.

5. **Repeat:** Repeat steps 2–4 for the specified number of generations.

6. **Find the best solution:** After completing all generations, identify the schedule with the best fitness score.

3.3.3 Genetic Algorithm for Driver Scheduling

Inputs:

- List of drivers and buses.
- List of trips to be assigned to drivers.
- Number of generations to run the algorithm.
- Population size (number of schedules to evaluate at each step).
- Mutation rate (probability of making random changes).

Steps:

1. **Create an initial population:** Start with a group of random schedules. Each schedule assigns drivers to buses for the required trips in a random way.

2. **Evaluate fitness:** For each schedule, calculate its fitness score based on the following factors:

    - Active Time Ratio: Maximize the active driving time for each driver. The higher the ratio of active driving time to total assigned time, the better.
    - Overlap Penalty: Any overlaps in driver shifts or assignments are penalized, reducing the fitness score.
    - Break Penalty: If a driver does not take required breaks or takes insufficient breaks, it results in a heavy penalty.
    - Work Hour Penalty: Violation of work hour regulations (e.g., exceeding the maximum allowable driving hours) results in a severe penalty.

- Idle Penalty: Penalty for idle time where a driver is not actively engaged in driving. Minimizing idle time improves the fitness score.
- Underutilization Penalty: Penalty for assigning underutilized drivers to fewer trips or idle periods.
- Trip Distribution Score: Encourages a more even distribution of trips across drivers. A lower score is penalized, rewarding more balanced workloads.

The fitness value is calculated using the formula:

$$f = \left(\frac{1.0}{A + 0.1}\right) + O + B + W + I + U + (1.0 - D) \tag{3.2}$$

Where:

- f = Fitness value
- A = Active driving time ratio for each driver.
- O = Penalty for any overlap in driver shifts or assignments.
- B = Penalty for insufficient or missing breaks.
- W = Penalty for violating work hour regulations.
- I = Penalty for idle time where the driver is not actively engaged in driving.
- U = Penalty for underutilization of drivers (assigned fewer trips).
- D = A score for the distribution of trips among drivers, with a lower score penalized.

3. **Select parents:** Choose the best-performing schedules (those with the highest fitness values) to serve as parents for the next generation.
4. **Create a new population:** Generate a new group of schedules using:
   - Crossover: Combine parts of two parent schedules to create new, potentially improved schedules.
   - Mutation: Introduce random changes to some schedules to introduce variety and explore new solutions.

5. **Repeat:** Repeat steps 2–4 for the specified number of generations, iteratively improving the schedule.

6. **Find the best solution:** After completing all generations, identify the schedule with the best fitness score as the optimal driver assignment solution.

3.4 Data Flow Diagrams

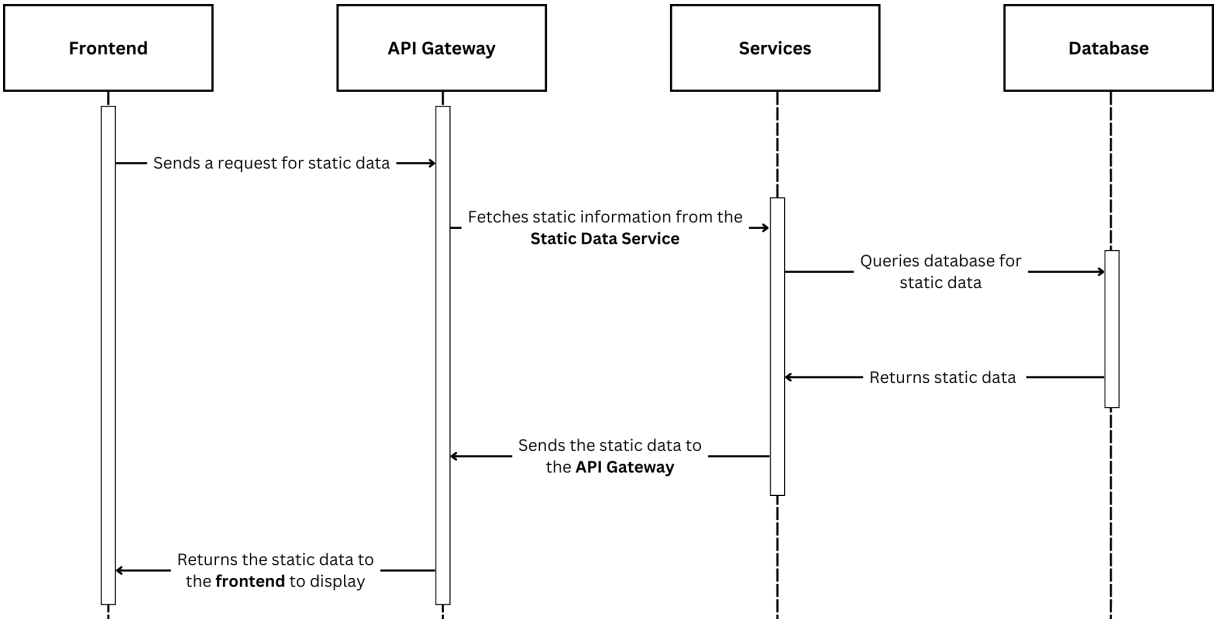


Figure 3.2: Static Information Sequence Diagram

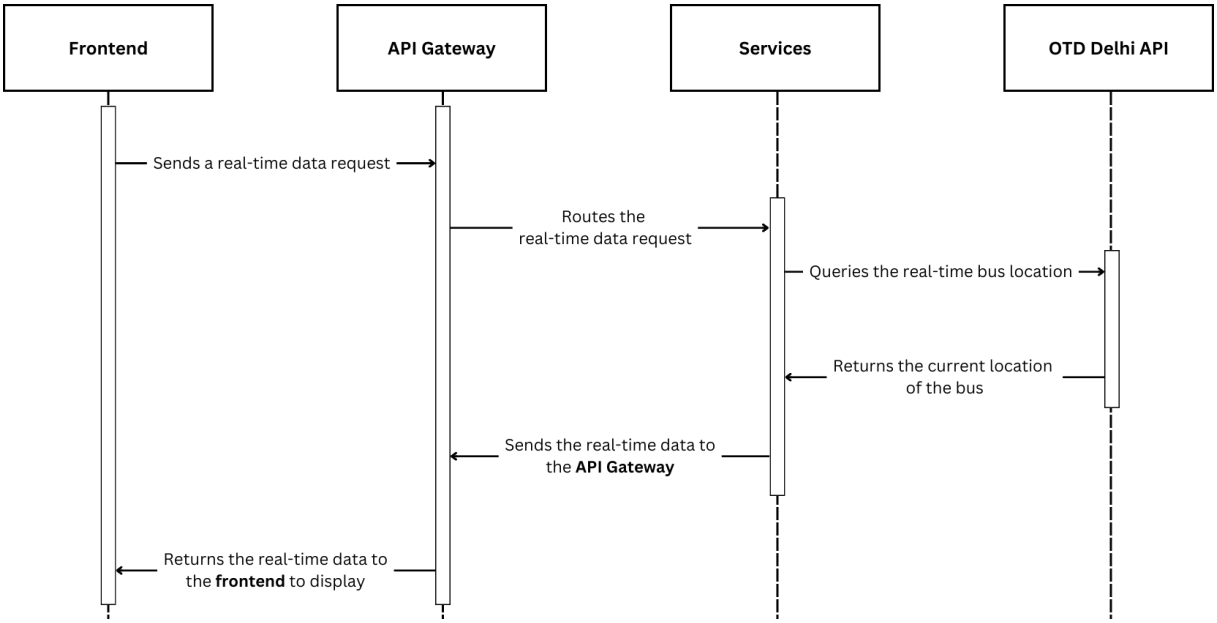


Figure 3.3: Real-time Data Sequence Diagram

3.5 Tools and Technologies: Software and Hardware Requirements

3.5.1 Hardware Requirements

The following hardware components are recommended for optimal system performance:

- **Processor:** Intel Core i5 or higher
- **RAM:** 8 GB
- **Storage:** 256 GB SSD
- **Graphics Card:** NVIDIA GeForce GTX 1650 (4 GB GPU)
- **Network:** High-speed Ethernet connection for real-time data communication
- **Display:** Full HD monitor (1920x1080)

3.5.2 Software Requirements

The system requires the following software components for development and deployment:

- **Operating System (OS):** Windows 10 64-bit or Ubuntu 20.04 LTS
- **Backend Development:** Node.js (Express)

- **Database:** MongoDB
- **Frontend Development:** React.js for creating an interactive user interface
- **Mapping Tools:** Leaflet.js with OpenStreetMap and OpenRoute Service for distance/time calculation

By utilizing the above hardware and software specifications, the system can ensure optimal performance and accuracy in handling real-time operations and data visualization.

3.6 Dataset Identified

3.6.1 Overview of the Dataset

The datasets used are in the form of CSV files, containing static data related to bus routes, stops, shapes, trip timings, and schedules. Below is a detailed description of each dataset:

3.6.2 Description of Datasets

Routes Dataset (routes.csv)

- **agency_id:** Agency responsible for the route.
- **route_id:** Unique identifier for the route.
- **route_long_name:** Full name of the route.
- **route_short_name:** Shortened name or number for the route.
- **route_type:** Type of transportation (e.g., bus, train).

Shapes Dataset (shapes.csv)

The shapes dataset describes the geographical path of each route using a series of points. Each point is identified by a shape ID, along with its latitude and longitude coordinates. The dataset structure includes:

- **shape_id:** Unique identifier for the shape.

- **shape_pt_lat:** Latitude of the shape point.
- **shape_pt_lon:** Longitude of the shape point.
- **shape_pt_sequence:** Sequence number indicating the order of points along the shape.
- **shape_dist_traveled:** Distance traveled along the shape up to the point.

Stop Times Dataset (stop_time.csv)

This dataset provides timing information for stops in a trip. It includes details such as the stop sequence and arrival/departure times. The fields in this dataset are:

- **trip_id:** Identifier for the trip.
- **arrival_time:** Time of arrival at the stop.
- **departure_time:** Time of departure from the stop.
- **stop_id:** Unique identifier for the stop.
- **stop_sequence:** Order of the stop in the trip.

Stops Dataset (stops.csv)

The stops dataset contains detailed information about all the bus stops in the network. It includes coordinates, names, and descriptions of the stops. The dataset fields are:

- **stop_id:** Unique identifier for the stop.
- **stop_code:** Code assigned to the stop.
- **stop_name:** Name of the stop.
- **stop_desc:** Description of the stop (if available).
- **stop_lat:** Latitude of the stop.
- **stop_lon:** Longitude of the stop.

Trips Dataset (trips.csv)

The trips dataset contains information about scheduled trips for each route. It links routes to shapes and specifies accessibility features. The dataset structure includes:

- **route_id:** Identifier for the route.
- **service_id:** Service period (e.g., weekday, weekend).
- **trip_id:** Unique identifier for the trip.
- **trip_headsign:** Head sign or destination of the trip.
- **shape_id:** Identifier for the shape representing the trip's path.
- **wheelchair_accessible:** Indicates whether the trip is wheelchair accessible.
- **bikes_allowed:** Indicates whether bicycles are allowed on the trip.

3.7 Module Divisions and Work Breakdown

3.7.1 Overview of Modules

The project is divided into the following key modules, each addressing critical areas of the system:

- **Route Management Module**
- **Bus and Crew Scheduling Module**
- **Geographical Visualization Module**
- **UI/UX and Data Analysis Module**
- **Crew Management Module**
- **Real-Time Tracking Module**
- **Data Analytics and Reporting Module**

3.7.2 Module Descriptions and Functionalities

Route Management Module

- Manage existing routes and allow the creation of new routes based on real-time data such as traffic congestion.
- Optimize routes to avoid overlaps, ensuring efficient service.
- Utilize GIS technology and real-time data from the Open Transit Data Delhi API for accurate route adjustments and planning.
- Minimize travel time and improve overall service reliability through optimized route planning.

Bus Scheduling Module

- Automate the assignment of routes to buses, ensuring efficient utilization of the fleet.
- Utilize GIS integration to optimize route assignments based on traffic conditions and demand.
- Reduce manual errors and improve operational efficiency by streamlining the route assignment process.
- Implement a genetic algorithm for bus scheduling:
    1. Define the problem and input data.
    2. Initialize a population with random bus assignments.
    3. Evaluate fitness based on idle time, demand satisfaction, and travel time.
    4. Perform selection, crossover, and mutation to evolve the solution.
    5. Extract the best solution for optimal bus allocation.

Geographical Visualization Module

- Use mapping tools to provide a visual representation of bus routes and stops.

- Display real-time bus locations using GPS data.
- Enable users to interact with geographical data for better understanding and decision-making.

UI/UX and Data Analysis Module

- Design an intuitive and user-friendly interface for accessing bus schedules and tracking.
- Analyze static and dynamic data for insights into system performance.
- Present data in an easily understandable format using charts and reports.

Crew Management Module

- Automate duty scheduling for drivers and conductors, ensuring compliance with shift regulations.
- Track performance metrics like punctuality and shift compliance.
- Optimize crew allocation and improve productivity through seamless scheduling and monitoring.

Real-Time Tracking Module

- Collect GPS data to track bus locations and display them in real-time on the user interface.
- Provide notifications for delays or traffic incidents to users.

Data Analytics and Reporting Module

- Use analytics tools to assess system efficiency and identify areas for improvement.
- Generate real-time and historical reports on bus performance, including timeliness and route optimization.

3.8 Key Deliverables

The key deliverables of the project are designed to enhance efficiency and responsiveness in DTC's operations. These include:

1. **Optimized Scheduling Algorithms:**
    - Development of advanced algorithms for bus scheduling, considering route priorities, fleet availability, and passenger demand.
    - Automation of crew duty scheduling with linked and unlinked duty rosters to ensure fair workload distribution and compliance with labor regulations.
2. **Real-Time Location Tracking and Dashboard:**
    - A live, interactive dashboard displaying the exact location of all buses in real-time using GPS data.
    - Delay predictions and visualized route maps that dynamically update based on traffic conditions and disruptions.
3. **Analytics-Driven Insights:**
    - Generation of reports on delays, crowdedness, and resource utilization using real-time and historical data.
    - Analysis of congestion trends, route performance, and crew efficiency to support informed decision-making.
4. **Reports and Visualizations:**
    - Delay and crowdedness reports to enable on-the-fly adjustments to schedules or resources.
    - Historical analytics for insights into recurring inefficiencies like underutilized buses or congested routes.
5. **Scalability and Customization:**
    - The system is scalable to accommodate DTC's growing network and increasing fleet size.

- Customizable outputs for easy adaptation to new routes, additional buses, or integration with other transportation systems.

3.9 Gantt Chart

The Gantt Chart below outlines the key phases and milestones of the project:

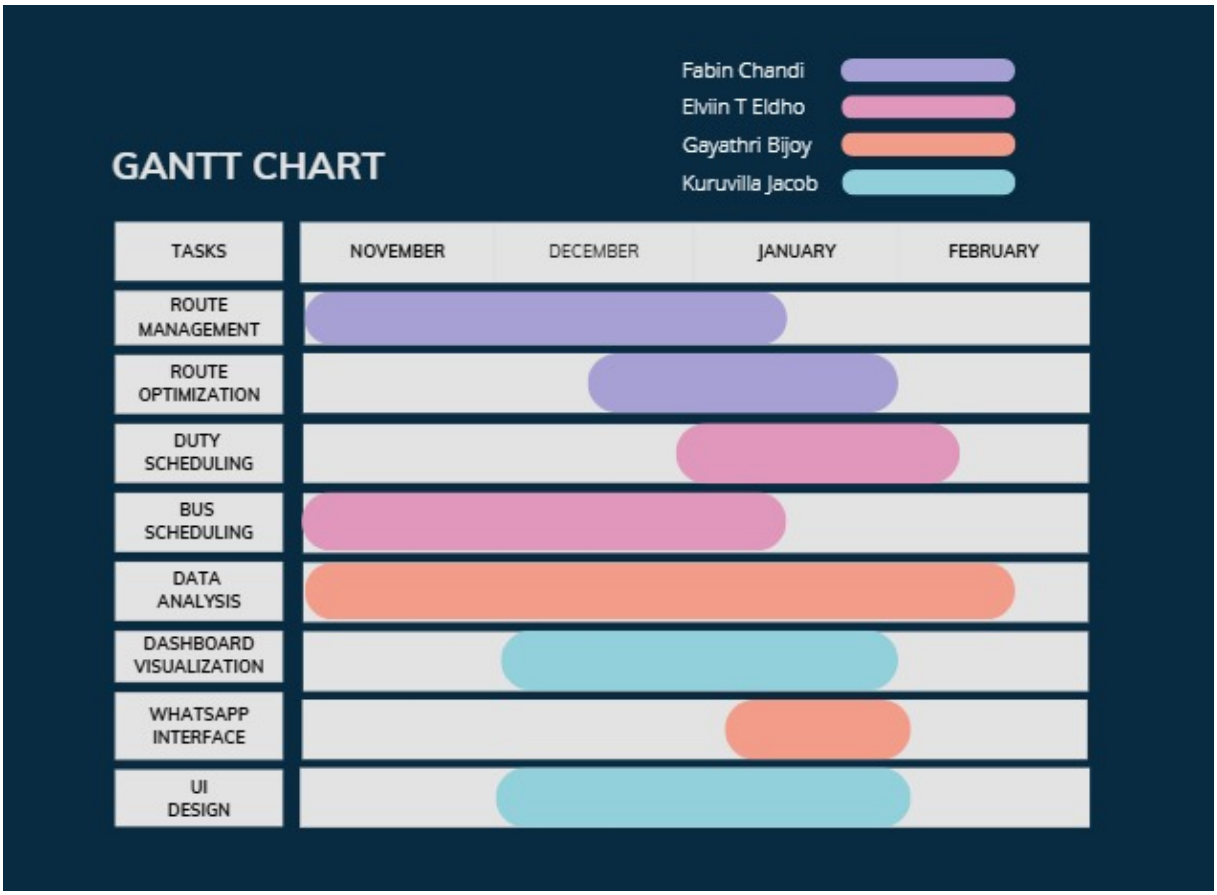


Figure 3.4: Gantt Chart for Project Timeline

Chapter 4

Results and Discussion

This chapter presents a comprehensive analysis of bus performance metrics, fitness score evolution, fitness score components, and correlation insights. The results provide valuable learnings on optimizing fleet efficiency.

