“CITY BUS PASSENGER DATA VISUALISATION TOOL”

A Project Report of Major Project Phase-II Submitted to
Rajiv Gandhi Proudyogiki Vishwavidhyalaya, Bhopal
Towards Partial Fulfillment of the Degree of
Bachelor of Technology in Computer Engineering

Guided By:

Dr Vandan Tewari
Professor
Department of Computer Engineering

Submitted By:

Amogh Mittal (0801CS201011)
Arpit Jain (0801CS201019)
Chinmay Dubey (0801CS201028)
Kanchan Hingorani (0801CS201046)
Vanshika Agrawal (0801CS201100)

Department of Computer Engineering
Shri G.S. Institute of Technology and Science, Indore(M.P.)
2023-2024

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

SESSION: 2023- 24

RECOMMENDATION

The project report for Major Project Phase-II entitled "**City Bus Passenger Data Visualisation Tool**" submitted by: **Amogh Mittal (0801CS20101), Arpit Jain (0801CS20101), Chinmay Dubey(0801CS201028), Kanchan Hingorani (0801CS201046), Vanshika Agrawal (0801CS201100)**, students of Bachelor of Technology IV year in the session 2023-24, towards fulfilment of the degree of **B.Tech. in Computer Science and Engineering** of Rajiv Gandhi Proudyogiki Vishwa Vidhyalaya, Bhopal is a satisfactory account of their work towards CO44498: Major Project (Phase II) and is recommended for the award of degree.

Dr Vandan Tewari
Professor
Project Guide
Department of Computer Engg.

Dr Vandan Tewari
Head of Department
Department of Computer Engg.

Dean (Academics)
S.G.S.I.T.S. Indore

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

(A Govt. Aided Autonomous Institute, Affiliated to RGPV, Bhopal)

DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

The project report for CO44498: Major Project Phase-II entitled "City Bus Passenger Data Visualisation Tool" submitted by: **Amogh Mittal (0801CS201011), Arpit Jain (0801CS201019), Chinmay Dubey (0801CS201028), Kanchan Hingorani (0801CS201046), Vanshika Agrawal (0801CS201100)**, students of Bachelor of Technology IV year in the session 2023-24, towards partial fulfilment of the degree of Bachelor of Technology in Computer Engineering of Rajiv Gandhi Proudyogiki VishwaVidhyalaya, Bhopal is a satisfactory account of their work.

Internal Examiner **External Examiner**

Date: **Date:**

SHRI G.S. INSTITUTE OF TECHNOLOGY AND SCIENCE, INDORE(M.P.)

DECLARATION

We **Amogh Mittal (0801CS201011), Arpit Jain (0801CS201019), Chinmay Dubey(0801CS201028), Kanchan Hingorani (0801CS201046), Vanshika Agrawal (0801CS201100)**, here by declare that work presented in the B. Tech project report has been carried out by us. We further declare that the work submitted for the award of the degree doesn't contain any part of the work which has been submitted for the award of any degree either in this university or any other university without proper citation.

Amogh Mittal (0801CS201011)

Arpit Jain (0801CS201019)

Chinmay Dubey (0801CS201028)

Kanchan Hingorani (0801CS201046)

Vanshika Agrawal (0801CS201100)

ACKNOWLEDGEMENT

We express our profound sense of gratitude to our project guide Dr. Vandan Tewari who had advised us to do project work entitled "City Bus Passenger Data Visualisation Tool ": a solution to visualize city bus routes on the basis of passenger frequency. Her continuous support and motivation always made us deliver our best. Having such guidance has always been an amazing experience which is a valuable gift for an engineer to progress in his/her life.

We are also grateful to Dr.Vandan Tewari, Head, Department of Computer Engineering and Dr.Rakesh Saxena, Director, S.G.S.I.T.S., Indore for providing us with numerous facilities and academic environment during the study.

We sincerely wish to express our gratefulness to all the members of staff of S.G.S.I.T.S. who have always extended their cooperation and have contributed in their way in developing the project.

The successful completion of a project is not an individual effort. It is an outcome of the cumulative number of people, each having their importance to the objective. We express love and respect towards our parents and all family members who are our strength in everything we do.

With a blend of gratitude, pleasure, and great satisfaction we convey our indebtedness to all those who have directly or indirectly contributed to the successful completion of our project work.

Amogh Mittal (0801CS201011) _____

Arpit Jain (0801CS201019) _____

Chinmay Dubey (0801CS201028) _____

Kanchan Hingorani (0801CS201046) _____

Vanshika Agrawal (0801CS201100) _____

ABSTRACT

The bus transportation service is an important part of a city's infrastructure and, as transit keeps increasing, the need to optimize this service grows. The urban transportation landscape is marked by complexity, and the conventional representation of bus routes often falls short in providing transportation authorities with a clear, dynamic, and user-friendly tool. The objective of this work is to enhance the accessibility, efficiency, and overall user experience of urban public transportation by providing an interactive dashboard which can be used for decision making during congestions due to increase in passenger frequency and other parameters.

This project report explores the design, development, and implementation of an innovative system for the visualization of city bus routes aimed at enhancing urban mobility. The work delves into the challenges associated with conventional bus route representations and establishes the need for a dedicated visualization solution.

The anticipated outcomes of this report include a comprehensive understanding of the significance of advanced visualization in urban transportation, the successful development and deployment of a robust city bus route visualization system, and insights that contribute to the broader discourse on improving the efficiency and accessibility of public transportation in urban environments. Ultimately, the report aims to present a compelling case for the integration of advanced visualization tools in the realm of urban mobility, paving the way for enhanced transportation experiences and sustainable urban development.

In conclusion, the visualization of the City Bus Routes project represents a valuable contribution to the field of urban transportation by offering an advanced, user-friendly solution for the visualization of bus routes. The successful implementation of this system has the potential to positively impact urban mobility, making public transportation more accessible and efficient for residents and visitors alike.

CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

In this chapter, there is a brief introduction about the project giving the need for the project, the problem being solved, objectives, and also the approach to be used.

1.1 Preamble

The modern urban landscape is characterized by its dynamic and intricate transportation networks, with city bus systems serving as lifelines for millions of commuters daily. In the face of burgeoning urban populations and evolving mobility demands, the need for innovative solutions to enhance the efficiency and accessibility of public transportation has become increasingly apparent.

One pivotal aspect that has long been underexplored is the adaptive visualization of city bus routes. Traditional representations, often confined to static maps or rudimentary navigation tools, fail to capture the complexity and nuances of urban transit networks.

Dynamic visualization of bus routes with respect to time could offer the bus transport department invaluable insights for efficient planning. By analyzing historical data on bus movements and passenger demand throughout different times of the day, the department can optimize bus schedules to align with peak commuting hours and adjust frequencies during periods of lower demand.

Integrating traffic data into the visualization of bus routes could provide the bus transport department with a dynamic strategic tool for proactive planning and response. Also, traffic-informed visualization would aid in the strategic placement of bus stops and the identification of locations where infrastructure improvements or traffic management interventions may be beneficial.

1.2 Need of the Project

The implementation of dynamic visualization of the city bus route project holds significant benefits for the transportation department, offering a valuable tool that can transform the efficiency, responsiveness, and overall effectiveness of urban transit systems. The key needs addressed by such a project for the transportation department include:

1. Optimized Resource Allocation:

Dynamic visualization of city bus routes provides the transportation department with a comprehensive understanding of the demand patterns and utilization of bus services over time. By analyzing the data in a visual format, transport planners can identify peak hours, busy routes, and areas of high demand. This insight enables more informed decisions regarding the allocation of resources, allowing the department to optimize bus schedules, adjust frequencies, and allocate buses strategically to meet the dynamic needs of commuters.