4.1 Bus Performance Metrics Analysis

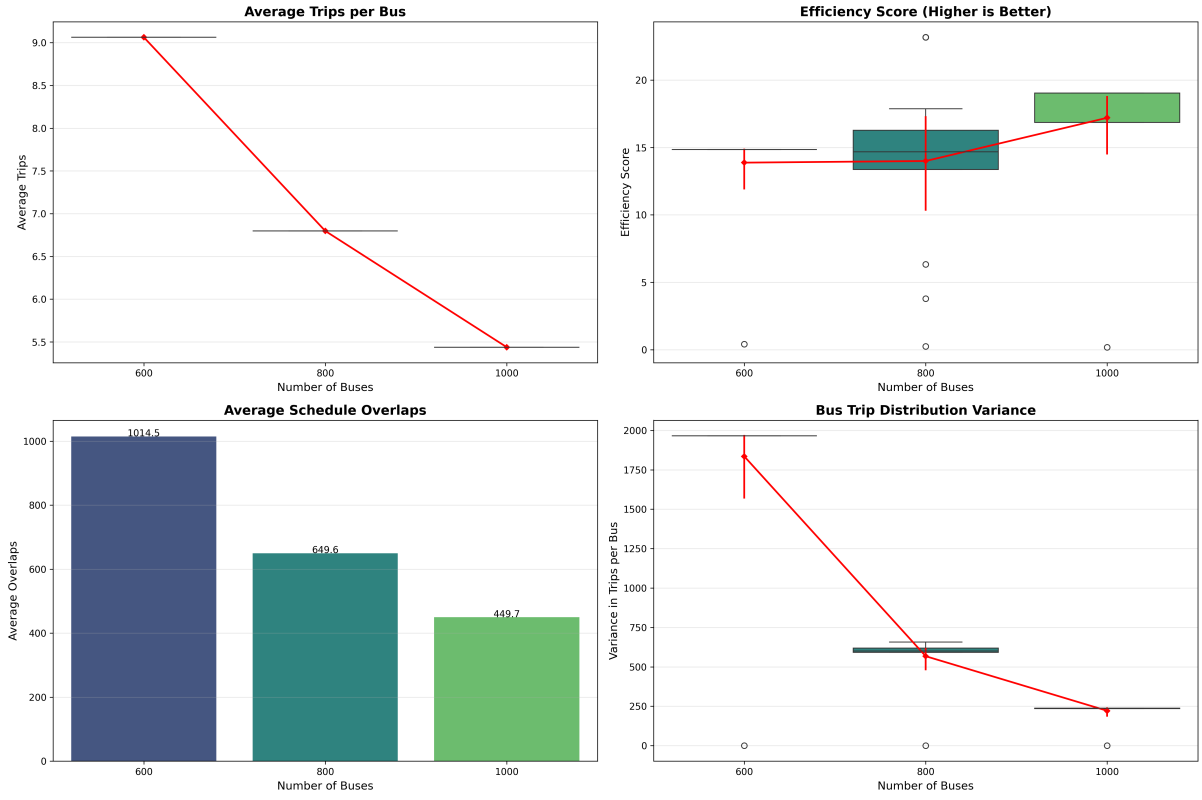


Figure 4.1: Comparison of key bus performance metrics across different fleet sizes (600, 800, and 1000 buses).

This section analyzes four key performance indicators: average trips per bus, efficiency score, schedule overlaps, and trip distribution variance for fleet sizes of 600, 800, and 1000

buses.

4.1.1 Average Trips per Bus

The top-left graph illustrates that as fleet size increases, the average number of trips per bus decreases. This trend, indicated by the red line, suggests that additional buses distribute the workload more evenly.

4.1.2 Efficiency Score

The top-right graph presents efficiency scores, with higher values indicating better efficiency. The box plots and red trend line reveal a general improvement in efficiency with increased fleet size, reflecting operational enhancements.

4.1.3 Schedule Overlaps

The bottom-left graph shows that schedule overlaps decline as the number of buses increases. This reduction indicates improved scheduling efficiency, minimizing route redundancy.

4.1.4 Trip Distribution Variance

The bottom-right graph represents the variance in trips per bus. The red trend line indicates that a larger fleet leads to a more balanced distribution of trips, reducing disparities among buses.

Overall, the analysis suggests that increasing fleet size reduces individual workload, enhances efficiency, minimizes overlaps, and balances trip distribution.

4.2 Evolution of Fitness Scores Across Generations

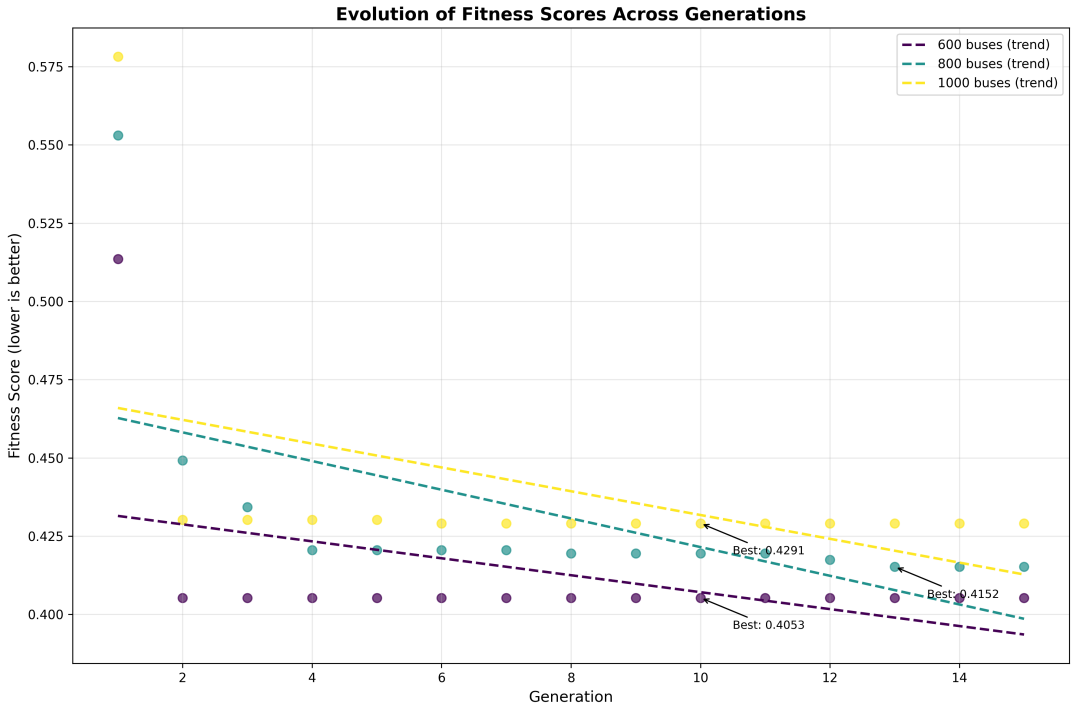


Figure 4.2: Fitness Score Evolution Across Generations

This section explores the evolution of fitness scores across multiple generations, assessing optimization performance for fleet sizes of 600, 800, and 1000 buses.

4.2.1 Fitness Score Trends

The graph tracks fitness scores over time, where lower values indicate better optimization. The dashed trend lines for fleet sizes—600 (purple), 800 (teal), and 1000 (yellow)—demonstrate a consistent decrease in scores, confirming the optimization process's effectiveness.

4.2.2 Best Fitness Scores

Annotations highlight the best fitness scores:

- 600 buses: **0.4053** (lowest)
- 800 buses: **0.4152**

- 1000 buses: **0.4291**

Smaller fleets appear to achieve better optimization results.

4.2.3 Conclusion

The analysis indicates that all fleet sizes improve fitness scores over time. However, the final fitness scores vary, suggesting that different fleet sizes require distinct optimization strategies.

4.3 Analysis of Fitness Score Components

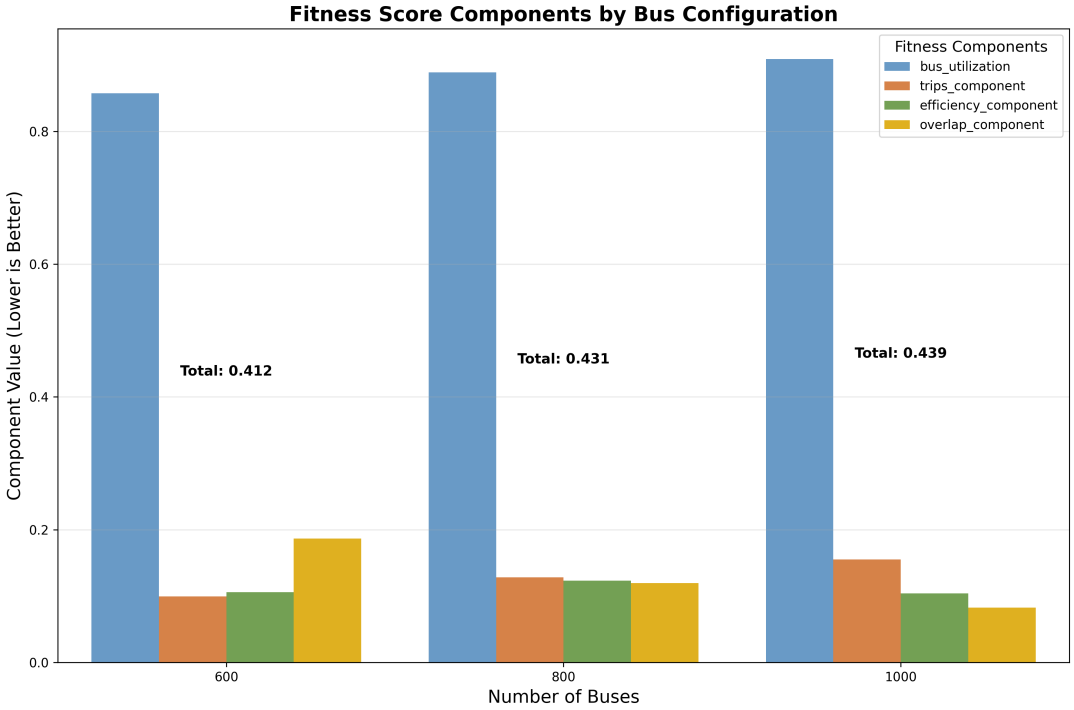


Figure 4.3: Breakdown of Fitness Score Components for Different Fleet Sizes

Figure 4.3 dissects the fitness score into key components for fleet sizes of 600, 800, and 1000 buses.

4.3.1 Key Observations

- **Dominance of Bus Utilization:** The largest contributor to the fitness score is *bus utilization* (blue bars), highlighting its importance in overall efficiency.

- **Effect of Fleet Size on Total Fitness Score:** The total fitness score increases slightly from **0.412** (600 buses) to **0.439** (1000 buses). This suggests that merely increasing fleet size does not guarantee better efficiency.
- **Impact of Other Components:** The *trips component*, *efficiency component*, and *overlap component* show smaller variations across fleet sizes, indicating their secondary role in influencing the total fitness score.
- **Optimal Fleet Size Considerations:** The lowest fitness score at 600 buses suggests this configuration may be the most efficient. Larger fleets introduce diminishing returns due to potential inefficiencies.
- **Strategic Implications:** Instead of merely increasing fleet size, efforts should focus on optimizing bus utilization and minimizing overlaps for a more efficient transit system.

4.4 Correlation Analysis of Fitness Metrics

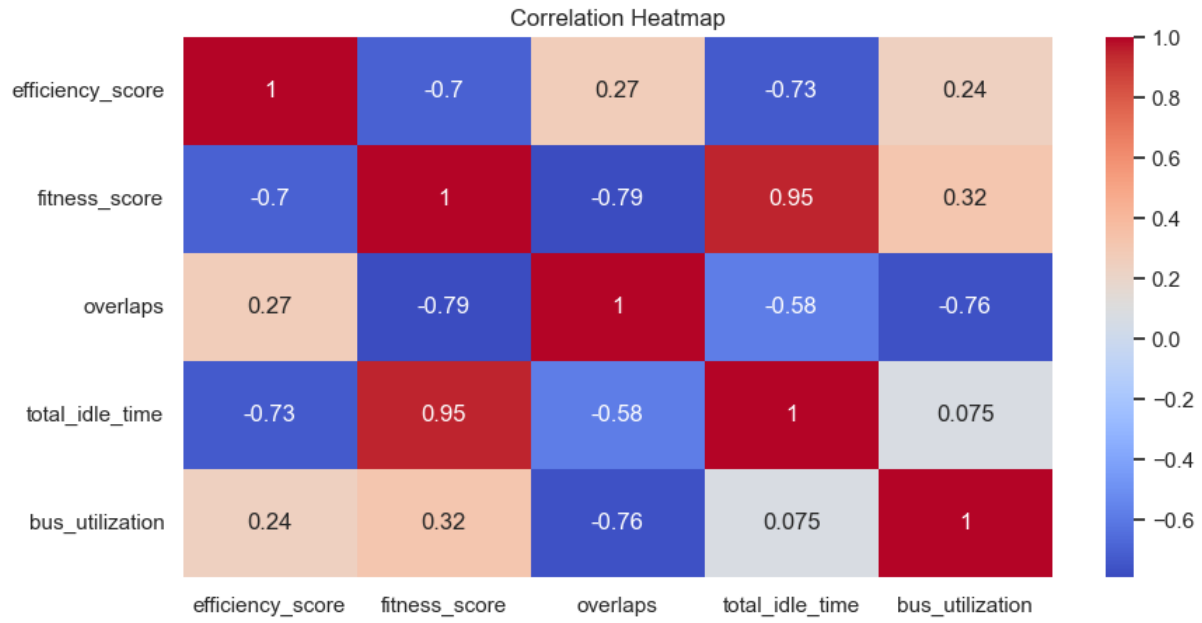


Figure 4.4: Correlation Heatmap of Key Fitness Metrics

Figure 4.4 provides insights into relationships between key performance metrics, where the color scale ranges from -1 (strong negative correlation) to 1 (strong positive correlation).

4.4.1 Key Findings

- **Efficiency Score vs. Fitness Score:** A strong negative correlation (**-0.7**) suggests that improved efficiency leads to better (lower) fitness scores.
- **Fitness Score vs. Total Idle Time:** A very high positive correlation (**0.95**) implies that reducing idle time is crucial for improving fitness scores.
- **Overlaps vs. Fitness Score:** A strong negative correlation (**-0.79**) indicates that higher overlaps worsen fitness scores.
- **Bus Utilization vs. Overlaps:** A negative correlation (**-0.76**) suggests that improved bus utilization reduces overlaps.
- **Total Idle Time vs. Efficiency Score:** A strong negative correlation (**-0.73**) reinforces that reducing idle time enhances efficiency.
- **Bus Utilization vs. Fitness Score:** A moderate positive correlation (**0.32**) suggests that simply increasing bus utilization does not always improve fitness scores, emphasizing the need for a balanced approach.

4.5 Optimization Performance Analysis

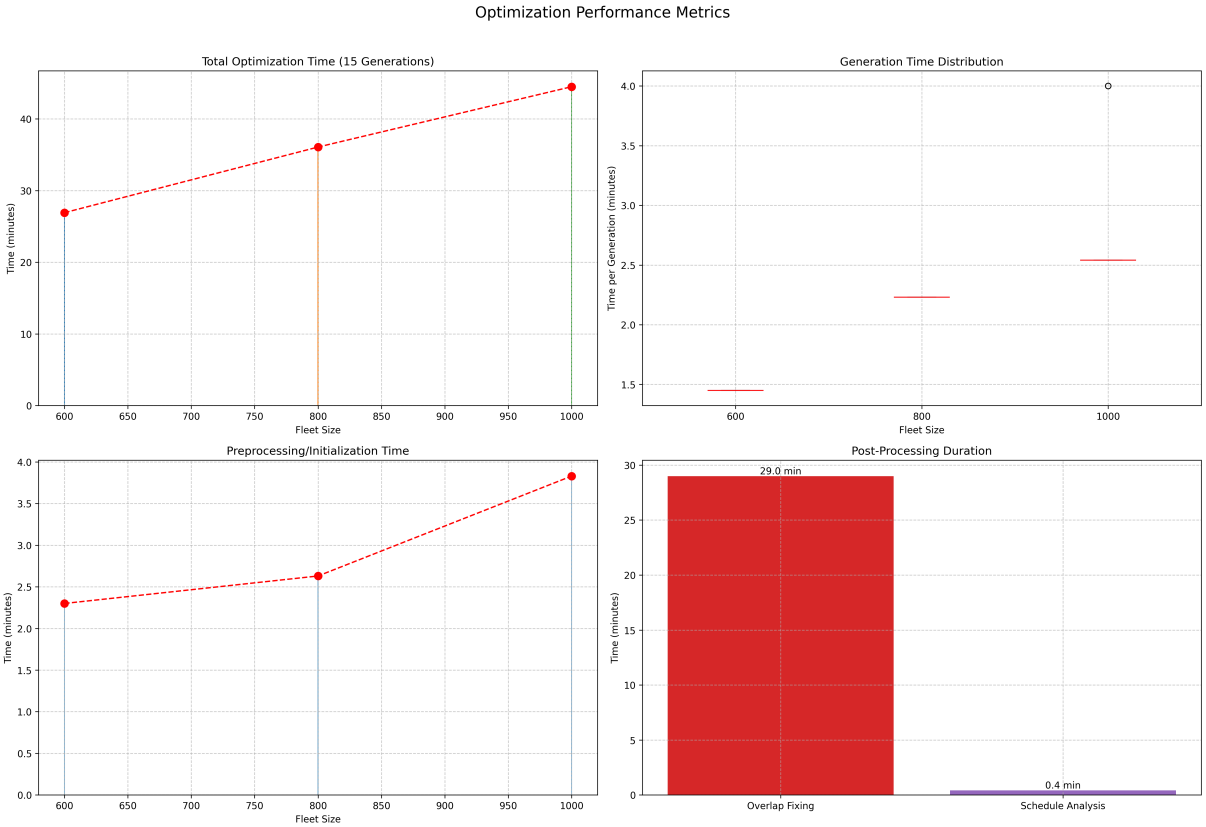


Figure 4.5: Comparison of optimization performance metrics for fleet sizes of 600, 800, and 1000 buses.

This section analyzes four key computational performance indicators: total optimization time, average generation time, preprocessing/initialization time, and post-processing duration across fleet sizes of 600, 800, and 1000 buses.

4.5.1 Total Optimization Time

The top-left graph shows the total runtime for completing 15 generations. The red trend line highlights a non-linear increase in duration as fleet size grows, with 600 buses taking 27 minutes versus 44.5 minutes for 1000 buses.

4.5.2 Average Generation Time

The top-right graph displays the mean time per generation. While the 600-bus fleet averaged 1m45s per generation, the 1000-bus fleet required 2m54s, with significant variance

observed in later generations (e.g., 4m00s for Gen6-7).

4.5.3 Preprocessing/Initialization Time

The bottom-left graph illustrates the time spent on data preprocessing and population initialization. Larger fleets required progressively longer setup periods, with 1000 buses needing 3m50s compared to 2m18s for 600 buses.

4.5.4 Post-Processing Overhead

The bottom-right graph quantifies post-optimization steps. Overlap fixing dominated with 29 minutes of runtime, exceeding the optimization time for 800 buses (36m05s). Schedule analysis completed faster at 25 seconds.

4.6 Scalability and Efficiency

The red trend lines in all graphs emphasize the algorithm's scalability limitations. The 33% increase from 600 to 800 buses caused a 36% longer runtime, while a 25% increase to 1000 buses added 21% more time, suggesting near-linear scaling for larger fleets but highlighting post-processing bottlenecks.

4.7 Recommendations

- Prioritize parallelization for overlap fixing to address the 29-minute post-processing delay
- Investigate initialization optimizations for large fleets (3m50s for 1000 buses)
- Implement hardware monitoring to explain generation time variances (e.g., 4m00s spike)
- Evaluate early stopping criteria for larger fleets given diminishing returns

4.7.1 Implications for Optimization

The correlation analysis suggests that reducing idle time and overlaps while improving efficiency and bus utilization are key to enhancing overall performance. These insights can guide scheduling improvements and fleet optimization strategies.