2. Data-Driven Decision-Making:

The adaptive visualization project empowers the transportation department with a data-driven approach to decision-making. Access to dynamic visual representations of bus route data allows planners to identify trends, analyze historical performance, and make informed adjustments to improve overall system efficiency. This data-driven decision-making process is essential for adapting to changing urban landscapes, accommodating population growth, and addressing emerging patterns in commuter behavior.

3. Enhanced Planning for Infrastructure Development:

The dynamic visualization of bus routes aids the transportation department in identifying areas where infrastructure improvements are needed. By visualizing traffic conditions, bus stops, and potential congestion points dynamically, planners can make informed decisions about the placement of new stops, the introduction of dedicated bus lanes, and other enhancements to improve the flow of bus services. This contributes to the long-term sustainability and effectiveness of the urban transit infrastructure.

1.3 Problem Statement

To develop a comprehensive dashboard tool tailored for city bus administrators, aimed at providing dynamic data visualization of city bus routes and passenger data on Google Maps.The dashboard will enable administrators to analyze passenger density and route performance trends over time, facilitating data-driven decisions for route optimization, scheduling, and resource allocation.

1.4 Objectives

1. **Enhanced Data Visualization**: Develop geographical maps to visually represent bus routes and related data to improve comprehension and analysis.
2. **Solving Real-World Transport Problems**: Address challenges faced by transport authorities by providing tools to plan and execute a more efficient transport system.
3. **Efficiency Improvement**: Enable transport authorities to optimize routes and schedules through spatial analysis, leading to reduced congestion, shorter travel times, and improved service reliability.
4. **Dynamic Visualization**: Create interactive maps that can adapt to real-time data, allowing for dynamic representation of bus routes, stops, and operational information.
5. **Readability**: Design maps that are easily readable and understandable for both transport authorities and passengers, facilitating effective decision-making and route planning.
6. **Spatial Analysis**: Utilize spatial analysis techniques to identify patterns, trends, and areas of improvement within the transport network, aiding in strategic planning and resource allocation.
7. **Optimization**: Offer tools for optimizing bus routes based on factors such as passenger demand, traffic patterns, and environmental considerations to enhance overall system efficiency.
8. **Admin Friendly Interface**: Develop a user-friendly interface that allows transport authorities to input data, perform analysis, and visualize results intuitively, without requiring specialized technical skills.

9. **Collaboration**: Foster collaboration between transport authorities, urban planners, and other stakeholders by providing a platform for sharing insights, feedback, and proposed improvements to the transport system.
10. **Continuous Improvement**: Establish a framework for ongoing updates and enhancements to the visualization platform, ensuring that it remains relevant and effective in addressing evolving transport challenges.

1.5 Proposed Approach

The proposed solution involves analyzing historical data to generate different types of geographical maps for transport authorities. The first type of map, a Density Map, illustrates traffic on each bus route by depicting the number of passengers during specific time intervals. The second type of map, a Heat Map, portrays the level of busyness at each stop by showcasing the number of tickets sold within designated time intervals. The third type of map depicts the average number of passengers in a bus during specific time intervals.

The interface will allow the Administrator to input a time interval, and in response, will get maps that provide a visual representation of the corresponding data during that specified time period. Detailed route specifics, including stops, timings, and prominent landmarks, will be presented to provide a holistic understanding of each bus route.

Chapter 2

BACKGROUND STUDY

In this chapter, the background information of the project is presented. The first section contains a brief description of the main technology domain in which our project falls, i.e., geographical data visualization. The tools and technologies, different frameworks, APIs required to build this project are specified in the next section.

2.1 Area of Concern

A critical area of concern because of lack of visual data representation of bus routes revolves around the limitations and challenges inherent in the current representation of urban transportation systems. Traditional methods of displaying bus routes often rely on static maps or rudimentary diagrams, which fails to capture the complexity and dynamics of modern urban transit networks. This deficiency poses significant obstacles for the transportation authorities in understanding, planning, and optimizing bus routes efficiently. Problems include, inefficient resource allocation, limited schedule optimization, ineffective route planning, reduced ability to analyze performance metrics. Addressing this concern is imperative for enhancing the overall effectiveness, accessibility, and sustainability of public transportation in rapidly evolving urban environments.

2.2 Tools and Technology

2.2.1 APIs -

We used two APIs -

- Heat Maps API
  Google Maps API Heat Maps offer a dynamic visualization tool to represent data intensity across geographical areas. By employing a spectrum of colours, from cooler hues for lower densities to warmer tones for higher concentrations, these maps vividly illustrate patterns and hotspots within your dataset. Whether

displaying foot traffic in urban centres, crime rates in neighbourhoods, or population densities across regions, Google's Heat Maps empower users to glean insights and make informed decisions at a glance. With customizable features and seamless integration into Google Maps, developers can effortlessly create compelling visualizations that enhance understanding and engagement with spatial data.

- Google Maps Route API

  The Google Maps Route API is a robust solution for developers seeking to incorporate seamless routing features into their applications. Leveraging Google's vast database and cutting-edge algorithms, this API empowers users to calculate the most efficient routes between multiple points, whether by car, public transit, walking, or cycling. By considering real-time traffic conditions, alternate routes, and user preferences, it provides accurate and up-to-date navigation guidance to optimize travel time and enhance the user experience. With comprehensive documentation and flexible customization options, developers can easily integrate dynamic routing capabilities into their applications, from transportation logistics to location-based services, unlocking a world of possibilities for efficient and intuitive navigation.

2.2.2 Python-

Python will be required for implementing various data visualization models because it supports various useful libraries. Python offers a rich ecosystem of libraries for data visualization, empowering developers and data scientists to create compelling and insightful visual representations of data. We will use pandas, numpy, matplotlib, numpy, seaborn libraries for various graphs like bar graphs, line graphs, line charts for comparison, analysis and visual presentation of the historical data provided. These Python libraries collectively offer a diverse set of tools for creating static and interactive visualizations, allowing users to choose the most appropriate library based on their specific requirements and preferences. The flexibility and extensibility of these libraries make Python a popular choice for data visualization tasks in various domains.

2.2.3 Frontend Technologies -

- HTML5: The latest version of HTML with enhanced features for structuring content.
- CSS3: The latest version of CSS with advanced styling capabilities.
- JavaScript (ES6+): Modern JavaScript for client-side scripting and interaction.

Chapter 3

LITERATURE REVIEW

This chapter describes the literature survey done to study the concepts of existing systems.

3.1 Inception

A study of various research papers on optimizing bus routes, data related to transport of Indore city, numerous studies highlighting the challenges faced by urban transportation systems globally, including issues such as traffic congestion, inefficient route planning, and the need for sustainable mobility solutions was done.

Examination of a case study from Delhi that tried to implement advanced bus route visualization but couldn't be applied and just remained theory offered valuable insights into successful strategies, challenges faced, and the outcomes achieved.

Several efforts have been made in the past to make the transport system visually better, which has motivated and guided us throughout the process.

3.2 A Study of Existing Solutions

A study of currently available transport apps (like Chalo App), the official website of AiCTSL (the official transport authority of the city) and google maps was done. Our challenge was to provide what none of these platforms provided - data visualization on maps based on historical data which could give insights to the transport department on planning and efficient execution of the city bus transport system.

3.2.1 AICTSL

AICTSL, as the official transport authority of the city, primarily focuses on providing basic bus data through platforms like its official website and mobile applications. While these resources offer valuable information such as bus schedules, routes, and stops, they typically lack advanced data visualization capabilities beyond static maps. AICTSL's current approach emphasizes disseminating essential transit information to commuters without delving deeply into comprehensive data analysis or historical insights

3.3 Limitations of current solutions

- Static Data Presentation: AICTSL's system presents bus data statically, limiting its ability to capture real-time changes and trends in passenger behavior.
- Lack of Historical Analysis: Without historical data analysis, AICTSL's system misses opportunities to identify long-term patterns and make informed decisions for future planning.
- Limited Spatial Understanding: AICTSL's system lacks spatial analysis capabilities, hindering its ability to pinpoint areas of congestion or optimize bus routes based on passenger densities.
- Inflexible Planning: The absence of dynamic visualization hampers AICTSL's ability to adapt quickly to changing conditions, potentially leading to inefficient resource allocation and service disruptions.
- Ineffective Resource Allocation: AICTSL's system may struggle to allocate resources efficiently without real-time insights into passenger demand, risking over-servicing low-demand routes and neglecting high-demand areas.

3.4 Differentiating Factors of our solution

- **Adaptively Visualize Passenger Densities**: Dynamically adjust visualization parameters to reflect changing passenger activity in real-time.
- **Responsive Peak Travel Time Identification**: Adapt to shifting commuter patterns to accurately pinpoint peak travel times for efficient resource allocation.

- **Agile Congestion Management**: Employ adaptive techniques to swiftly identify and address congestion hotspots as they emerge.
- **Flexibly Address Underutilized Areas**: Utilize adaptive strategies to identify and repurpose underutilized stops or routes for improved service coverage.
- **Data-Driven Adaptive Decision-Making**: Utilize real -time data visualization to dynamically adapt strategies and optimize city bus operations.
- **Strategically Adaptive Planning**: Harness adaptive analytics to continuously refine and enhance strategic capabilities for optimal network performance.

Chapter 4

ANALYSIS

In this chapter, a detailed analysis of the project has been discussed. It includes requirement analysis. i.e. functional requirements and non-functional requirements of the proposed system, various graphs based analysis of the city bus routes data.

4.1 Detailed Problem Statement

Despite the critical role of bus transportation in urban mobility, transportation authorities currently lack a comprehensive tool to effectively monitor and analyze passenger movement across city bus routes. This absence of a centralized data visualization platform hampers decision-making processes regarding route optimization and resource allocation. As a result, transportation administrators are unable to identify and address potential inefficiencies within the bus network, leading to suboptimal service quality and customer satisfaction.

The primary goal of this project is to develop a City Bus Data Visualization Tool that leverages Google Maps to dynamically illustrate the passenger frequency on different bus routes throughout the day. The tool should allow admin to interactively explore and analyze the data, gaining valuable insights into the patterns and trends of passenger movement on each route. Detailed route specifics, including stops, timings, and prominent landmarks, will be presented to provide a holistic understanding of each bus route. The tool aims to empower the bus transport authority with actionable insights by highlighting critical points where passenger frequency experiences notable fluctuations.

Addressing this problem will lead to optimized bus routes, improved admin experience, and a more data-driven approach to managing Indore's public transportation network.

4.2 Requirement Analysis

In this section the requirements of the system are specified in both functional and non-functional requirements and resource requirements.

4.2.1 Functional Requirements

1. **Data Collection:**
    - The system shall collect real-time and historical data on passenger frequency for each city bus route in Indore.
    - It should integrate with existing data sources or deploy sensors for accurate and up-to-date information.
2. **Google Maps Integration:**
    - The tool shall leverage the Google Maps API to accurately visualize city bus routes and stops.
    - It should provide an intuitive interface allowing users to select and view specific routes seamlessly**.**
3. **Time-Based Visualization:**
    - The system shall implement a time-slider or similar functionality to dynamically visualize passenger frequency at different times of the day.
    - Users should be able to switch between daily, weekly, or monthly views for comprehensive trend analysis.
4. **Route Specifics:**
    - The tool shall display detailed information for each bus route, including stops, timings, and key landmarks.
    - It should highlight critical points where passenger frequency experiences significant fluctuations.

5. **User Interaction:**
    - The system shall allow users to click on specific routes or bus stops to access more detailed information.
    - It should provide a user-friendly interface catering to the needs of both transportation authorities and the general public**.**
6. **Customization Options:**
    - The system shall allow users to customize views based on specific parameters such as date, time, and routes.
    - Users should be able to filter and focus on particular aspects of the data for in-depth analysis.

4.2.2 Non Functional Requirements

Requirements, which are not related to the functional aspect of software, fall into this category. They are implicit or expected characteristics of a software. The major non-functional requirements of the system are:

1.Performance: The system should exhibit fast response times to ensure real-time or near-real-time processing of inputs and data.

2.Scalability: The system should be designed to handle varying workloads, including a large number of concurrent users and high speed data processing.

3.Reliability: The tool should have high uptime to ensure continuous availability for users and transportation authorities.

4.**Security and Privacy**: The system should adhere to security best practices to protect user data and prevent unauthorized access.

5. Compatibility: The system should be compatible with major web browsers (Chrome, Firefox) and accessible across different devices.

6. Usability: The system should have a user-friendly interface that is intuitive and easy to navigate.

7. Maintainability : The system should be designed with modularity and documented code to facilitate easy maintenance, updates and future enhancements.

8. Portability: The system should be designed to be easily deploy able across different platforms and environments.

4.3 Feasibility Study

The City Bus Data Visualization Tool is proposed to address the challenges in managing and optimizing Indore's public transportation system. This feasibility study assesses the technical, economic, operational, and scheduling aspects to determine the viability and potential success of the project.

4.3.1 Technical Feasibility:

Assessment: The technical infrastructure required for real-time data collection, Google Maps integration, and the implementation of predictive analysis algorithms is readily available. The proposed technologies align with industry standards and can be effectively implemented.

Conclusion: The project is technically feasible, leveraging proven technologies for seamless integration and functionality.

4.3.2 Economic Feasibility:

Assessment: The initial investment includes development costs, integration expenses, and potential sensor deployment. The tool's benefits include improved route optimization, enhanced user experience, and data-driven decision-making for transportation authorities.

Conclusion: The project is economically feasible, with a positive return on investment expected through improved public transportation efficiency.

4.3.3 Operational Feasibility:

Assessment: The tool is designed to be user-friendly for both transportation authorities and the general public. Training requirements are minimal, and the system allows for easy maintenance and updates.

Conclusion: The project is operationally feasible, with a focus on user acceptance and efficient system management.

4.4 Conclusion:

The feasibility study demonstrates that the City Bus Data Visualization Tool is technically, economically, operationally, and legally viable. The proposed system addresses the current challenges in public transportation management and offers a tangible and beneficial solution for both transportation authorities and the public. The project is recommended to proceed to the development phase based on the positive outcomes of this feasibility study.

4.5 Graphical Analysis of City Bus Data:

4.5.1 Objective: To analyse and visualize the number of passengers boarding at various stops for different routes.

There are multiple routes that connect with a single stop.We analyzed the bus ridership data for various bus routes and tried to determine the passenger density at different stops.

Example: We have performed an analysis of the data for all stops on the date 14/11/2022. There were certain stops with a passenger count exceeding 5000, while others had counts lower than 2**.**