4.8 Conclusions

The analysis of bus performance metrics, fitness score evolution, and correlation insights reveals the following key takeaways:

- Increasing fleet size distributes workload more evenly but may introduce inefficiencies.
- Optimal fleet size (600 buses) achieves the lowest fitness score, indicating better efficiency.
- Bus utilization plays a dominant role in determining overall fitness.
- Reducing idle time and overlaps significantly improves efficiency and fitness scores.
- Optimization strategies should balance fleet size, utilization, and scheduling to achieve the best results.

This study provides a data-driven foundation for optimizing transit systems, ensuring efficient resource allocation and enhanced scheduling performance.

Chapter 5

Conclusions & Future Scope

The project is designed to revolutionize DTC's bus scheduling and crew management using cutting-edge algorithms, real-time data streams and interactive dashboards. The system achieves efficient resource utilization through optimized bus schedules together with the integration of both linked and unlinked duties. Real-time tracking capabilities along with delay predictions and analytics tools provide planners actionable data which enables them to boost reliability and minimize congestion while enhancing total operational efficiency.

The bus scheduling algorithm is designed to generate schedules randomly. However, this approach often leads to overlapping schedules, where multiple buses are assigned to the same route at the same time or conflict with other scheduled constraints. To mitigate this issue, a post-processing step is implemented to analyze and adjust the schedules by identifying overlaps and redistributing assignments accordingly. This refinement ensures that conflicts are resolved while maintaining efficiency in scheduling. Once the post-processing is completed, the algorithm selects the most optimal schedule from a set of 15 generated schedules. The selection is based on key parameters.

A similar methodology is employed for driver scheduling. Random allocation can result in conflicts such as drivers being scheduled for overlapping shifts or exceeding their working hour limits. Post-processing adjustments are applied to ensure fairness, compliance with regulations, and optimal distribution of workloads. The best schedule is chosen from 15 generated alternatives, prioritizing factors such as shift balance, availability, and compliance with labor constraints.

One of the significant challenges encountered in the scheduling system is the evaluation of performance. Performance assessments rely on fitness functions, but differences in parameter definitions and weightage can lead to inconsistencies in evaluation results. This variation makes it difficult to establish a universally accepted standard for determining the best schedule. Fine-tuning the fitness functions and incorporating more robust evaluation

criteria can help mitigate these discrepancies.

To further enhance the accuracy and effectiveness of the scheduling system, future improvements will focus on integrating additional datasets. Expanding the dataset will allow the algorithm to consider more real-world factors such as historical traffic patterns, peak demand periods, and driver preferences. By incorporating these elements, the scheduling system can improve decision-making, leading to more reliable and adaptable scheduling outcomes.

Upcoming improvements may consist of machine learning model integration for passenger flow prediction and delay estimation alongside the creation of a mobile app for commuters featuring live tracking capabilities and the establishment of connections between the system and smart city transportation networks. Scalability and environmental protection would improve if we extend the project to additional cities while implementing AI-driven crew scheduling and sustainability practices such as fuel optimization and emission monitoring. The application of automated response mechanisms alongside real-time incident detection will strengthen system resilience and manage disruptions successfully.

References

[1] M. A. Tianyi *et al.*, "Mobana: A distributed stream-based information system for public transit," in *2016 IEEE international conference on services computing (SCC)*. IEEE, 2016, pp. 1–6.

[2] R. Li *et al.*, "Research on logistics route optimization based on gps and gis technology," in *3rd International Conference on Electronic Information Technology and Intellectualization, ICEITI*, vol. 2017, 2017.

[3] C. Yu *et al.*, "Data-driven approach for passenger mobility pattern recognition using spatiotemporal embedding," *Journal of advanced transportation*, vol. 2021, no. 1, p. 5574093, 2021.

[4] O. Dib, M.-A. Manier, and A. Caminada, "Memetic algorithm for computing shortest paths in multimodal transportation networks," *Transportation Research Procedia*, vol. 10, pp. 745–755, 2015.

[5] P. Rajput, M. Chaturvedi, and V. Patel, "Opportunistic sensing based detection of crowdedness in public transport buses," *Pervasive and Mobile Computing*, vol. 68, p. 101246, 2020.

Appendix A: Presentation

RouteSync

Final Presentation

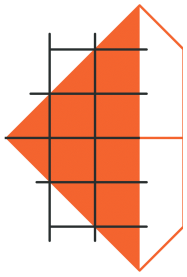
Members
Fabin Chandi
Elviin Thomas Eldho
Gayathri Bijoy
Kuruvilla Jacob

Guide
Dr. Sminu Izudheen
Professor

Figure 5.1: Slide 1

Problem Definition

The Delhi Transport Corporation (DTC) faces inefficiencies in bus scheduling and route planning due to manual processes, causing conflicts, route overlaps, and unreliable service. An automated system is needed to streamline scheduling, optimize routes, and improve public transportation in Delhi.



2

Figure 5.2: Slide 2

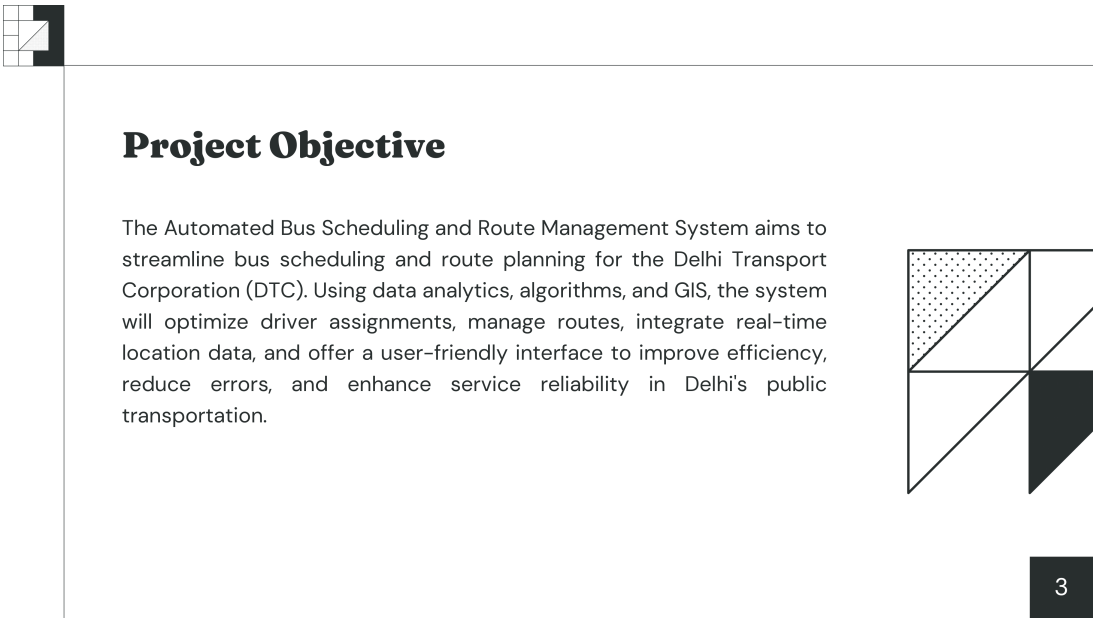


Figure 5.3: Slide 3

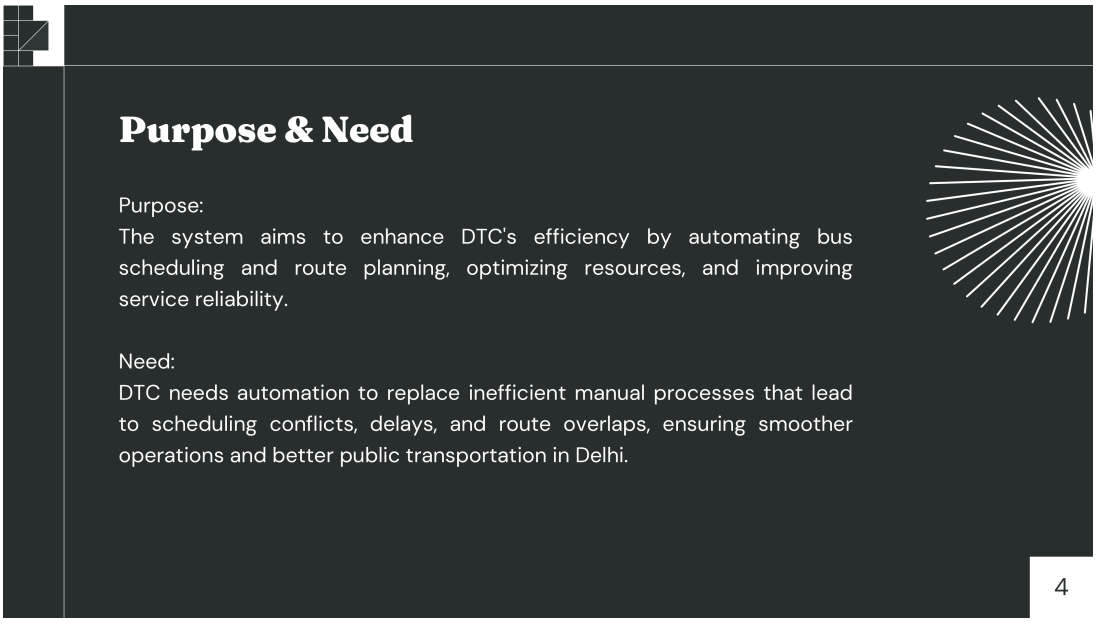


Figure 5.4: Slide 4

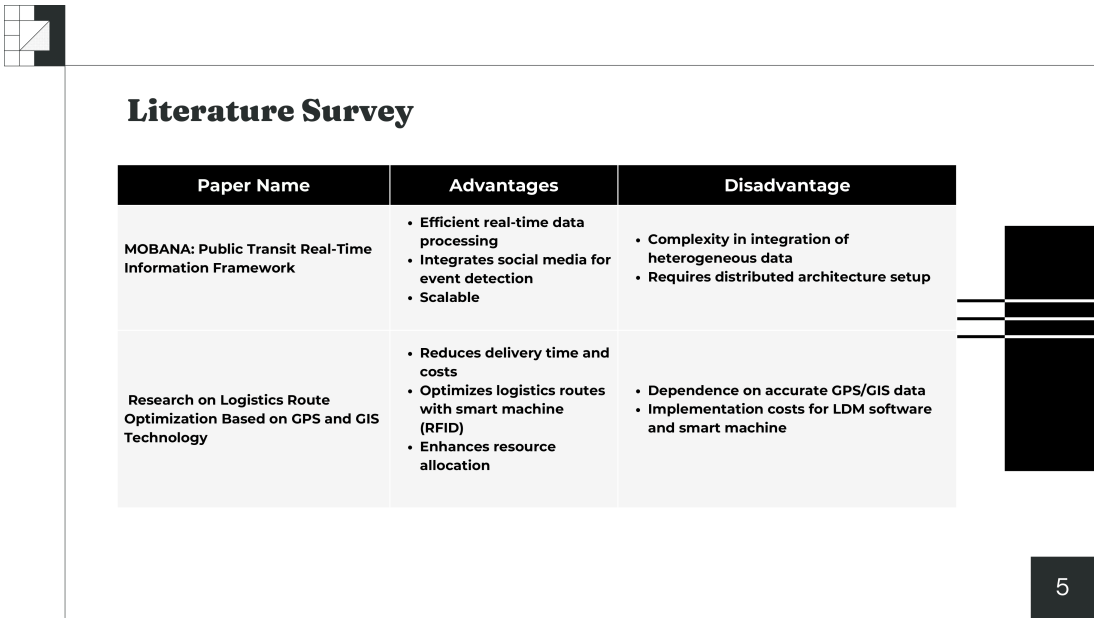


Figure 5.5: Slide 5

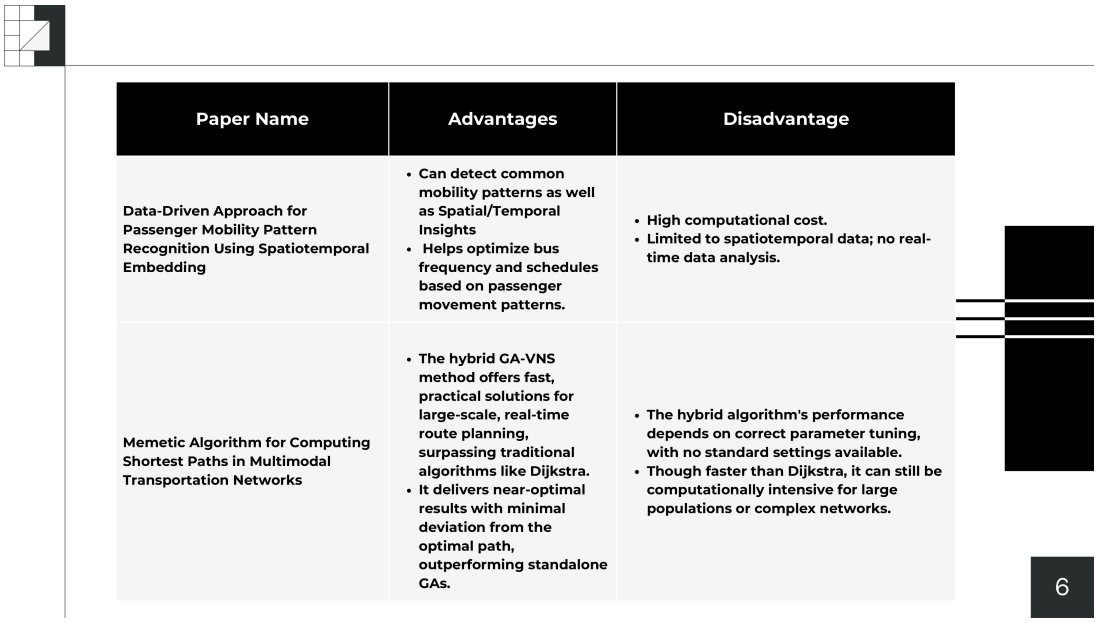


Figure 5.6: Slide 6

Paper Name	Advantages	Disadvantage
Opportunistic Sensing for Crowd Detection Using Smartphones	• Detects bus crowdedness using commuter smartphone data, improving route scheduling. • Conserves energy by minimizing GPS usage, making it suitable for real-time operations.	• Effectiveness drops with low commuter participation in the smartphone app. • Relies heavily on smartphone sensors, limiting accuracy in areas with lower smartphone usage.

7

Figure 5.7: Slide 7

Proposed Method

1 Historical Data Analysis
- Data Collection
- Data Cleaning
- Trend Analysis

2 Network Creation
- Mapping Stops and Routes
- Information Integration
- Network Evaluation

8

Figure 5.8: Slide 8

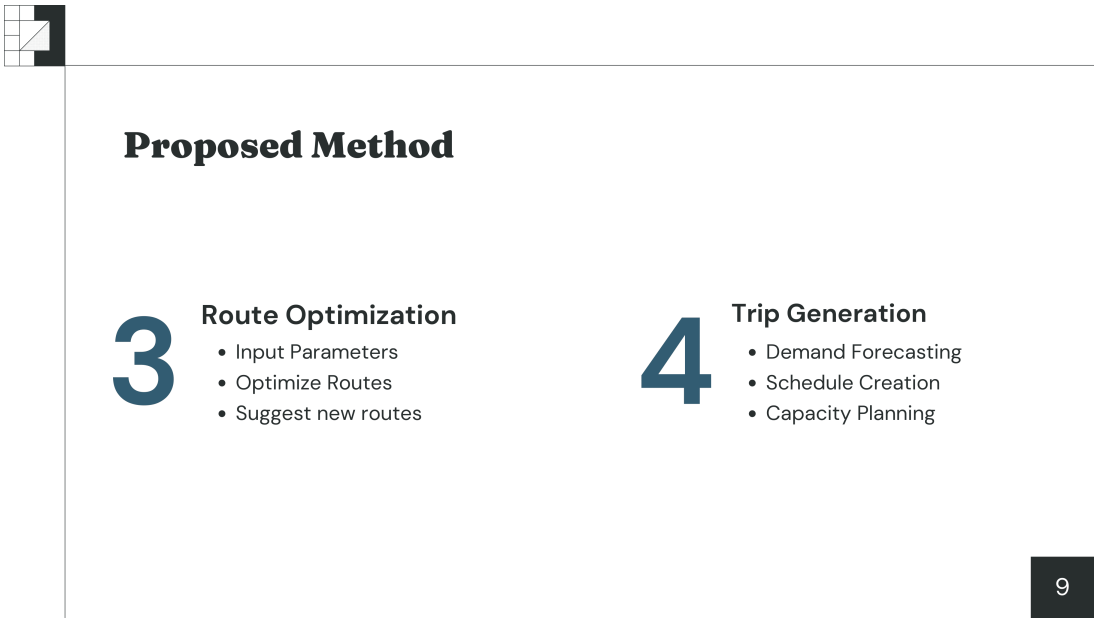


Figure 5.9: Slide 9

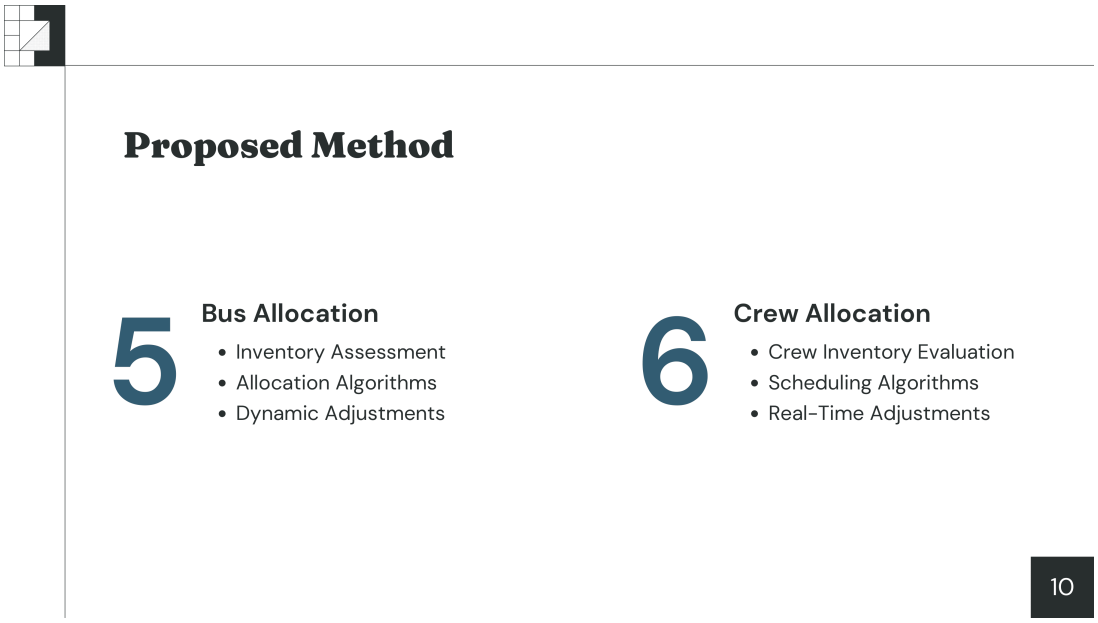


Figure 5.10: Slide 10

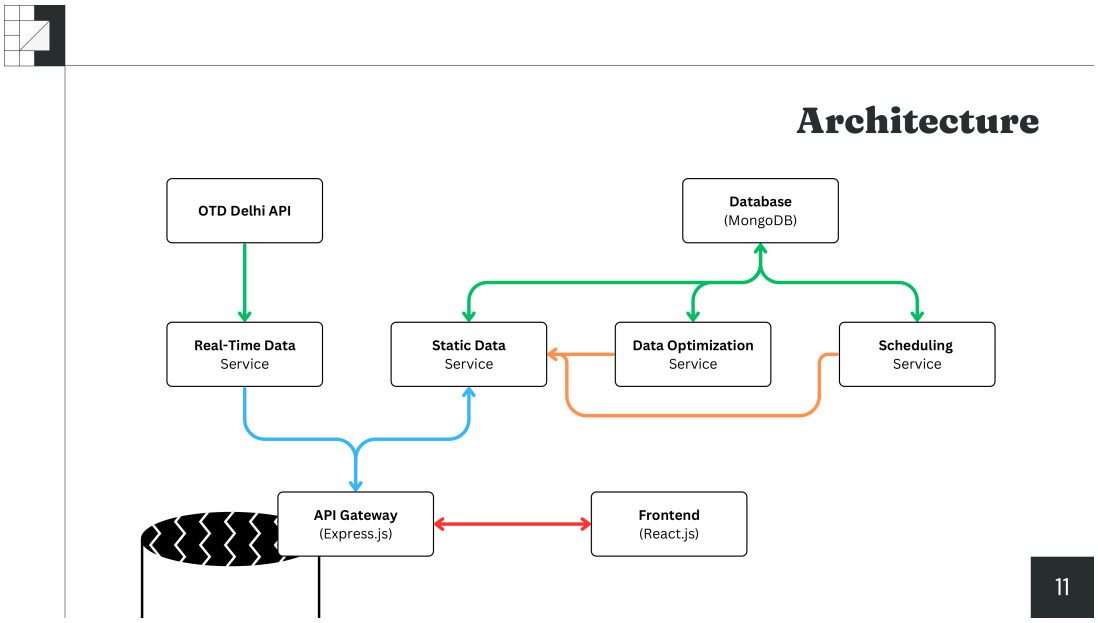


Figure 5.11: Slide 11

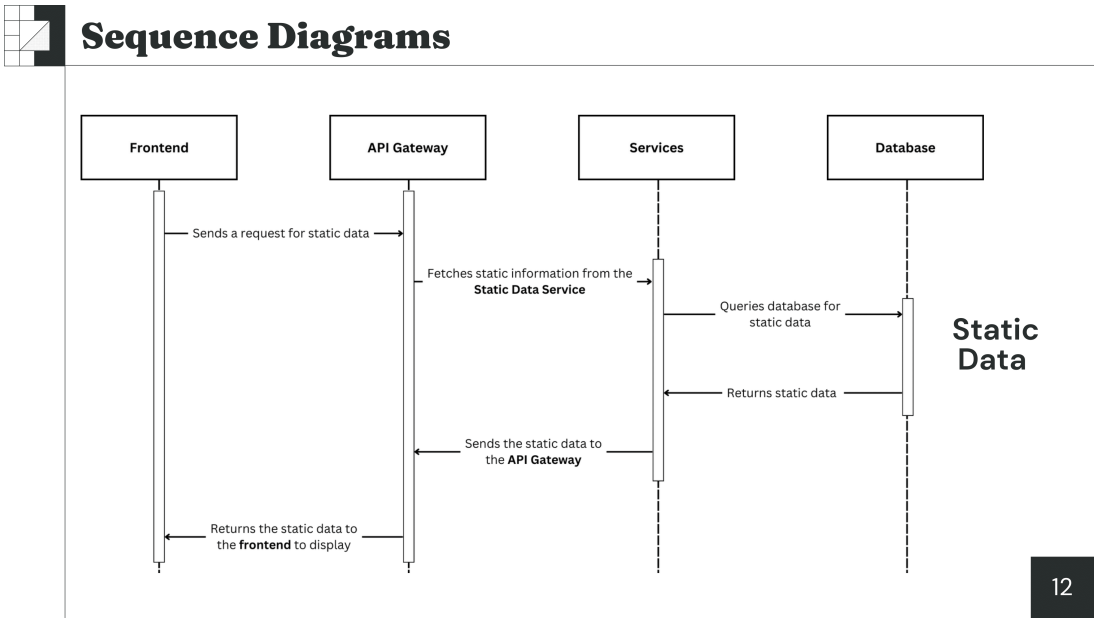


Figure 5.12: Slide 12

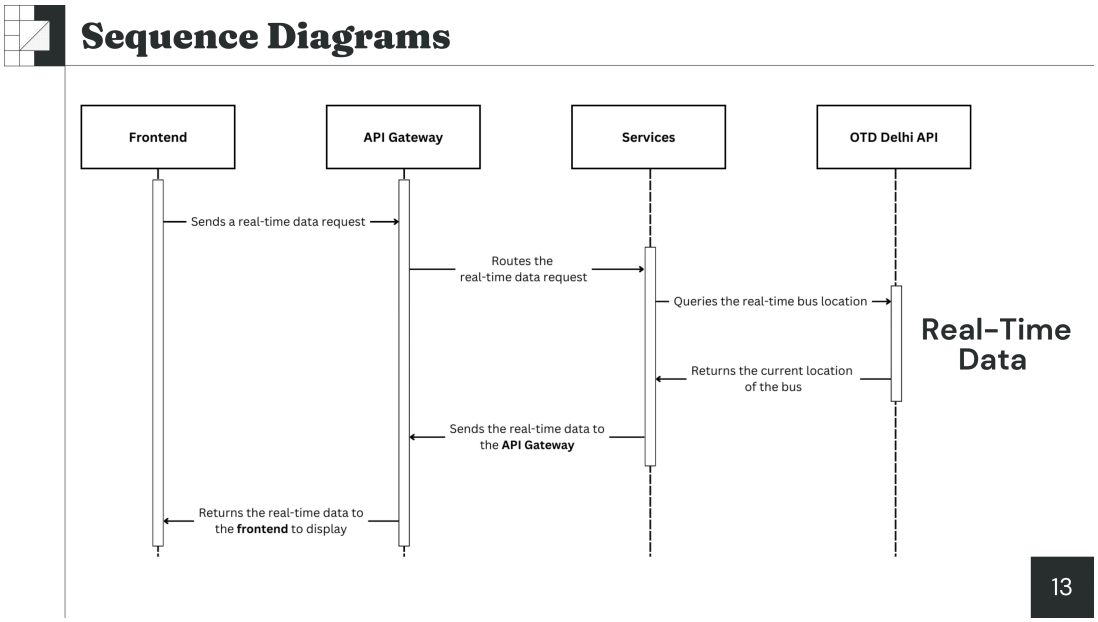


Figure 5.13: Slide 13

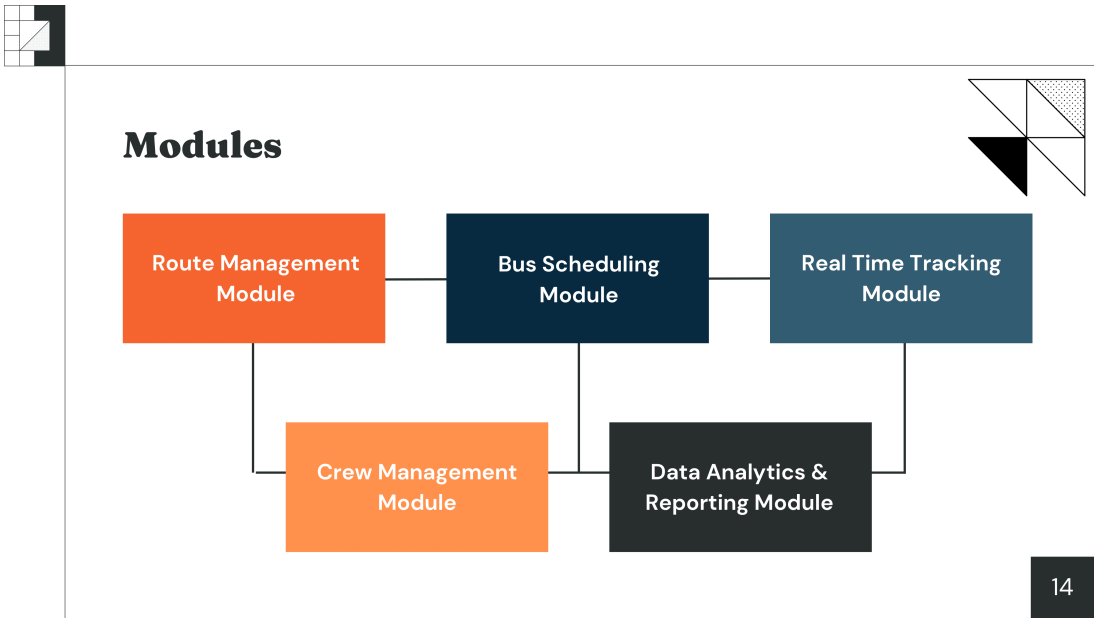


Figure 5.14: Slide 14

Route Management Module:

- Manage existing routes and allow for the creation of new routes based on real-time data such as traffic congestion.
- Optimize routes to avoid overlaps, ensuring more effective and efficient service.
- Utilize GIS technology and real-time data from the Open Transit Data Delhi API for accurate route adjustments and planning.
- Minimize travel time and improve overall service reliability through optimized route planning.

15

Figure 5.15: Slide 15

Route Management Module

Step 1: Load Input Data

- adjacency_matrix: A matrix showing the passenger flow between bus stops.
- routes: A DataFrame containing existing routes with stop sequences.
- shortest_path_matrix: A matrix containing the shortest travel time between bus stops.
- BUS_CAPACITY: Maximum passenger capacity of a bus.
- ALPHA: Weightage given to crowding in the cost function.