Fig 4.1 Number of passengers on different stops

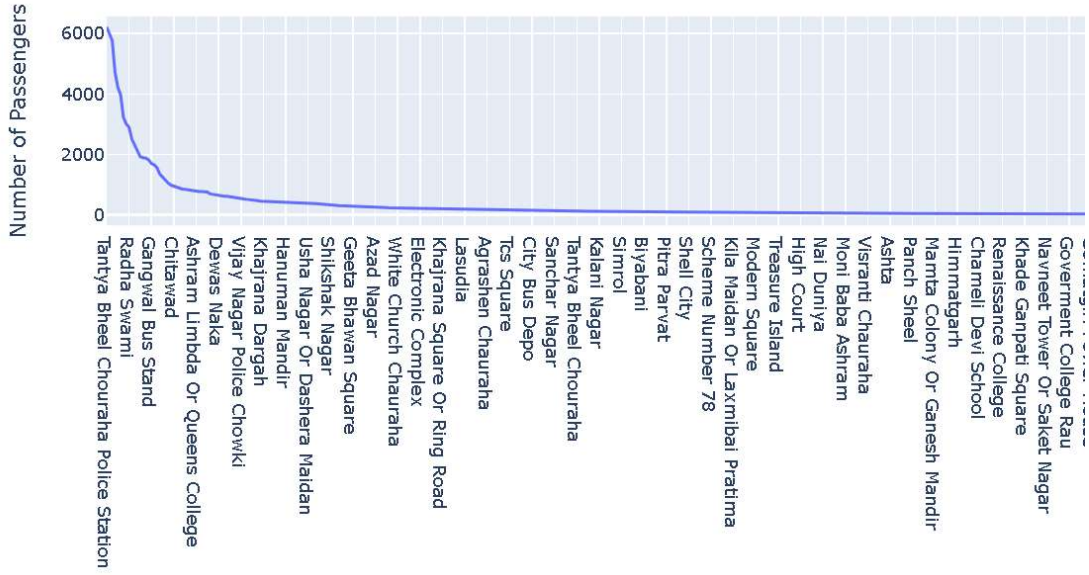


Fig 4.2 Maximum Stop density

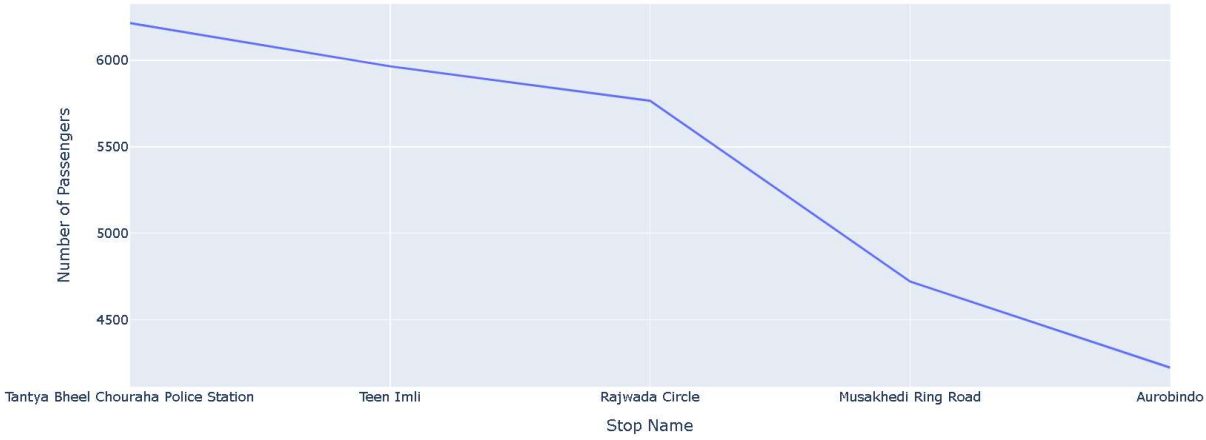


Fig 4.3 Minimum Stop density

Conclusion: Maximum number of passengers are on Tantya Bheel Choraha = 6881

Minimum number of passengers are on Gotiya Aam Road =1, Hoshangabad = 1

4.5.2 Objective: To analyse and visualize total no. of passengers onboarding the buses at each station in time interval of one hour . Visualizing passenger numbers by hour helps pinpoint peak travel times, enabling transit agencies to allocate more resources and optimize services during periods of high demand, reducing overcrowding and improving service quality.

Identifying busy stations and time intervals allows for targeted route adjustments, such as adding extra buses or optimizing routes to accommodate higher passenger loads.

Fig 4.4 Passengers Boarding between 10 AM to 11 AM

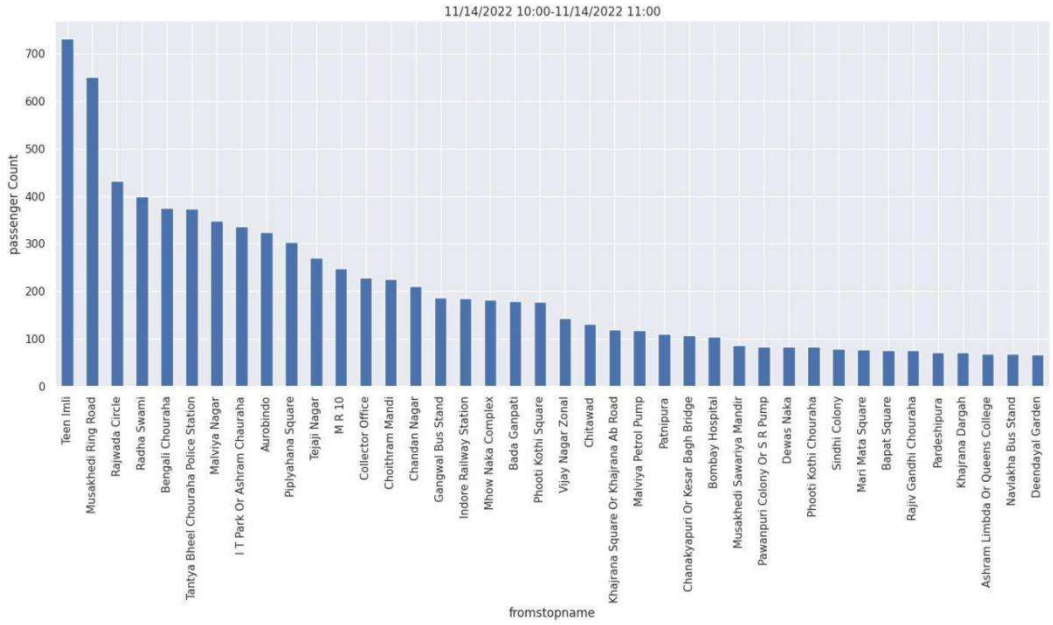
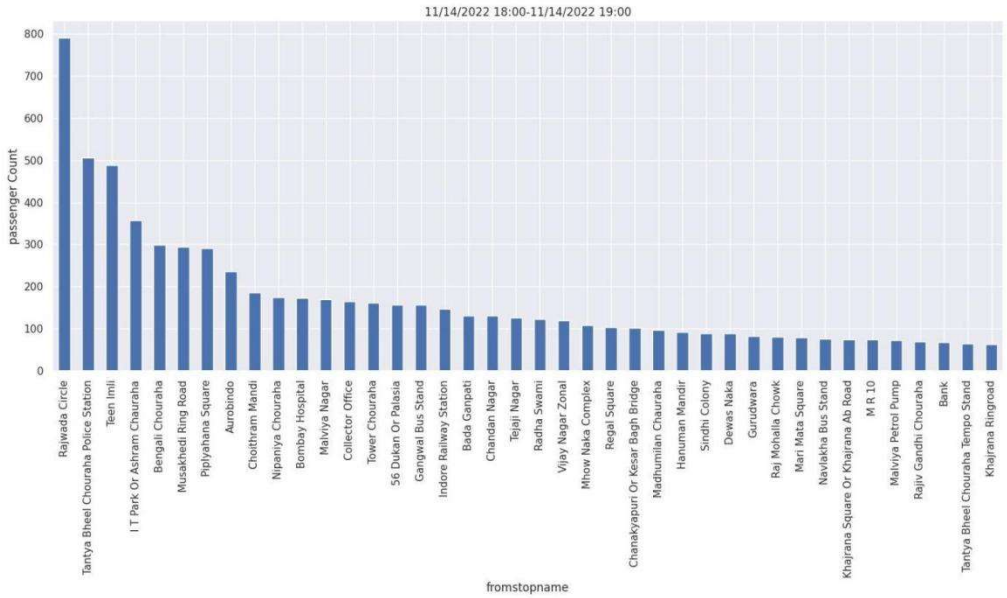


Fig 4.5 Passengers Boarding between 6PM to 7PM



Conclusion:

- Some of the bus stops had higher passenger frequency during morning hours and lesser during evening hours.
- Some bus stops such as Rajwada, Teen Imli have high frequency of passengers throughout the daytime.