16

Figure 5.16: Slide 16

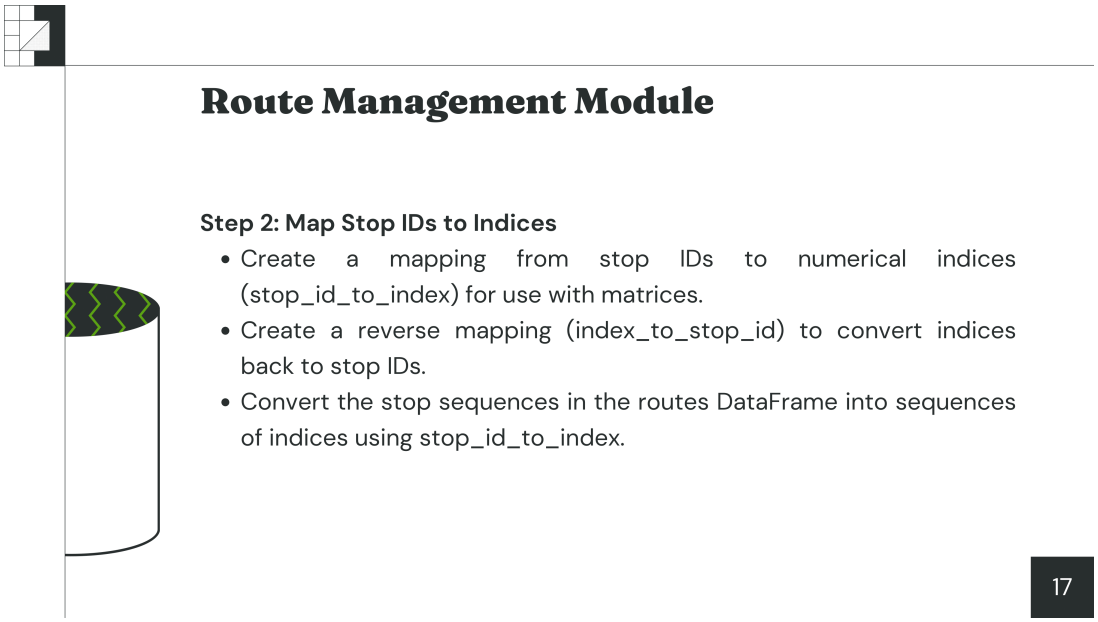


Figure 5.17: Slide 17

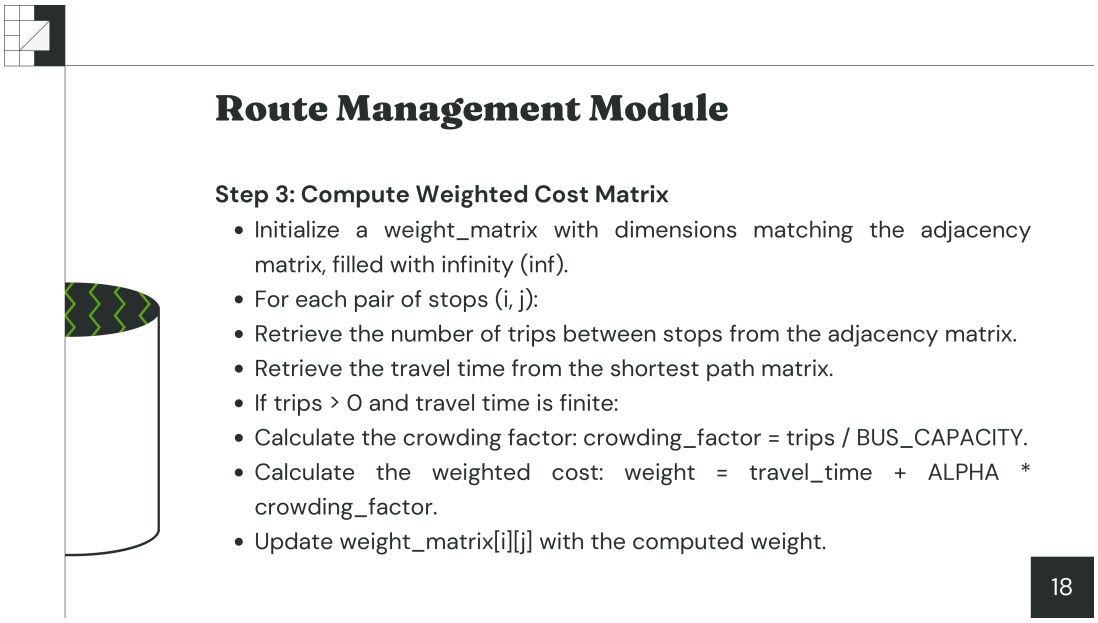


Figure 5.18: Slide 18

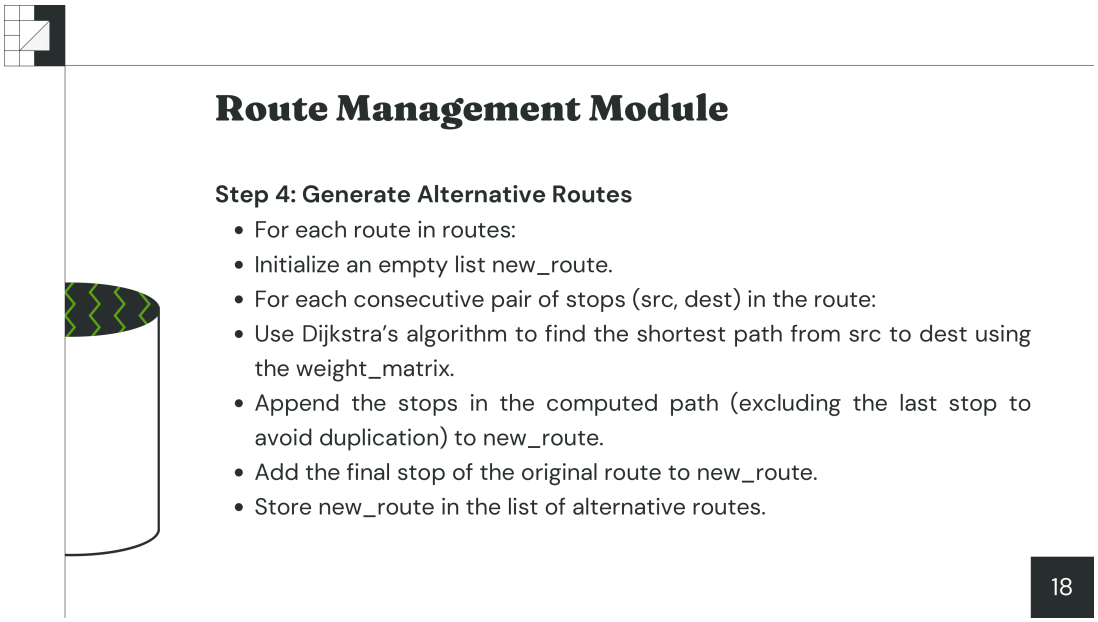


Figure 5.19: Slide 19

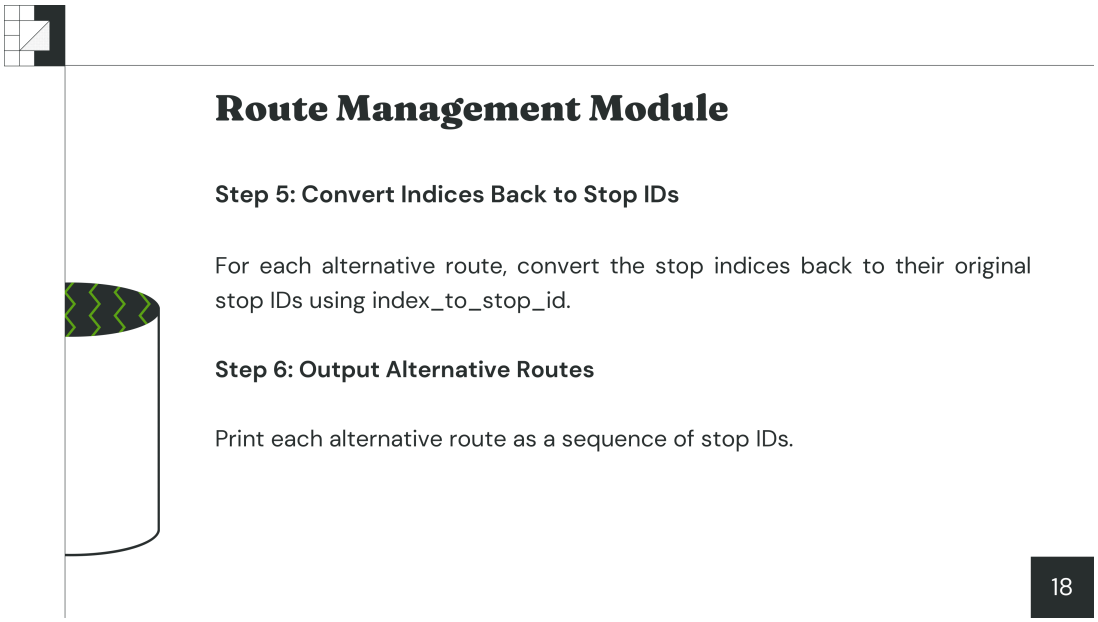


Figure 5.20: Slide 20

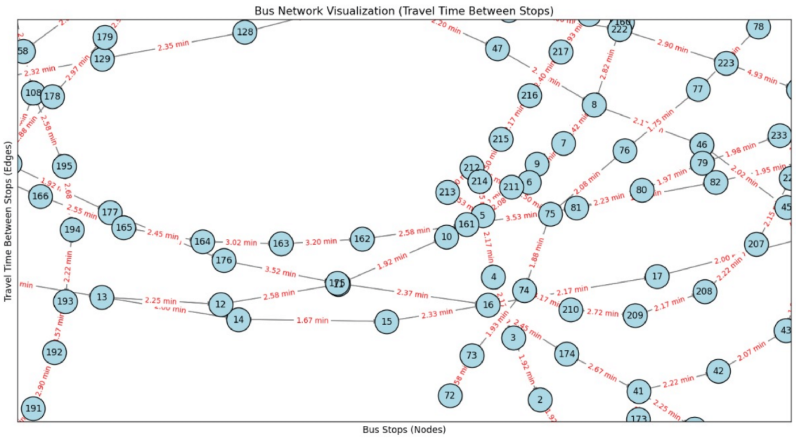


Figure 5.21: Slide 21

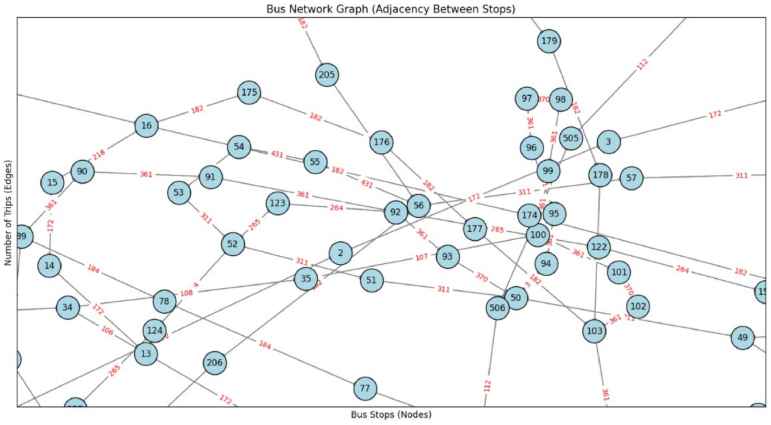


Figure 5.22: Slide 22

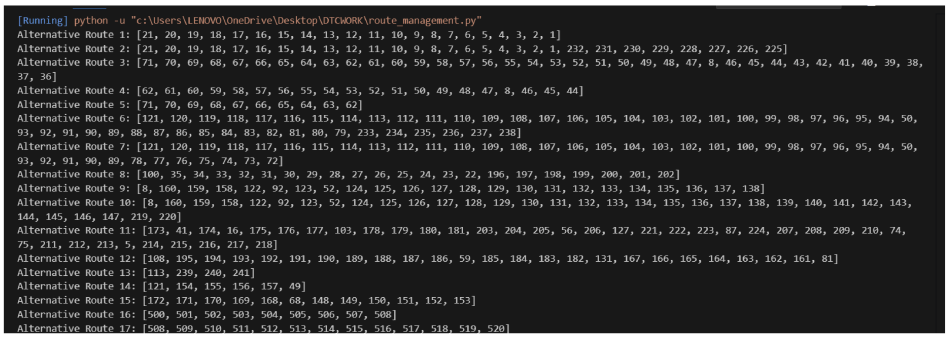
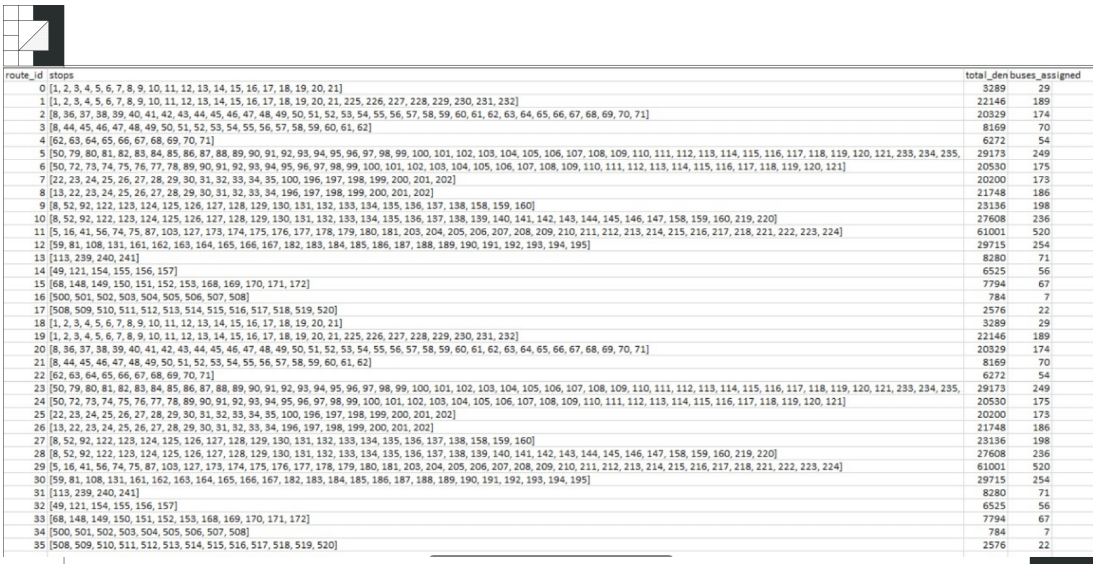
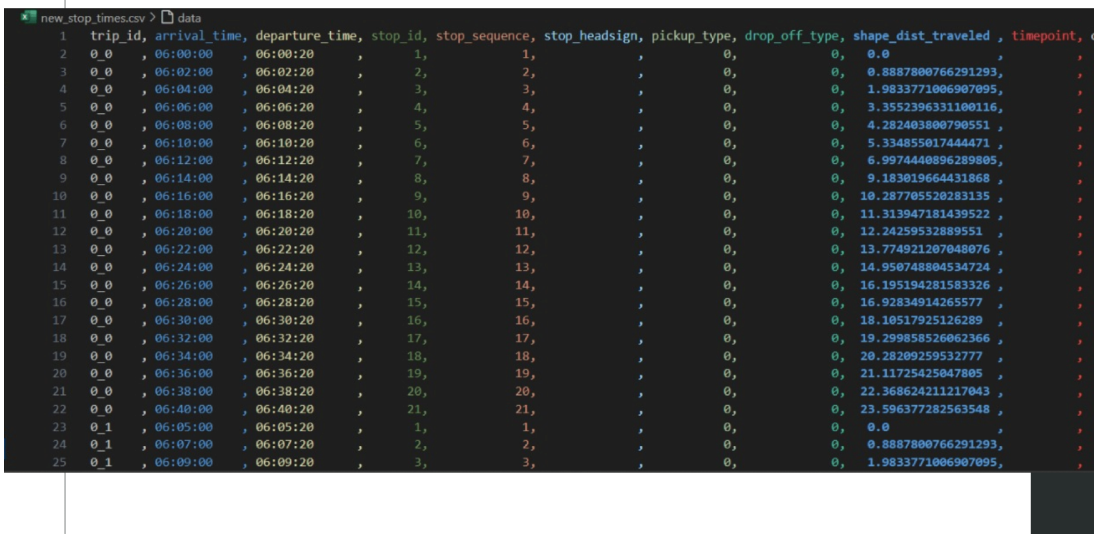


Figure 5.23: Slide 23



route_id	stops	total_den	buses_assigned
0	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]	3289	29
1	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 225, 226, 227, 228, 229, 230, 231, 232]	22146	189
2	[8, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71]	20329	174
3	[8, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	8169	70
4	[62, 63, 64, 65, 66, 67, 68, 69, 70, 71]	6272	54
5	[50, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 233, 234, 235,	29173	249
6	[50, 72, 73, 74, 75, 76, 77, 78, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121]	20530	175
7	[22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 100, 196, 197, 198, 199, 200, 201, 202]	20200	173
8	[13, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 196, 197, 198, 199, 200, 201, 202]	21748	186
9	[8, 52, 92, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 158, 159, 160]	23136	198
10	[8, 52, 92, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 158, 159, 160, 219, 220]	27608	236
11	[5, 16, 41, 56, 74, 75, 87, 103, 127, 173, 174, 175, 176, 177, 178, 179, 180, 181, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 221, 222, 223, 224]	61001	520
12	[59, 81, 108, 131, 161, 162, 163, 164, 165, 166, 167, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195]	29715	254
13	[113, 239, 240, 241]	8280	71
14	[49, 121, 154, 155, 156, 157]	6525	56
15	[68, 148, 149, 150, 151, 152, 153, 168, 169, 170, 171, 172]	7794	67
16	[500, 501, 502, 503, 504, 505, 506, 507, 508]	784	7
17	[508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520]	2576	22
18	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]	3289	29
19	[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 225, 226, 227, 228, 229, 230, 231, 232]	22146	189
20	[8, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71]	20329	174
21	[8, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62]	8169	70
22	[62, 63, 64, 65, 66, 67, 68, 69, 70, 71]	6272	54
23	[50, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 233, 234, 235,	29173	249
24	[50, 72, 73, 74, 75, 76, 77, 78, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121]	20530	175
25	[22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 100, 196, 197, 198, 199, 200, 201, 202]	20200	173
26	[13, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 196, 197, 198, 199, 200, 201, 202]	21748	186
27	[8, 52, 92, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 158, 159, 160]	23136	198
28	[8, 52, 92, 122, 123, 124, 125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141, 142, 143, 144, 145, 146, 147, 158, 159, 160, 219, 220]	27608	236
29	[5, 16, 41, 56, 74, 75, 87, 103, 127, 173, 174, 175, 176, 177, 178, 179, 180, 181, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 217, 218, 221, 222, 223, 224]	61001	520
30	[59, 81, 108, 131, 161, 162, 163, 164, 165, 166, 167, 182, 183, 184, 185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 195]	29715	254
31	[113, 239, 240, 241]	8280	71
32	[49, 121, 154, 155, 156, 157]	6525	56
33	[68, 148, 149, 150, 151, 152, 153, 168, 169, 170, 171, 172]	7794	67
34	[500, 501, 502, 503, 504, 505, 506, 507, 508]	784	7
35	[508, 509, 510, 511, 512, 513, 514, 515, 516, 517, 518, 519, 520]	2576	22

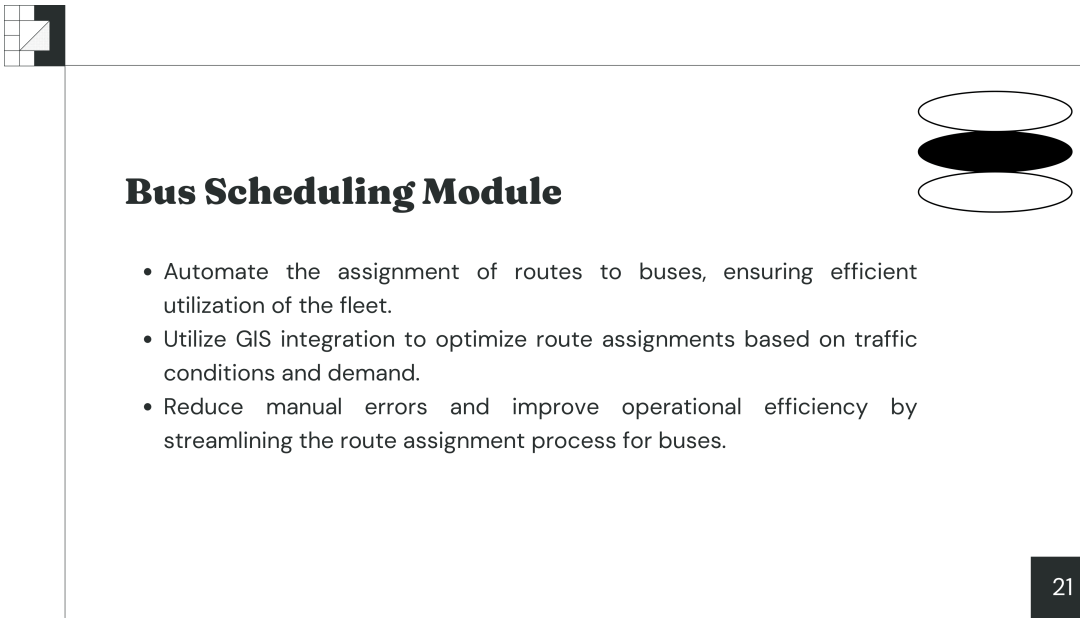
Figure 5.24: Slide 24



new_stop_times.csv > data

	trip_id	arrival_time	departure_time	stop_id	stop_sequence	stop_headsign	pickup_type	drop_off_type	shape_dist_traveled	timepoint
1	0_0	06:00:00	06:00:20	1	1		0	0	0.0	
2	0_0	06:02:00	06:02:20	2	2		0	0	0.8887800766291293	
3	0_0	06:04:00	06:04:20	3	3		0	0	1.9833771006907095	
4	0_0	06:06:00	06:06:20	4	4		0	0	3.3552396331100116	
5	0_0	06:08:00	06:08:20	5	5		0	0	4.282403800790551	
6	0_0	06:10:00	06:10:20	6	6		0	0	5.334855017444471	
7	0_0	06:12:00	06:12:20	7	7		0	0	6.9974440896289805	
8	0_0	06:14:00	06:14:20	8	8		0	0	9.183019664431868	
9	0_0	06:16:00	06:16:20	9	9		0	0	10.287705520283135	
10	0_0	06:18:00	06:18:20	10	10		0	0	11.313947181439522	
11	0_0	06:20:00	06:20:20	11	11		0	0	12.24259532889551	
12	0_0	06:22:00	06:22:20	12	12		0	0	13.774921207048076	
13	0_0	06:24:00	06:24:20	13	13		0	0	14.950748804534724	
14	0_0	06:26:00	06:26:20	14	14		0	0	16.195194281583326	
15	0_0	06:28:00	06:28:20	15	15		0	0	16.92834914265577	
16	0_0	06:30:00	06:30:20	16	16		0	0	18.10517925126289	
17	0_0	06:32:00	06:32:20	17	17		0	0	19.299858526062366	
18	0_0	06:34:00	06:34:20	18	18		0	0	20.28209259532777	
19	0_0	06:36:00	06:36:20	19	19		0	0	21.11725425047805	
20	0_0	06:38:00	06:38:20	20	20		0	0	22.368624211217043	
21	0_0	06:40:00	06:40:20	21	21		0	0	23.596377282563548	
22	0_1	06:05:00	06:05:20	1	1		0	0	0.0	
23	0_1	06:07:00	06:07:20	2	2		0	0	0.8887800766291293	
24	0_1	06:09:00	06:09:20	3	3		0	0	1.9833771006907095	

Figure 5.25: Slide 25



Bus Scheduling Module

- Automate the assignment of routes to buses, ensuring efficient utilization of the fleet.
- Utilize GIS integration to optimize route assignments based on traffic conditions and demand.
- Reduce manual errors and improve operational efficiency by streamlining the route assignment process for buses.

21

Figure 5.26: Slide 26

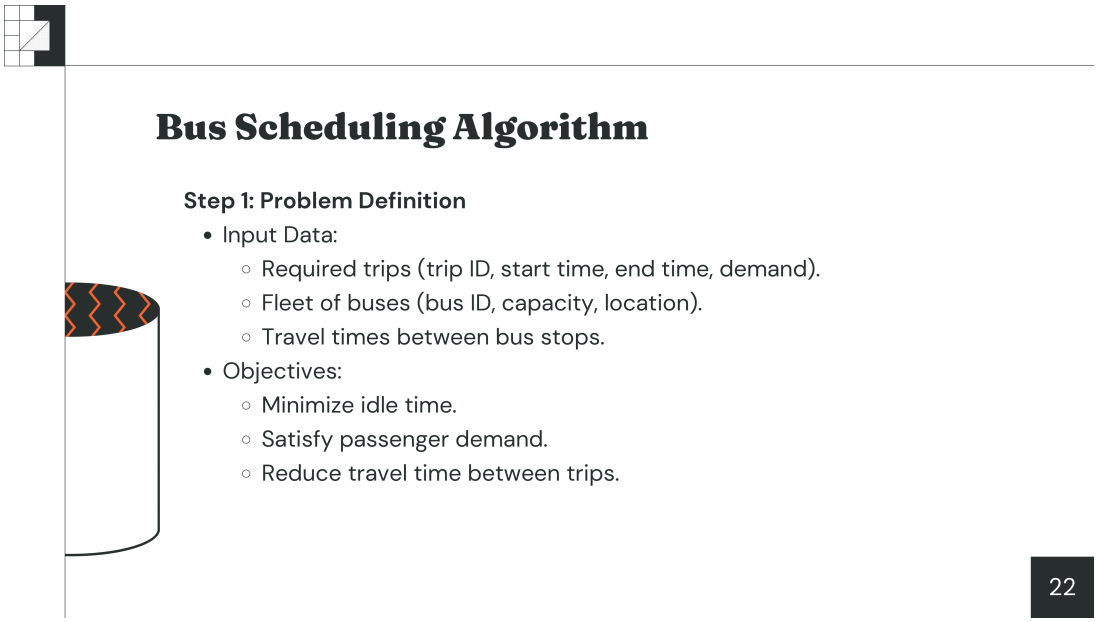


Figure 5.27: Slide 27

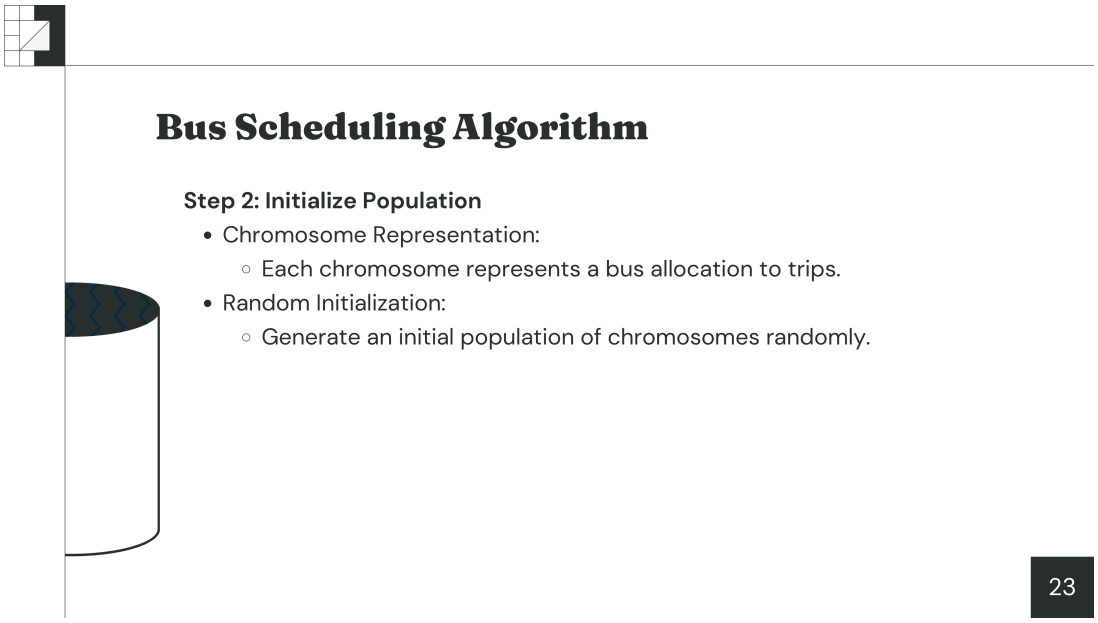


Figure 5.28: Slide 28

Bus Scheduling Algorithm

Step 3: Fitness Function

Calculate Fitness: Evaluate each chromosome based on:

- Bus Utilization
- Average Trips per Bus
- Efficiency Score
- Overlaps

$$\text{efficiency\_score} = \frac{\text{total\_distance}}{(\text{total\_idle\_time} + 1)}$$

$$\text{fitness\_value} = 0.4 \times \text{bus\_utilization} + \frac{0.3}{\text{avg\_trips\_per\_bus} + 1} + \frac{0.2}{\text{efficiency\_score} + 1} + \frac{0.1 \times \text{overlaps}}{\text{total\_trips} + 1}$$

24

Figure 5.29: Slide 29

Bus Scheduling Algorithm

Step 4: Selection

Select Parents:

- Use methods like tournament selection to choose chromosomes based on fitness scores.

Step 5: Crossover

Combine Parents:

- Exchange bus assignments between two parent chromosomes at selected points.

Step 6: Mutation

Introduce Randomness:

- Randomly change bus assignments in offspring to ensure diversity

25

Figure 5.30: Slide 30

Bus Scheduling Algorithm

Step 7: Replacement

Update Population:

- Replace the least fit chromosomes with newly created offspring.

Step 8: Termination Condition

Check for Termination:

- Repeat Steps 4-7 until a fixed number of generations is reached or fitness improvement is minimal.

Step 9: Result Extraction

- Best Solution: Identify the chromosome with the highest fitness score as the optimal bus allocation.
- Output: Provide bus assignments and metrics (idle time, demand satisfaction, travel times).

26

Figure 5.31: Slide 31

Crew Management Module

- Automate linked and unlinked duty scheduling for drivers and conductors, ensuring compliance with shift regulations.
- Track performance metrics like punctuality and shift compliance while supporting seamless crew handovers and rest period tracking.
- Align crew availability with bus schedules, optimizing crew allocation and improving productivity.

27

Figure 5.32: Slide 32

Real-Time data Integeration

- Integrate real-time bus tracking to provide dynamic updates on bus locations and conditions.
- Utilize crowdsourced data about bus crowdedness to enhance resource allocation and improve decision-making.
- Ensure continuous real-time communication using WebSocket and GPS data integration APIs.