Based on the insights from this data analysis, a bus route can be re planned by eliminating the bus stops which have very less number of passengers throughout the day. Also, the frequency of buses can be increased for the routes having many stops with high number of passenger counts, thus preventing overcrowding in buses.

4.5.3 Objective: To analyse and visualize the number of passengers for different routes from each bus stop. Analyzing the number of passengers for different routes from each bus stop for the entire day helps us to get a comprehensive understanding of travel patterns and demand trends. By understanding the demand for different routes from each bus stop, transportation authorities can optimize the routes to ensure that buses are assigned based on passenger demand. This helps reduce overcrowding and ensures efficient utilization of resources.

From the initial data set, 3 features are required for this analysis:

1. fromStopName 2.Passengers 3. Route_name. Using these features, data conversion process is performed to get the stop wise data of count of total passengers for each route.

Fig 4.6 Count of passengers on boarding at a bus stop

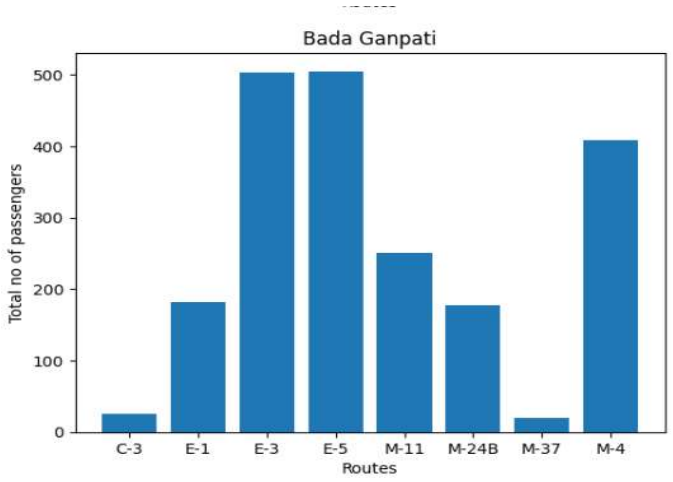


Fig 4.7 Total passengers on boarding for each route from a bus stop

	A	B	C
1	**fromstopname**	**route_name**	**Count of total passengers**
2	56 Dukan	C-1	28
3	56 Dukan Or Palasia	E-3	289
4	56 Dukan Or Palasia	R-9	406
5	7 Meel	M-22	23
6	8 Meel	M-22	65
7	9 Mile	M-19	363
8	Aastha Talkies	R-17	47
9	Abhinandan Pump	C2-(16)	32
10	Abhvaj Udyan Or Choi	R-5	278
11	Advanced Academy	C-1	12
12	Agarwal Public School	M-29	27
13	Agrashen Chauraha	E-4	189
14	Agrawal Toll Kanta	M-17	41
15	Agrawal Toll Kanta	M-26	17
16	Agrawal Toll Kanta	M-26A	1
17	Agrawal Toll Kanta	N-5	8
18	Agrawal Toll Kanta	R-4	13
19	Ahilya Ashram	M-29	12
20	Airport	E-1	108
21	Airport	M-24B	38
22	Airport Gate	E-1	20

Conclusion:

- Many of the bus stops have some routes for which the count of passengers is very less as compared to the other routes.
- Only a few bus stops had an almost equal amount of passenger distribution among all their routes**.**

Based on the insights from this data analysis, a bus route can be re planned by eliminating the bus stops which have very less number of passenger counts. Also, the frequency of buses can be increased for the routes having many stops with high number of passenger counts.

4.5.4 Objective: To analyze and visualize the number of passengers on a particular route with respect to time. We studied the ticket sales volume for a specific route across various time intervals, aiming to depict the variation in passenger numbers corresponding to different times of the day. Then we compared the data of multiple routes to analyze the demand difference at various times of a day**.**

Example: We analyzed the passenger count corresponding to time data for 3 bus routes on hourly time interval :**M-6, R-9, M-36.** We studied these Indore city bus routes and compared them to put in context the variations of time and passengers.

Fig 4.8 Bus Route M-6 data analysis

Route : Tejaji nagar to Rajwada

Distance : 10.5 Kms Number of Buses : 8

Peak Passenger Count at : 5 PM

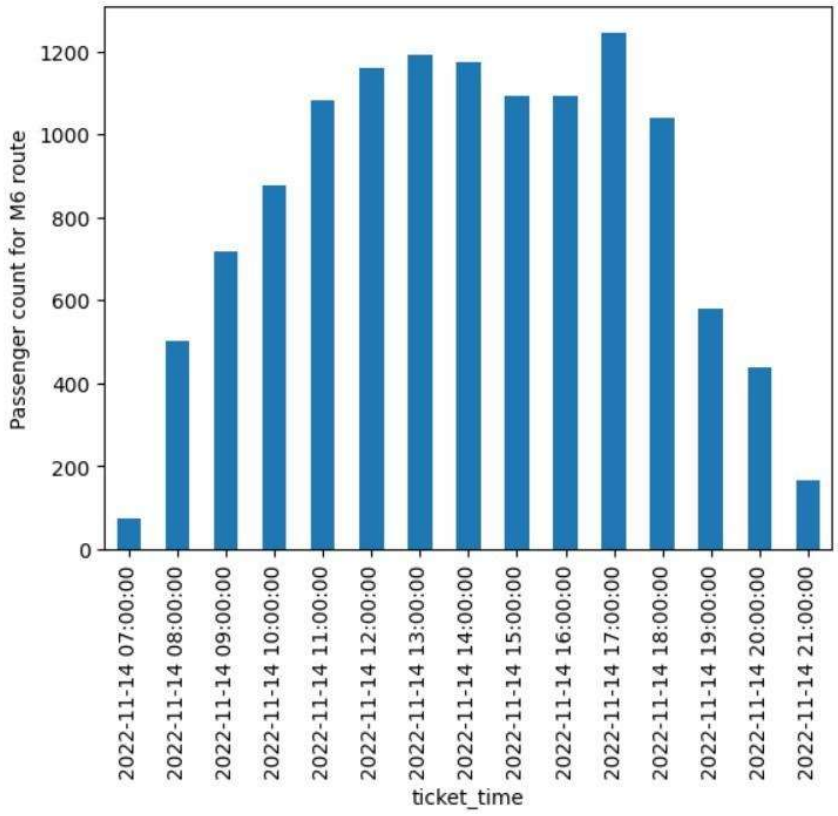


Fig 4.9 Bus Route R-9 data analysis

Route: Suryadev Nagar to By pass

Distance: 15.5 Kms Number of Buses: 13

Peak Passenger Count at: 4 PM

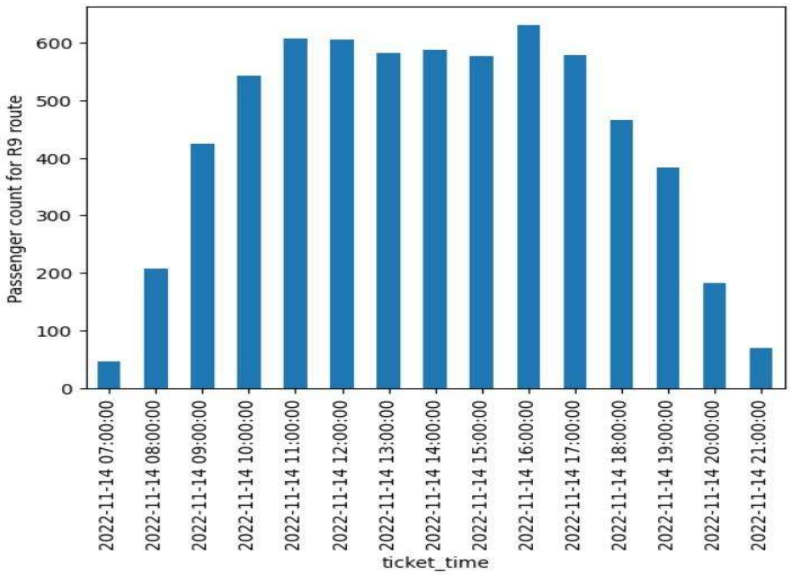


Fig 4.10 Bus Route M-36 data analysis

Route : MR-10 To Rajwada

Distance : 7.2 Kms Number of Buses : 5

Peak Passenger Count at : 2 PM

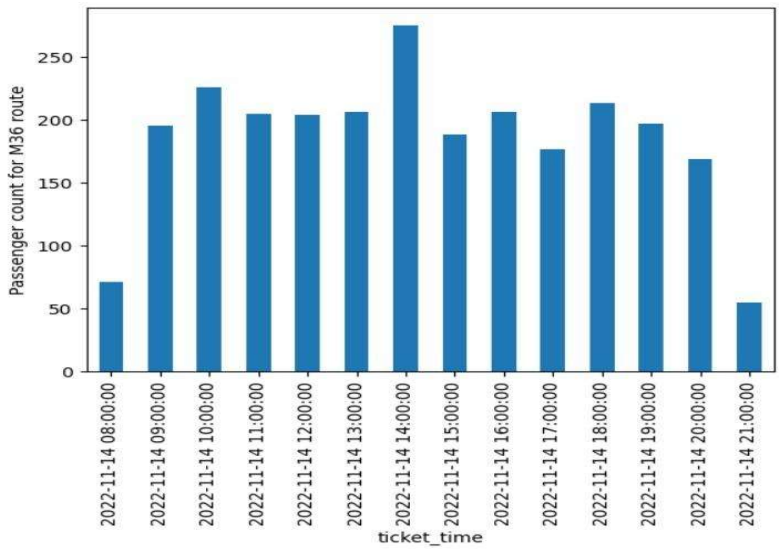
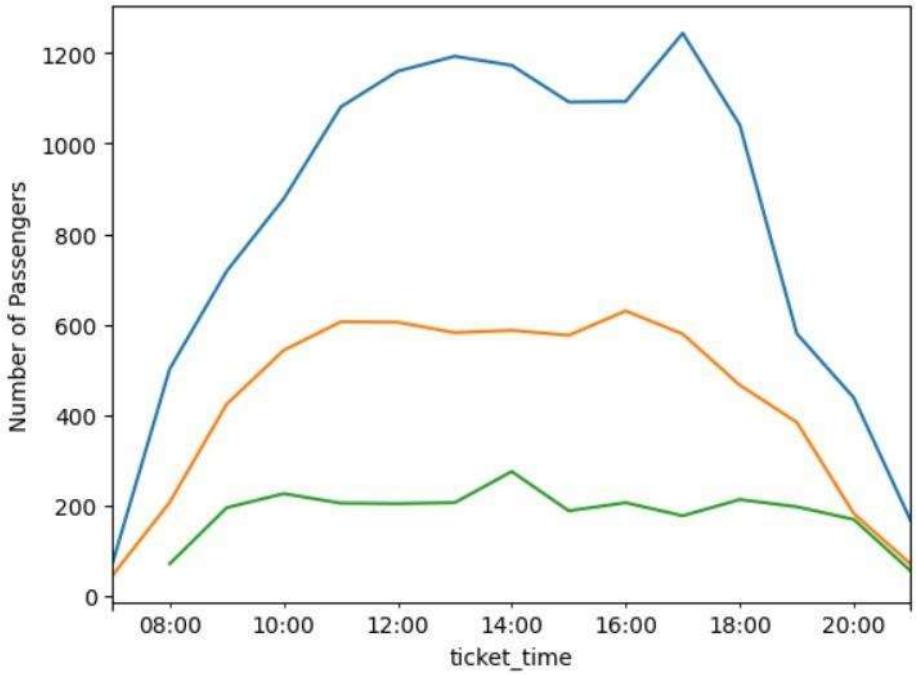


Fig 4.11 Comparing M-6, R-9 and M-36 Bus Routes

M-6 routes has way more passengers than R-9 and M-36. All the three routes have a peak passenger demand at different times in a day. M-6 and R-9 bus routes have longer ticket sale times i.e. tickets sell even before 8:00 AM unlike M-36 route.



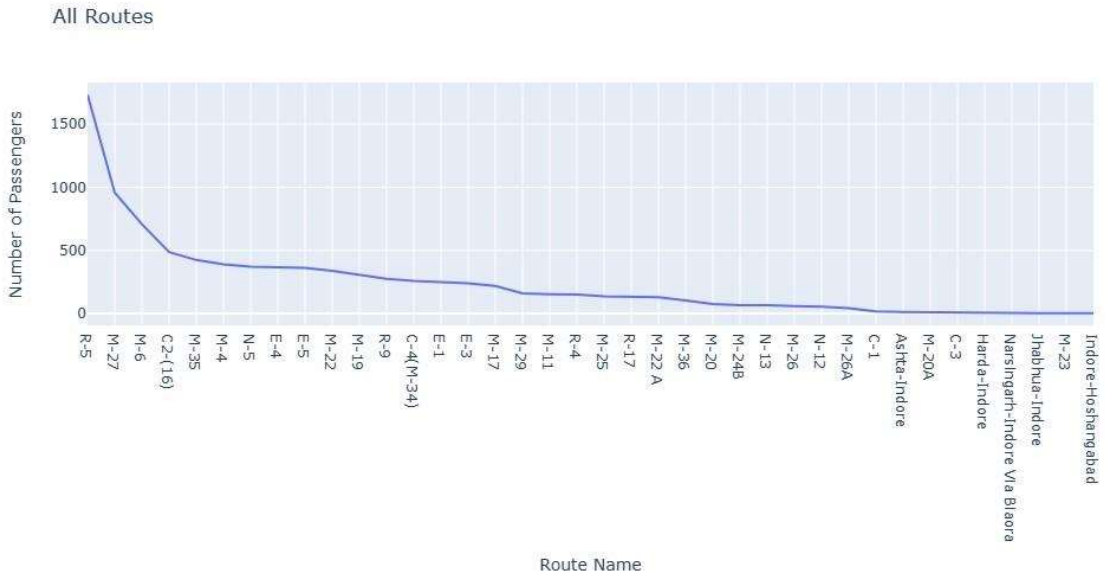
Conclusion :

- We got insights into how passenger numbers fluctuate over different time periods, offering a comprehensive view of the route's temporal passenger trends.
- We deduce that the buses can be efficiently used if the bus frequency and distribution is done corresponding to the passenger count variations with respect to time.
- Bus frequency should not be fixed for a route and be dynamic instead, depending on various factors, one of them being time.

4.5.5 Objective: To analyze and visualize the number of passengers travelling through different routes in a given time interval. Identify the routes with maximum passengers, identify the routes with minimum passengers.

Example: We have conducted an analysis of the data for all the routes in a time interval of 8AM-9AM. There were some routes which were over-utilized and some routes which were under-utilized.