28

Figure 5.33: Slide 33

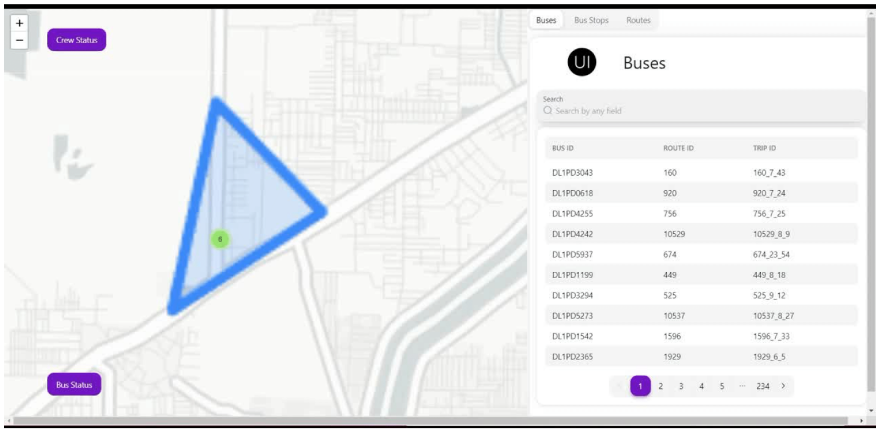


Figure 5.34: Slide 34

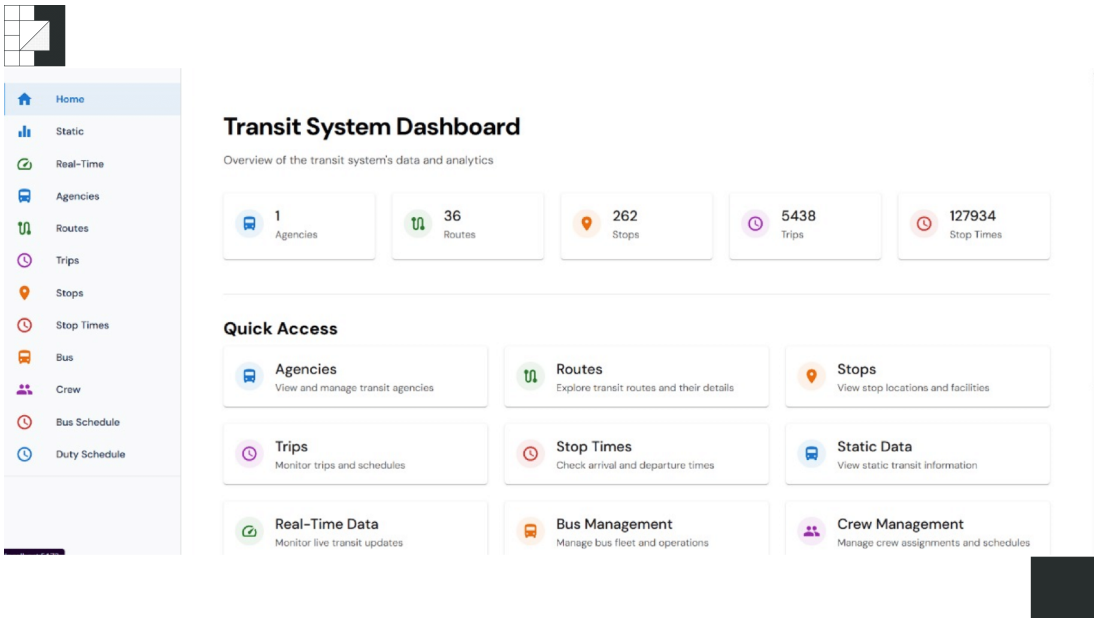


Figure 5.35: Slide 35

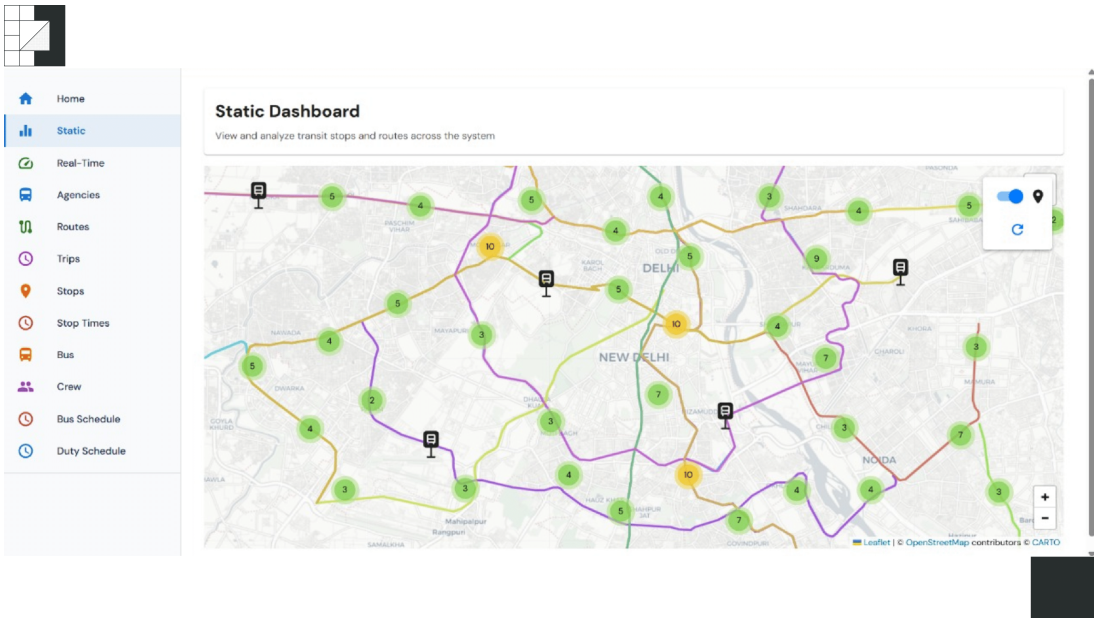


Figure 5.36: Slide 36

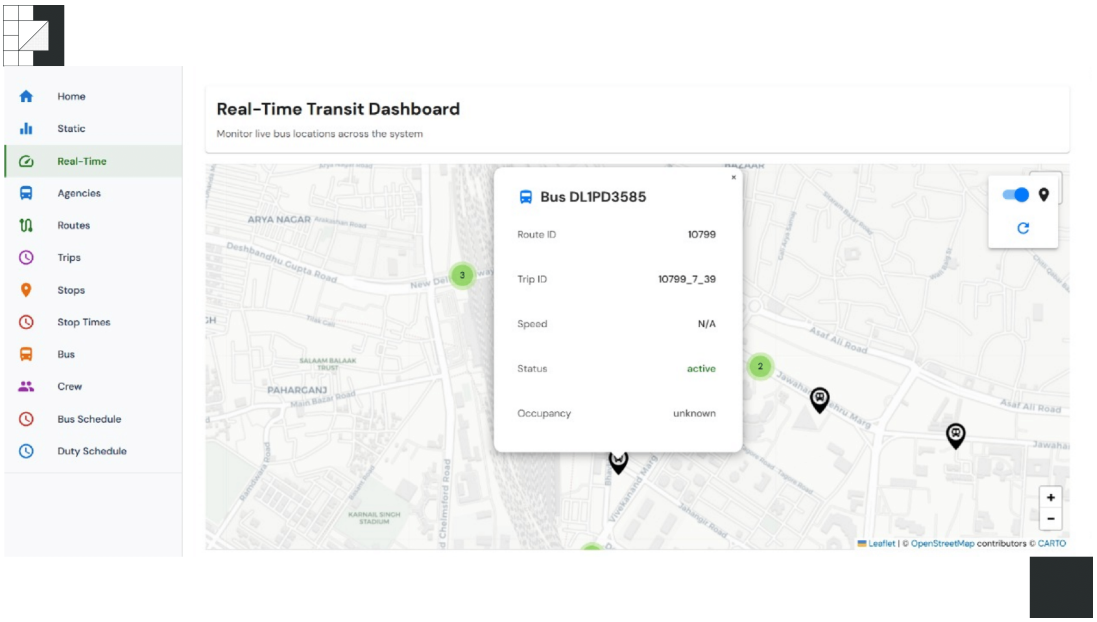


Figure 5.37: Slide 37

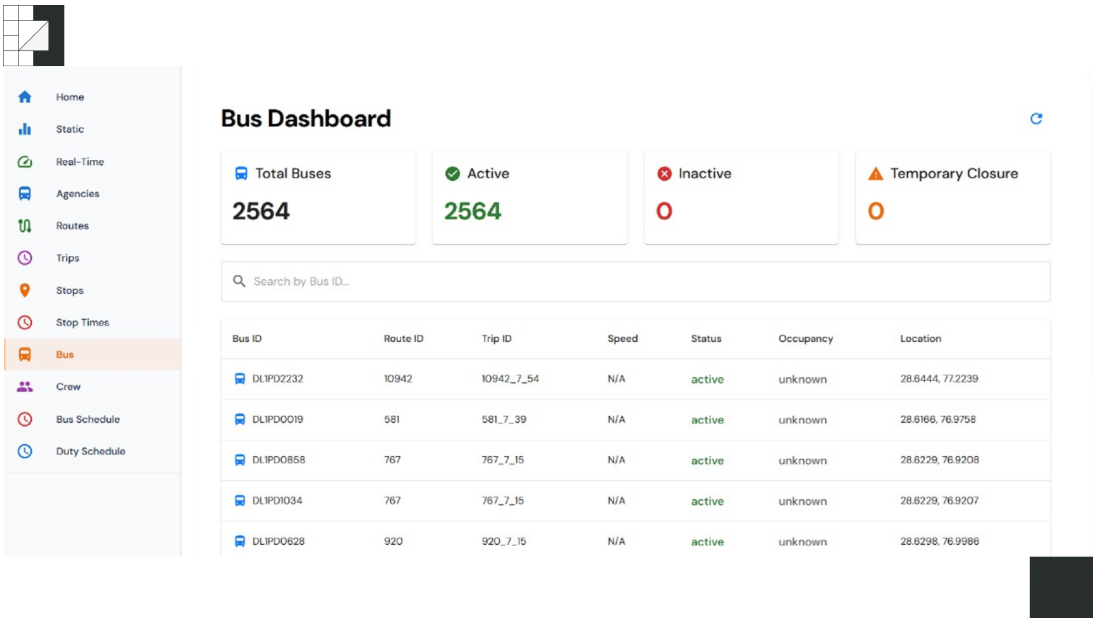


Figure 5.38: Slide 38

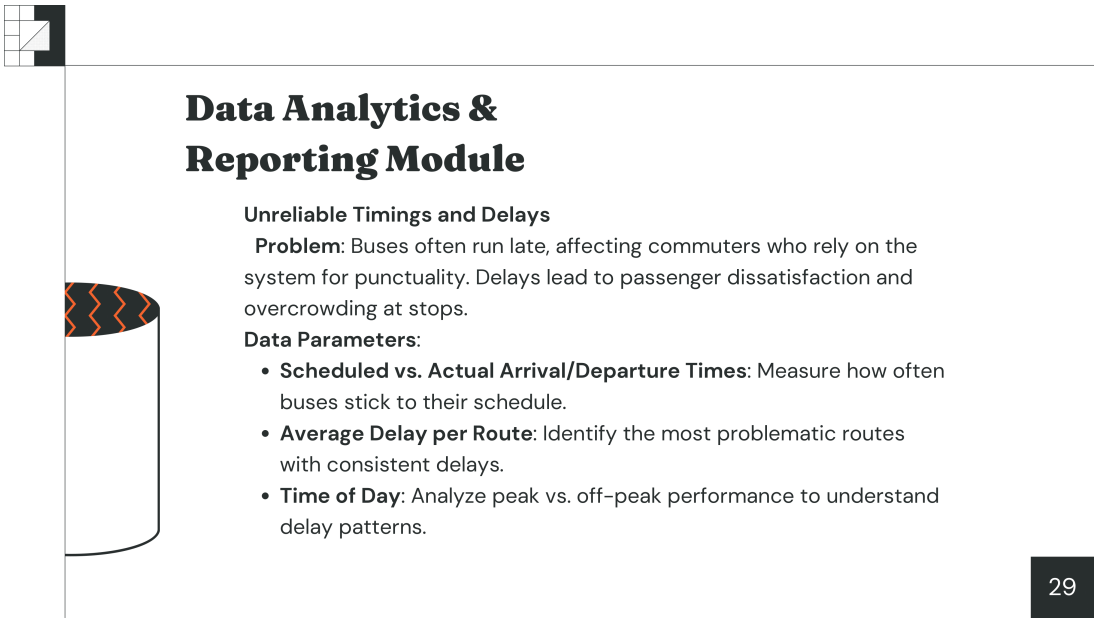


Figure 5.39: Slide 39

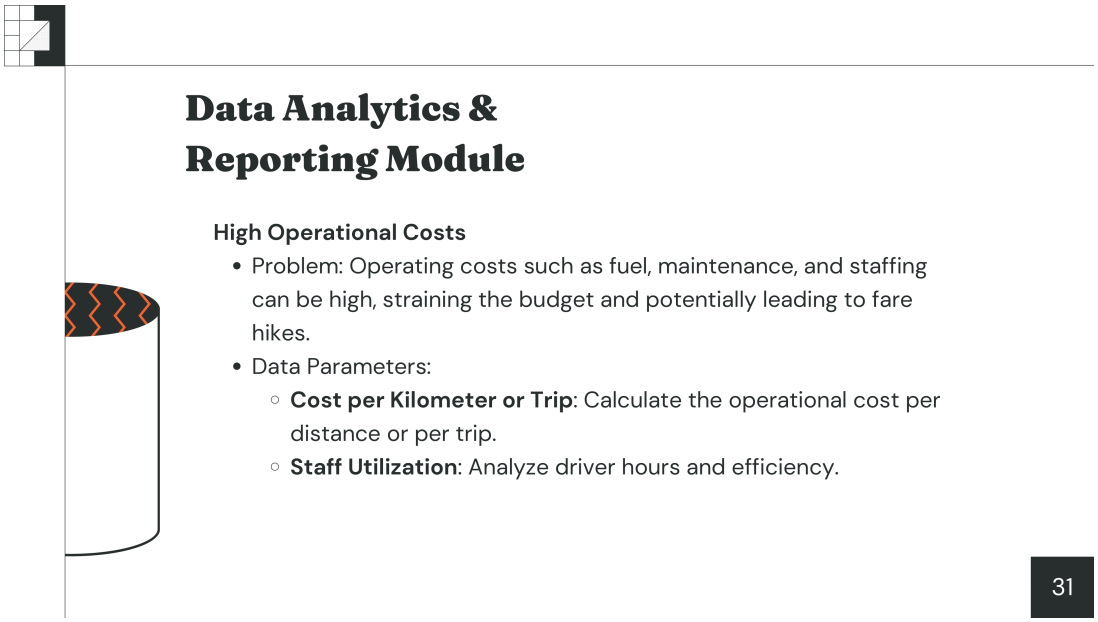


Figure 5.40: Slide 40

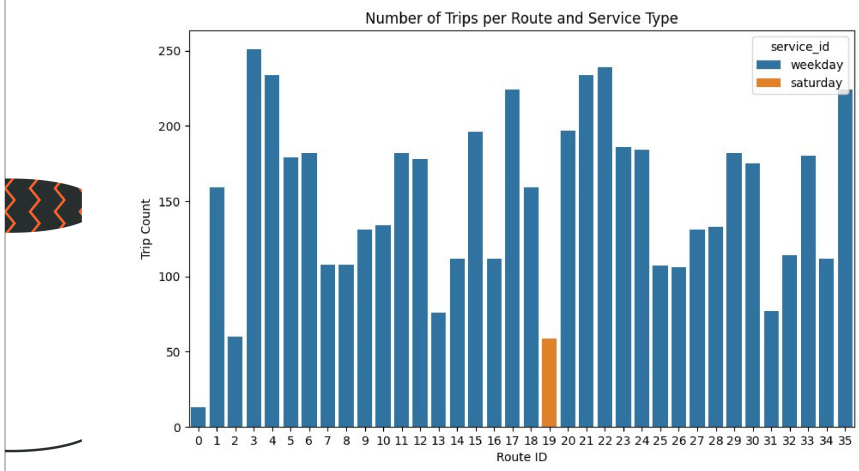


Figure 5.41: Slide 41

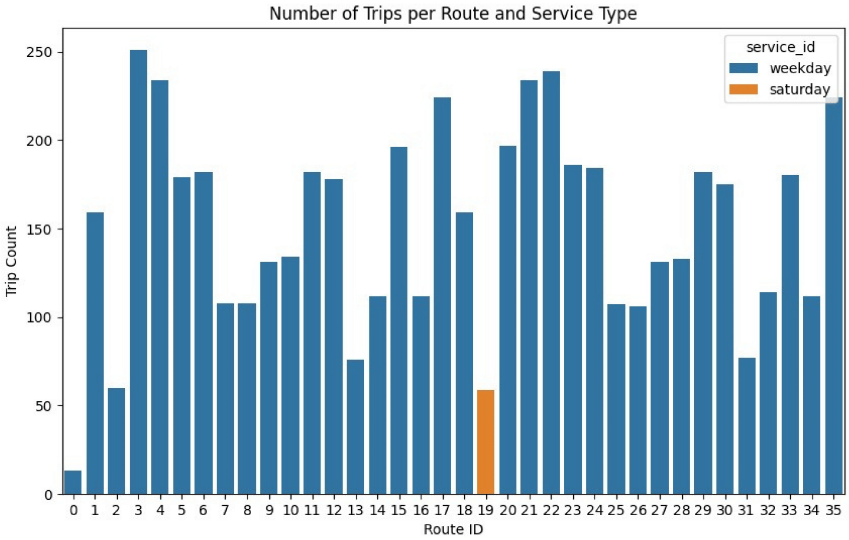


Figure 5.42: Slide 42

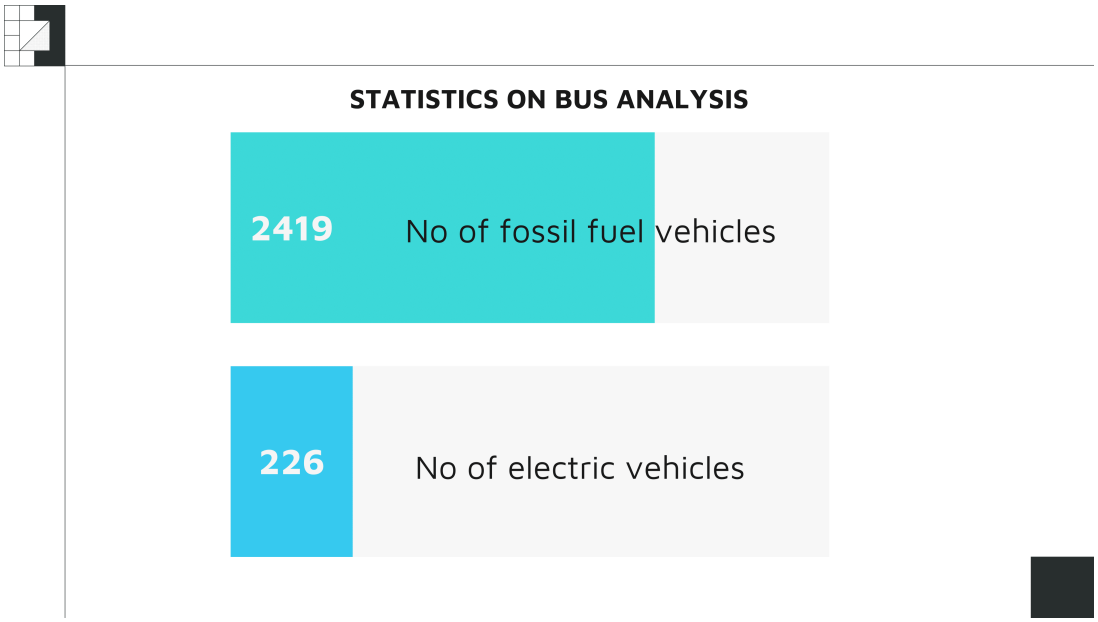


Figure 5.43: Slide 43

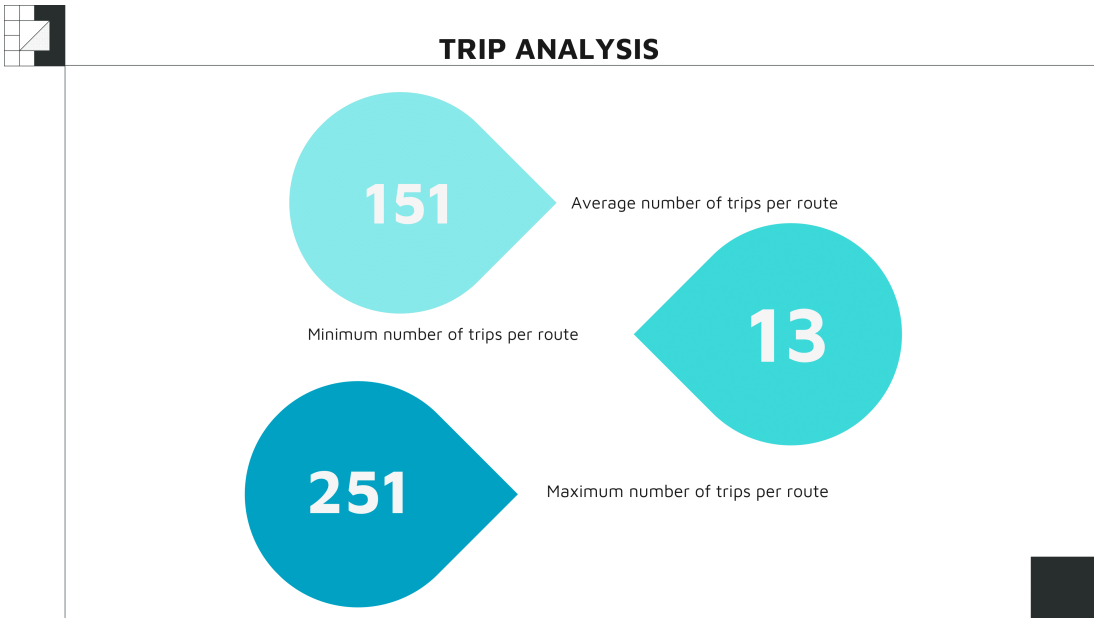


Figure 5.44: Slide 44

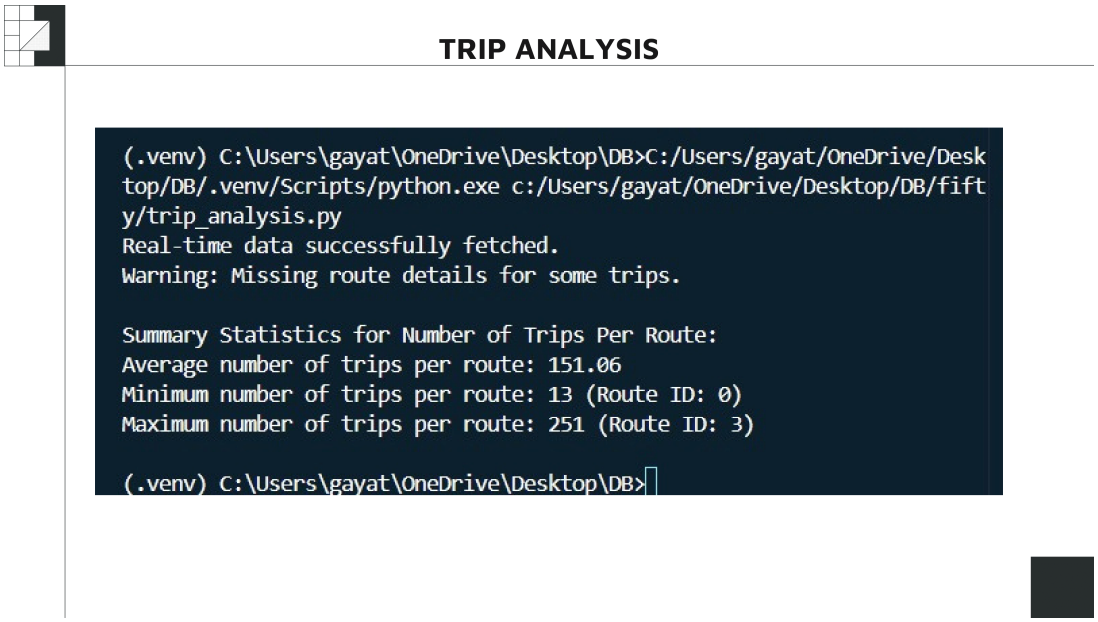


Figure 5.45: Slide 45

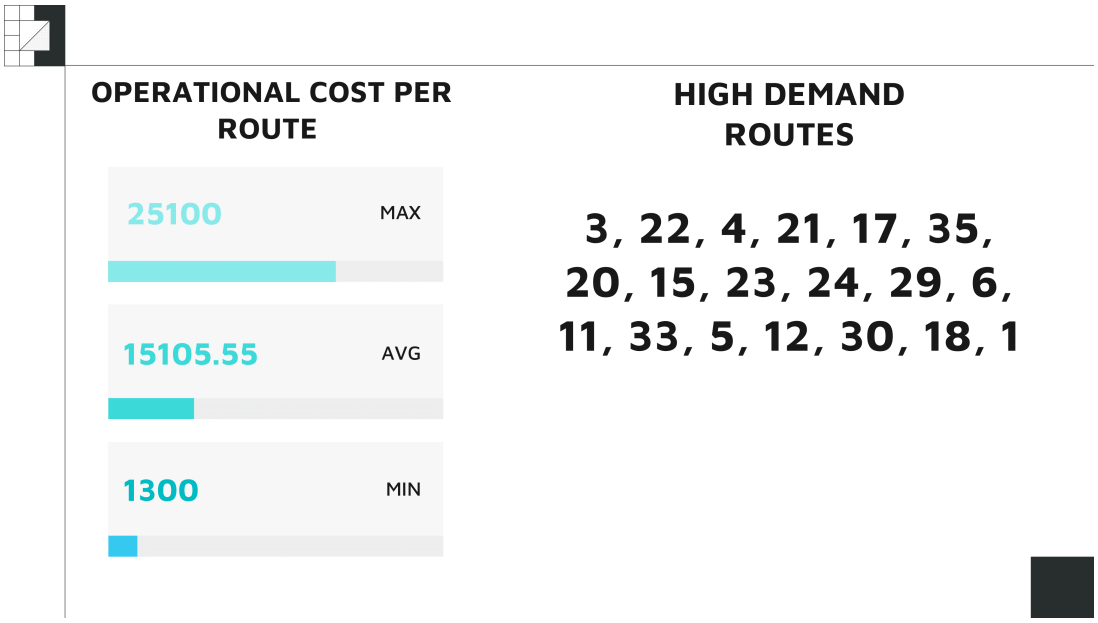


Figure 5.46: Slide 46

```
(.venv) C:\Users\gayat\OneDrive\Desktop\DB>C:/Users/gayat/OneDriv
e/Desktop/DB/.venv/Scripts/python.exe c:/Users/gayat/OneDrive/Des
ktop/DB/fifty/Routes.py
Total number of routes: 2403
Average number of trips per route: 151.05555555555554
Minimum number of trips per route: 13
Maximum number of trips per route: 251
High demand routes: [3, 22, 4, 21, 17, 35, 20, 15, 23, 24, 29, 6,
 11, 33, 5, 12, 30, 18, 1]
Average operational cost per route: 15105.555555555555
Minimum operational cost per route: 1300
Maximum operational cost per route: 25100
```

Figure 5.47: Slide 47

BUS STOP DENSITY :

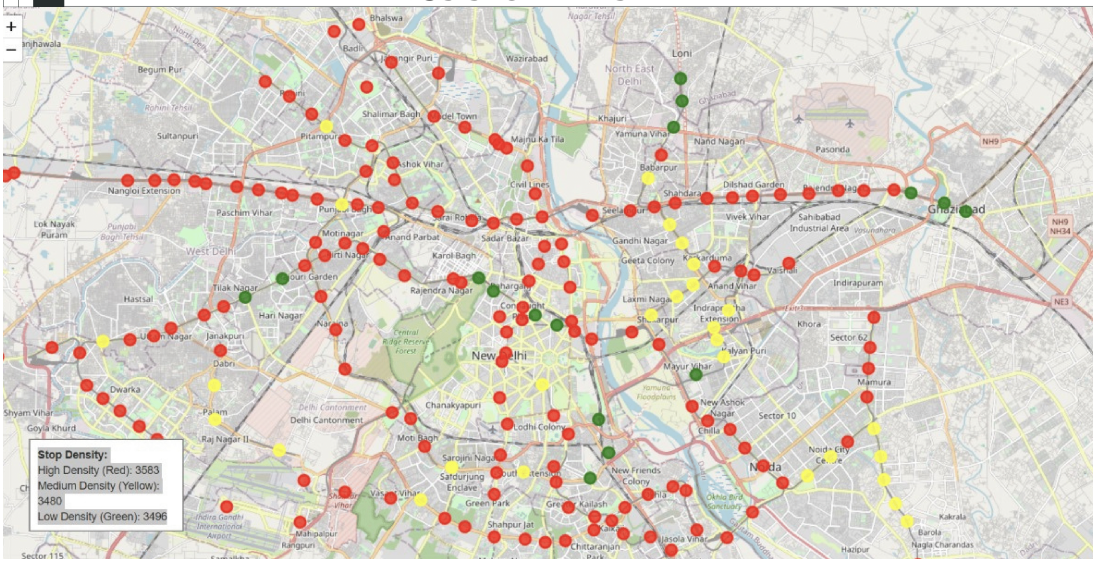


Figure 5.48: Slide 48

BUS STOP DENSITY :

```
(.venv) C:\Users\gayat\OneDrive\Desktop\DB>C:/Users/gayat/OneDriv
e/Desktop/DB/.venv/Scripts/python.exe c:/Users/gayat/OneDrive/Des
ktop/DB/fifty/density_busStop.py
High Density (Red): 3583
Medium Density (Yellow): 3480
Low Density (Green): 3496
```

Figure 5.49: Slide 49

ALL DIFFERENT ROUTES PLOTTED

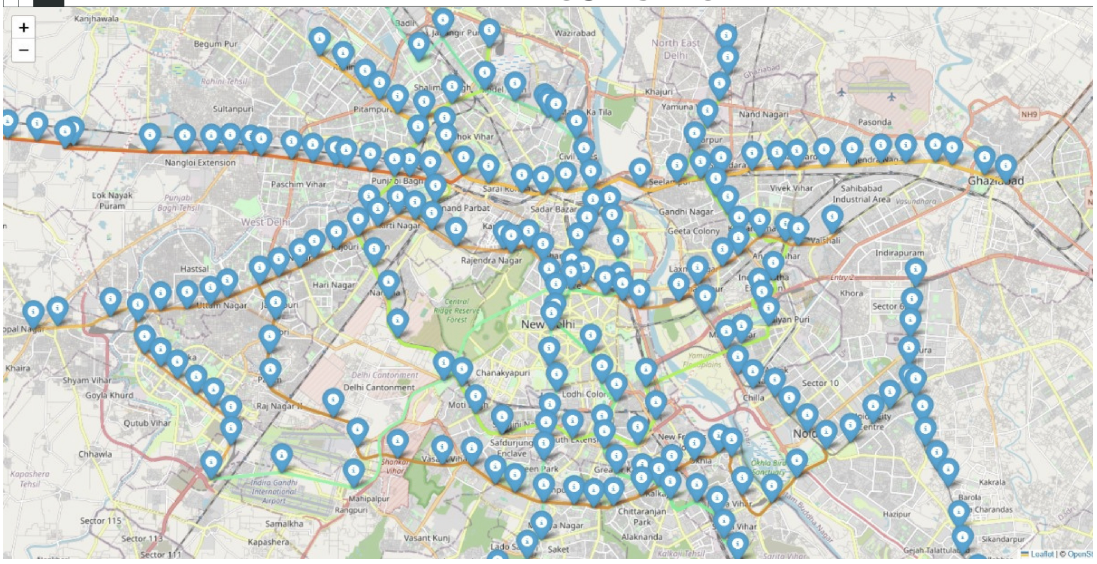


Figure 5.50: Slide 50

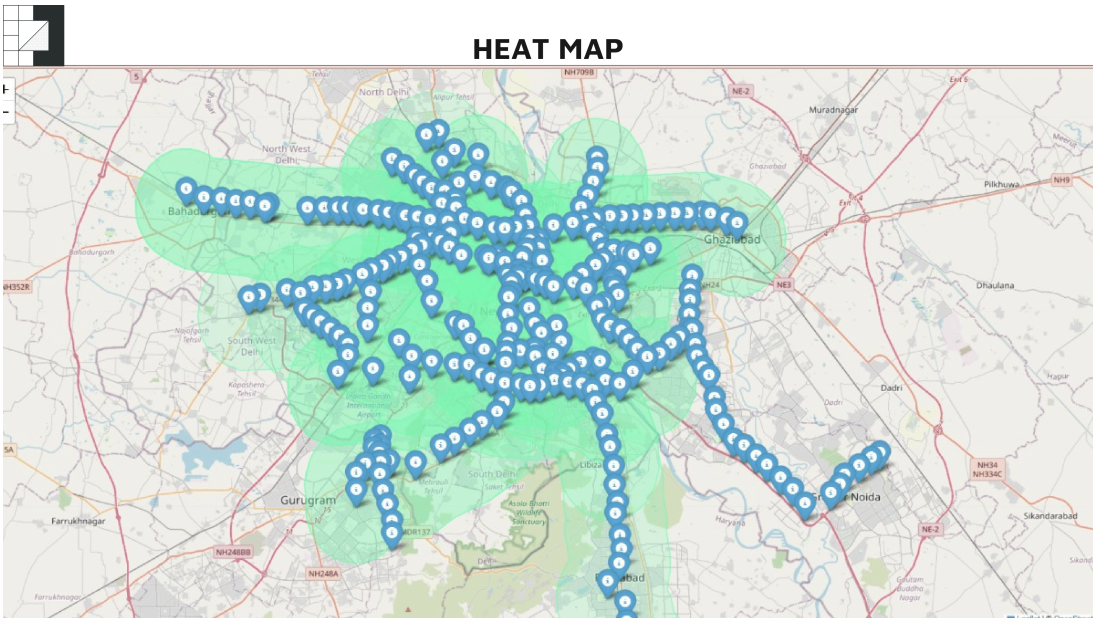


Figure 5.51: Slide 51

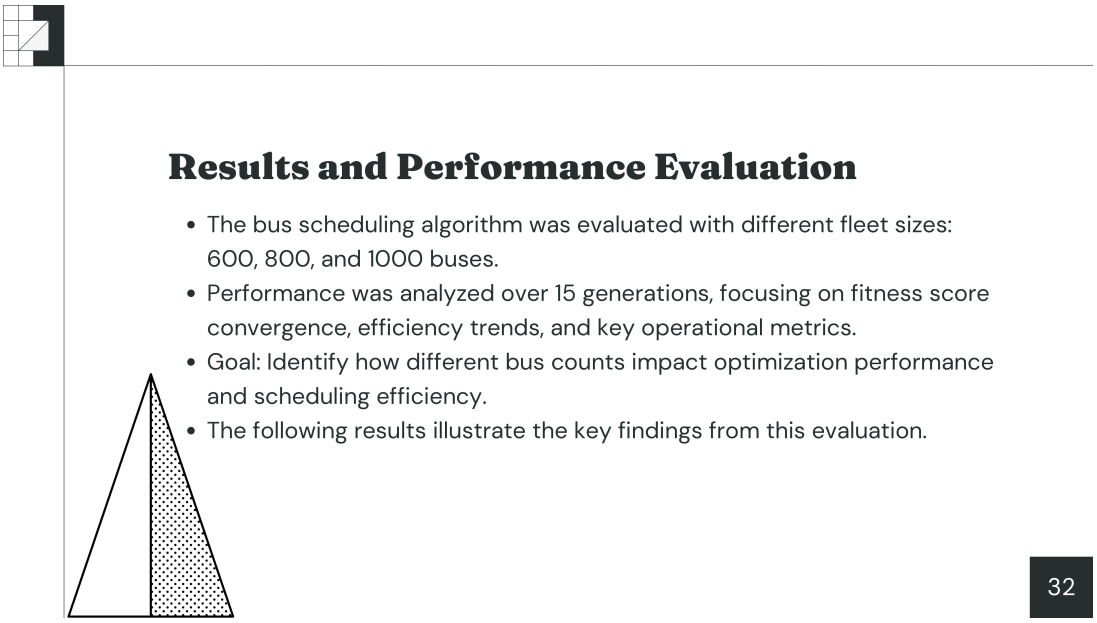


Figure 5.52: Slide 52

Fitness Score Evolution Over Generations

- The **fitness score** is a key indicator of optimization quality, where a lower value represents a better scheduling solution.
- Early generations show a rapid drop in fitness score, indicating strong initial improvements.
- The rate of improvement slows down after Generation 6, as the system approaches stability.

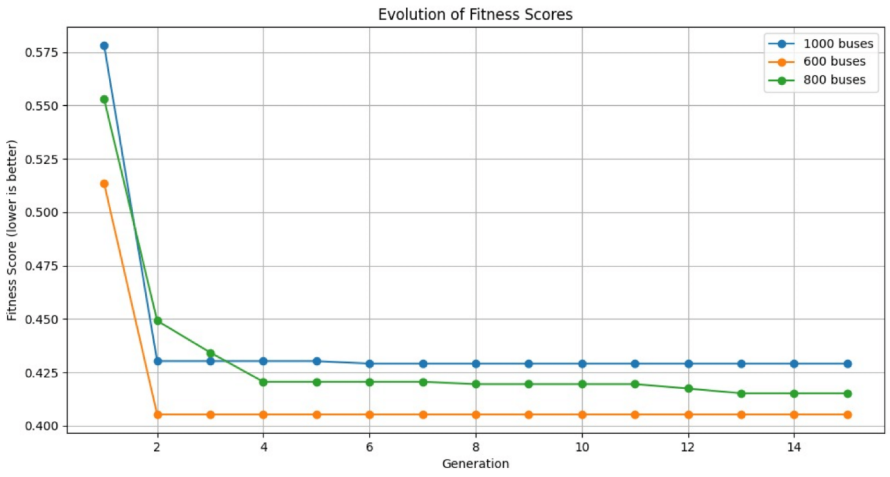
Among all configurations:

- 600-bus fleet achieves the best optimization (lowest fitness score).
- 800-bus and 1000-bus fleets also improve but at a slower rate.

This trend suggests that a smaller fleet may lead to a more efficient schedule under the given constraints.

33

Figure 5.53: Slide 53



33

Figure 5.54: Slide 54

Efficiency Score Trends Over Generations

- The efficiency score quantifies how well the bus network is utilized, with a higher score indicating better efficiency.
- All configurations show a gradual rise in efficiency during the first few generations.

Key Observations:

- The 1000-bus fleet achieves the highest efficiency plateau, meaning it can serve more trips optimally.
- 800 buses follow closely, while 600 buses show a lower efficiency ceiling.
- Diminishing returns appear after Generation 10, suggesting that adding more buses has limited efficiency gains beyond a certain point.
- This result highlights the trade-off between fleet size and efficiency, where increasing buses does improve efficiency but not proportionally.

Figure 5.55: Slide 55

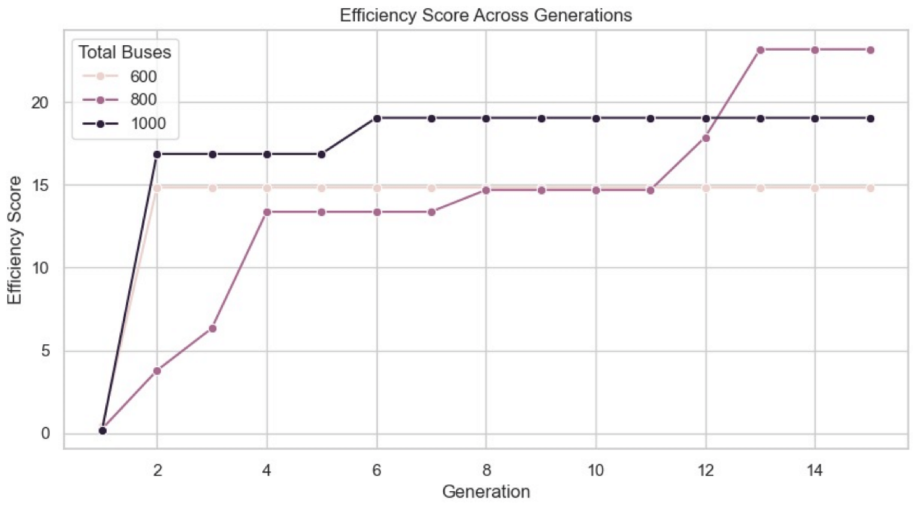


Figure 5.56: Slide 56

Comparative Performance Across Metrics

- To better understand performance differences, the following three key metrics were analyzed:
- Average Trips Per Bus → Measures how many trips each bus completes.
- Efficiency Score → Quantifies scheduling effectiveness.
- Overlaps → Measures redundancy in routes.

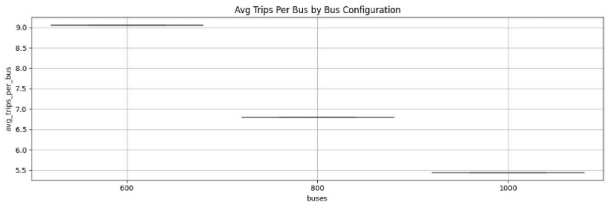


Figure 5.57: Slide 57

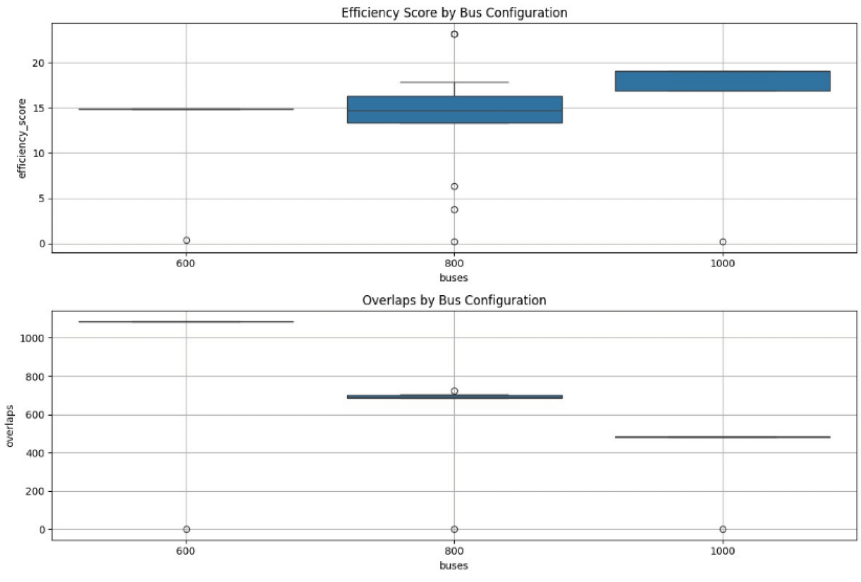


Figure 5.58: Slide 58

Comparative Performance Across Metrics

Average Trips Per Bus:

- 600-bus fleet has the highest trips per bus, indicating higher individual bus utilization.
- 1000-bus fleet has the lowest, meaning more buses reduce individual workload.

Efficiency Score:

- 1000-bus fleet achieves the highest efficiency, followed by 800 buses.
- 600 buses have lower variability but also lower efficiency.

Overlaps in Scheduling:

- 600 buses have the highest overlap, meaning buses cover similar routes more frequently.
- 1000 buses show the least redundancy, leading to better route distribution.

Figure 5.59: Slide 59

Optimization Algorithm Performance Across Fleet Sizes

- The bus scheduling algorithm was tested with 600, 800, and 1000 buses over 15 generations.
- The goal was to evaluate how the optimization algorithm improves scheduling efficiency over time

600 Buses:

- Initial fitness score: 0.6874 → improves to 0.4767 by Generation 6.
- Final fitness score: 0.4767 at Generation 15 (indicating early convergence).
- Strong early performance, suggesting that smaller fleets optimize quickly.

Figure 5.60: Slide 60

Optimization Algorithm Performance Across Fleet Sizes

800 Buses:

- Initial fitness score: 0.6412 → improves to 0.4692 by Generation 6.
- Final fitness score: 0.4692 at Generation 15.
- Shows consistent optimization, proving the algorithm is effective across different fleet sizes.

1000 Buses:

- Initial fitness score: 0.6145 → improves to 0.4666 by Generation 2.
- Final fitness score: 0.4655 at Generation 15.
- Even with a larger problem space, the algorithm continues to fine-tune results effectively.

Figure 5.61: Slide 61

Assumptions

- **Real-Time Data Accuracy**: Assumes GPS and crowdsourced data are consistently accurate for reliable tracking and decision-making.
- **Data Availability:** DTC's operational data will be accessible; otherwise, external APIs or manual input will be required.
- **User Adoption:** Assumes users will adapt with adequate training, as resistance could hinder the system's effectiveness.
- **Scalability:** The system can scale with growing demand, assuming sufficient hardware and software resources.

33

Figure 5.62: Slide 62

Work Breakdown and Responsibilities

- Route Management → Fabin
- Bus & Crew Scheduling → Elviin
- Geographical Visualization → Kuruvilla
- UI/UX and Data Analysis → Gayathri

34

Figure 5.63: Slide 63

Hardware & Software Requirements

Hardware

Processor: Intel Core i5 or higher
RAM: 8 GB
Storage: 256 GB SSD
Graphics Card: NVIDIA GeForce GTX 1650 (4 GB GPU)
Network: High-speed Ethernet connection for real-time data communication
Display: Full HD monitor (1920x1080)

Software

OS: Windows 10 64-bit or Ubuntu 20.04 LTS
Backend: Node.js (Express)
Database: MongoDB
Frontend: React.js for user interface
Mapping: Leaflet.js with OpenSteetMaps, Openroute Service for distance/time calculation .

35

Figure 5.64: Slide 64

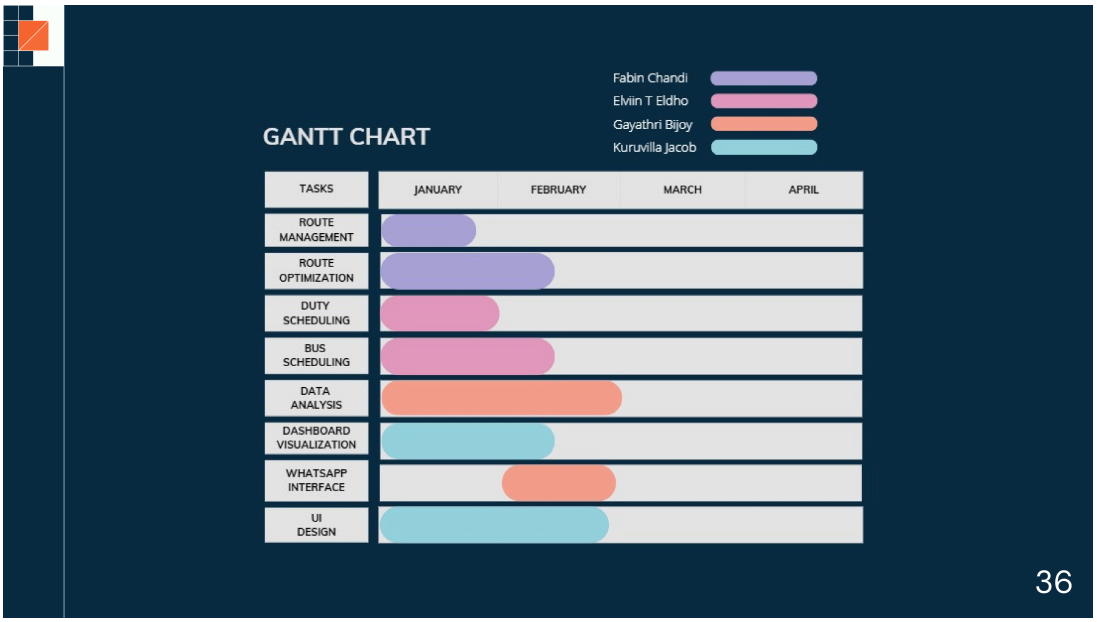


Figure 5.65: Slide 65

Risk & Challenges

Data Integration and Accuracy

Challenge: Integrating real-time (GPS, traffic) and historical data (routes, demand) from various sources can be complex and error-prone.
Risk: Inaccurate data may result in poor route optimization and scheduling.
Mitigation: Use rigorous validation and source data from multiple providers to ensure accuracy.

Real-Time System Performance

Challenge: The system must process real-time GPS data, track buses, and update dashboards continuously. Handling large-scale, real-time data streams efficiently is a technical challenge.
Risk: System performance might degrade under heavy traffic, causing delays in updates or system downtime. This could impact real-time scheduling and monitoring.

37

Figure 5.66: Slide 66

Conclusion

- Efficiency: Automates scheduling, reduces errors, and optimizes resource use.
- Real-Time Monitoring: Enables bus tracking, dynamic route adjustments, and delay management.
- Route Optimization: Uses GIS and data analytics to improve routes, reduce congestion, and enhance coverage.
- Crew Management: Automates shift scheduling, ensures legal compliance, and tracks performance.
- Scalability: Flexible system designed to scale with DTC's expanding network.
- Data Insights: Provides analytics for operational improvements and meeting passenger demand.

39

Figure 5.67: Slide 67

Citations

- Z. Li, Z. Ding, Q. Qin, H. Huang, and Y. Guo, "Research on logistics route optimization based on GPS and GIS technology," in *2017 3rd International Conference on Electronic Information Technology and Intellectualization* (ICEITI 2017), ISBN: 978-1-60595-512-4, pp. 1-6, Nov. 2017. **[1]**
- T. Ma, G. Motta, and K. Liu, "Delivering real-time information services on public transit: A framework," *IEEE Transactions on Intelligent Transportation Systems, vol. 18, no. 6, pp. 1584-1596*, Feb. 2017, doi: 10.1109/TITS.2017.2656387 **[2]**
- C. Yu, H. Li, X. Xu, J. Liu, J. Miao, Y. Wang, and Q. Sun, "Data-driven approach for passenger mobility pattern recognition using spatiotemporal embedding," *Journal of Advanced Transportation*, vol. 2021, pp. 1-21, May 2021 **[3]**

40

Figure 5.68: Slide 68

Citations

- Dib, Omar, Marie-Ange Manier, and Alexandre Caminada. "Memetic algorithm for computing shortest paths in multimodal transportation networks." *Transportation Research Procedia* 10 (2015): 745-755. **[4]**
- P. Rajput, M. Chaturvedi, and V. Patel, "Opportunistic sensing based detection of crowdedness in public transport buses," *Pervasive and Mobile Computing*, vol. 68, pp. 101246, 2020. **[5]**

Figure 5.69: Slide 69

Thank you

Figure 5.70: Slide 70

Appendix B: Vision, Mission, Programme Outcomes and Course Outcomes

Vision, Mission, Programme Outcomes and Course Outcomes

Institute Vision

To evolve into a premier technological institution, moulding eminent professionals with creative minds, innovative ideas and sound practical skill, and to shape a future where technology works for the enrichment of mankind.

Institute Mission

To impart state-of-the-art knowledge to individuals in various technological disciplines and to inculcate in them a high degree of social consciousness and human values, thereby enabling them to face the challenges of life with courage and conviction.

Department Vision

To become a centre of excellence in Computer Science and Engineering, moulding professionals catering to the research and professional needs of national and international organizations.

Department Mission

To inspire and nurture students, with up-to-date knowledge in Computer Science and Engineering, ethics, team spirit, leadership abilities, innovation and creativity to come out with solutions meeting societal needs.

Programme Outcomes (PO)

Engineering Graduates will be able to:

PO1: Engineering Knowledge: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

PO2: Problem analysis: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of

mathematics, natural sciences, and engineering sciences.

PO3: Design/development of solutions: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

PO4: Conduct investigations of complex problems: Use research-based knowledge including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

PO5: Modern Tool Usage: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

PO6: The engineer and society: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal, and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

PO7: Environment and sustainability: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

PO8: Ethics: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

PO9: Individual and Team work: Function effectively as an individual, and as a member or leader in teams, and in multidisciplinary settings.

PO10: Communication: Communicate effectively with the engineering community and with society at large. Be able to comprehend and write effective reports and documentation. Make effective presentations, and give and receive clear instructions.

PO11: Project management and finance: Demonstrate knowledge and understanding of engineering and management principles and apply these to one's own work, as a member and leader in a team. Manage projects in multidisciplinary environments.

PO12: Life-long learning: Recognize the need for, and have the preparation and ability to engage in independent and lifelong learning in the broadest context of techno-

logical change.

Programme Specific Outcomes (PSO)

PSO1: Computer Science Specific Skills
The ability to identify, analyze, and design solutions for complex engineering problems in multidisciplinary areas by understanding the core principles and concepts of computer science, thereby engaging in national grand challenges.

PSO2: Programming and Software Development Skills
The ability to acquire programming efficiency by designing algorithms and applying standard practices in software project development to deliver quality software products meeting the demands of the industry.

PSO3: Professional Skills
The ability to apply the fundamentals of computer science in competitive research and to develop innovative products to meet societal needs, thereby evolving as an eminent researcher and entrepreneur.

Course Outcomes (CO)

CO 1: Model and solve real world problems by applying knowledge across domains.

CO 2: Develop products, processes or technologies for sustainable and socially relevant applications.

CO 3: Function effectively as an individual and as a leader in diverse teams and to comprehend and execute designated tasks.

CO 4: Plan and execute tasks utilizing available resources within timelines, following ethical and professional norms.

CO 5: Identify technology/research gaps and propose innovative/creative solutions.

CO 6: Organize and communicate technical and scientific findings effectively in written and oral forms.

Appendix C: CO-PO-PSO MAPPING

	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3
CO 1	2	2	2	1	2	2	2	1	1	1	1	2	3		
CO 2	2	2	2		1	3	3	1	1		1	1		2	
CO 3									3	2	2	1			3
CO 4					2			3	2	2	3	2			3
CO 5	2	3	3	1	2							1	3		
CO 6					2			2	2	3	1	1			3

3/2/1: high/medium/low

Figure 5.71: CO-PO and CO-PSO Mapping Table