Fig 4.12 Number of passengers on different routes



- Maximum number of passengers are on route R-5 = 1733.
- Minimum number of passengers are on route Indore- Hoshangabad = 3

Fig 4.13 Maximum number of passengers per bus on different routes

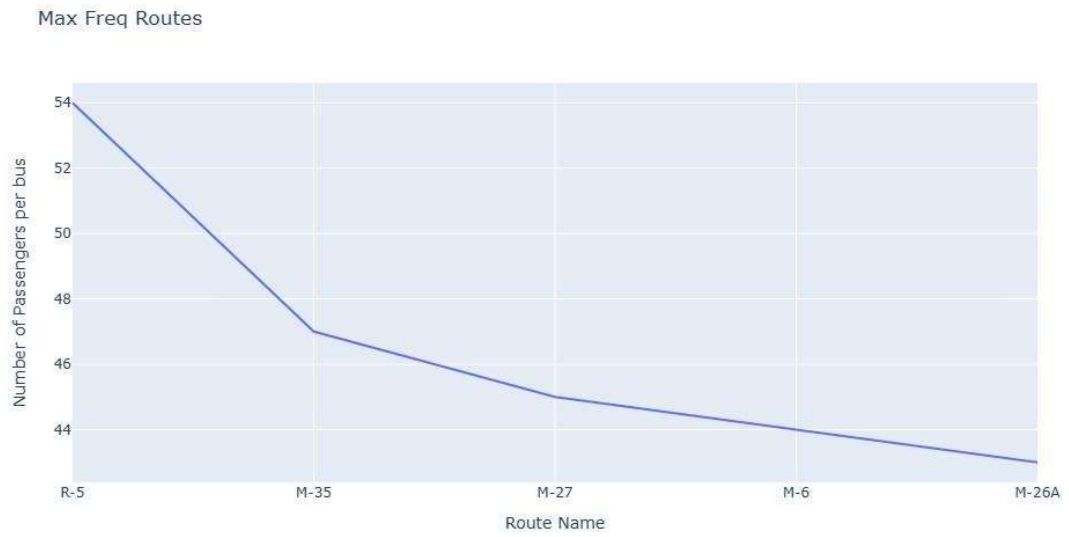


Fig 4.14 Minimum number of passengers per bus on different routes

Conclusion: We got insights into how passenger numbers fluctuate over different routes. We can conclude that some routes are under-utilized while others are over- utilized which can be handled by changing the frequency of buses.

4.5.6 Objective: To analyse and visualize the number of passengers in a bus at each stop for different trips of buses. There are multiple routes with several buses operating on each route. The buses undertake multiple trips throughout the day.We analyzed the bus ridership data for various bus routes and tried to determine the current passenger count on the bus for multiple trips throughout the day.

Example: We have conducted an analysis of the data related to bus number MP09FA6080 across its multiple trips on Route M-6. To determine the passenger count at each stop to find when the bus has the highest number of passengers.

Fig 4.15 Bus number MP09FA6080 Trip 1

Route: Tejaji nagar to Rajwada

Distance: 10.5 Kms Trip number: 1

Number of Buses: 8

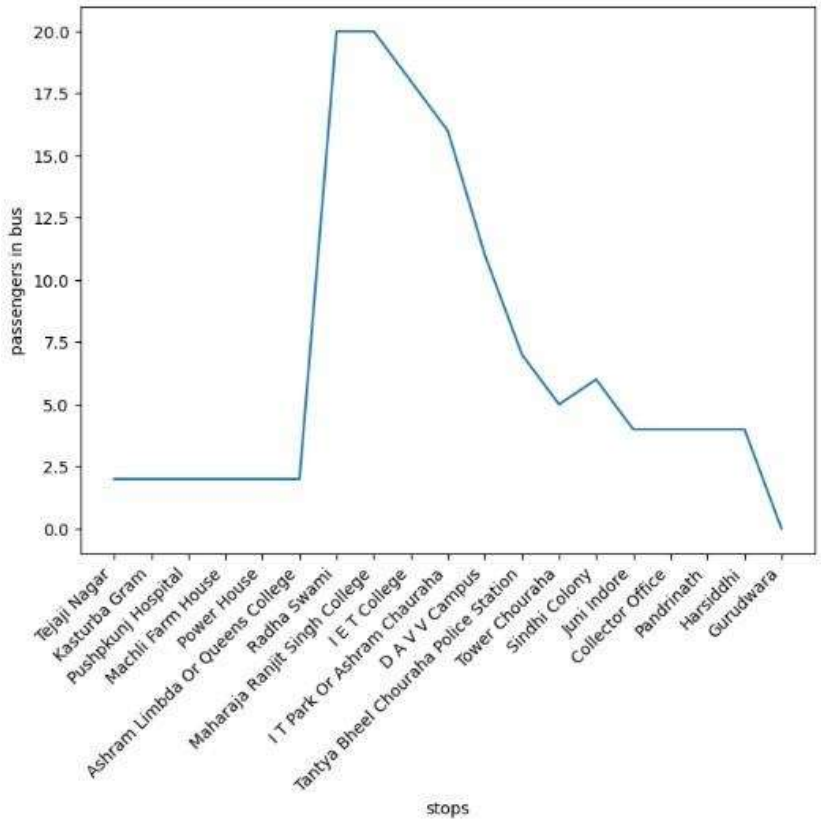
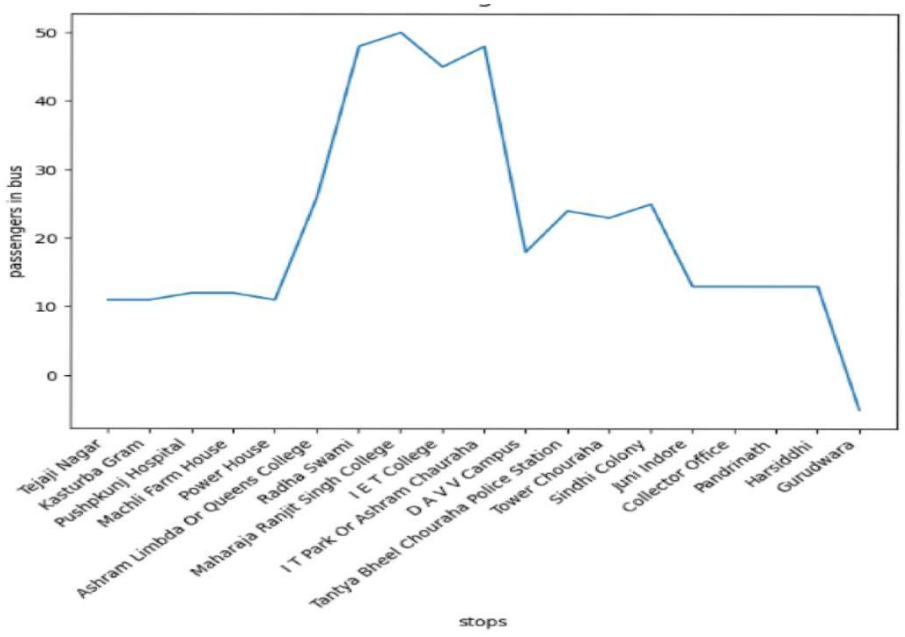


Fig 4.16 Bus number MP09FA6080 Route M-6

Route: Tejaji nagar to Rajwada

Distance: 10.5 Kms Trip number: 3

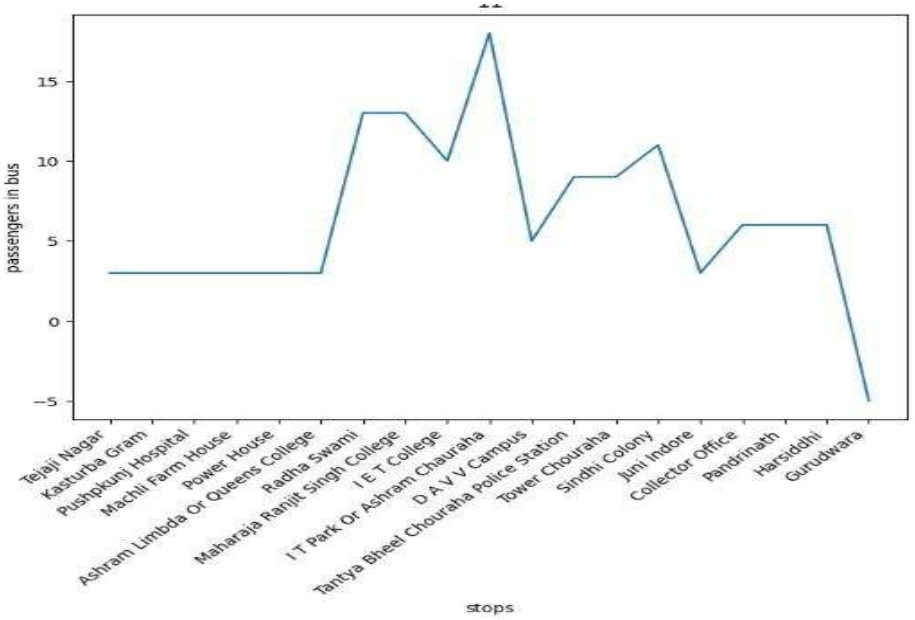
Bus Number: MP09FA6080



Route : Tejaji nagar to Rajwada

Distance : 10.5 Kms Trip number : 5

Bus Number : MP09FA6080



Conclusion:

- Peak Passenger Count: Trip 3, which typically begins around 11:15 AM, consistently registers the highest passenger count among all trips throughout the day.
- Passenger Patterns: Certain stops along Route M-6 exhibit higher passenger activity compared to others. The region stretches from Radha Swami to Tantya Bheel Choraha.
- Variation: The passenger count on bus no. MP09FA6080 varies across different trips and times of the day.

Chapter 5

DESIGN

5.1 System Architecture

A system architecture is the conceptual model that defines the structure, behavior, and more views of a system. An architecture description is a formal description and representation of a system, organized in a way that supports reasoning about the structures and behavior of the system.

The figure below shows the high-level design of the City Bus Data Visualization Tool.

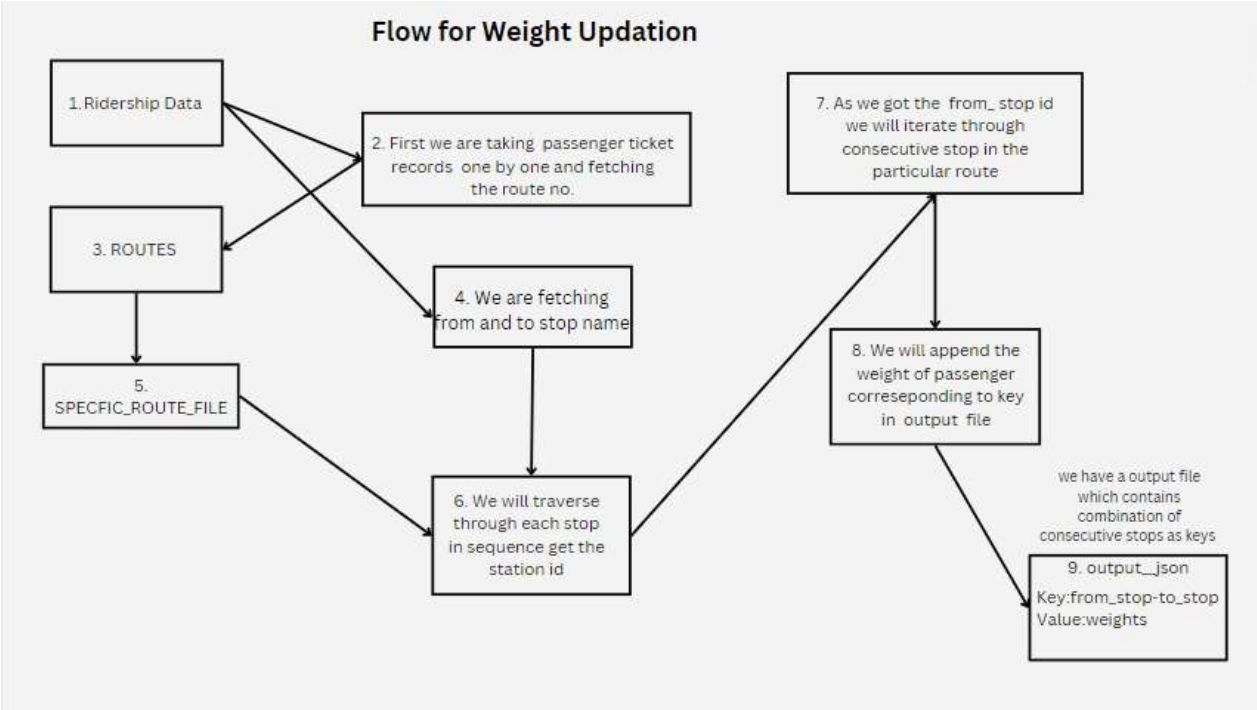


Figure 5.1: Flow Chart for Weight Updation

We can outline the approaches in the above architecture as follows:

1. Extract unique stops from the data.
2. Mapping routes to corresponding coordinates
3. Calculating density on each route

In the data extraction phase, unique stop names and their coordinates are obtained from Google Maps and stored in all_stations.json. Similarly, unique routes are identified and assigned unique IDs, stored in all_routes.json. Individual route files are then created, containing the order of stations for each route. The iteration through data involves processing passenger information one by one, matching their start and destination stops to determine the route ID, and updating a result file (result_density.json) with the corresponding passenger numbers. Finally, the data is represented visually by creating a map where stops are plotted using their coordinates, and passenger density between stops is color-coded. The final output is generated in a suitable format (e.g., JSON or CSV) using a chosen programming language and visualization tool (e.g., Python with Matplotlib). The resulting visualizations offer insights into passenger density along different routes at specific

5.2 Algorithm for Dynamic Data Visualization:

Algorithm of our particular City Bus Data Visualization tool includes these 4 major steps:

1. **Data Extraction**

    1.1 Extracting Stop Coordinates

    - A file (all_stations.json) containing unique stop names and their corresponding coordinates obtained from Google Maps.

    1.2 Extracting Route Information

- A file (all_routes.json) contains unique route names with assigned unique IDs.

1.3 Creating Route Files

- Individual route files are created, each containing the order of stations for that route along with the station Ids.

2. **Iterating Through Data**

2.1 Processing Passenger Data

- Iterate through each passenger's data.
- Match the passenger's start and destination stops with the routes to determine the route ID.
- Update a result file (result_density.json) with the passenger number corresponding to the particular route.
- Determine the number of buses running over a particular route.
- Calculate the passenger-to-bus density using the number of passengers in a bus and the number of buses on a route.

3. **Representing the Data**

3.1 Dynamic Visualization for Passenger Density

- From result_density.json, create a visual representation of passenger density for each route.
- Use the coordinates of stops from all_stations.json to plot the stops on a map.
- Create a visual representation of passenger density for each route by mapping the stations with varying colors or symbols based on the density of passengers.

3.2 Dynamic Visualization for Bus Density

- From result_density.json, containing passengers and bus count on a station, create a visual representation of bus density for each route.

- Utilize the coordinates of stops from all_stations.json to accurately plot the routes on a map.
- Create a visual representation of bus density for each route by mapping the stations with varying colors or symbols based on the density of buses.

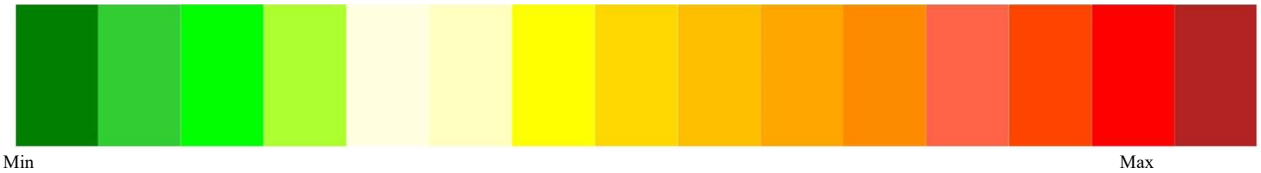
3.3 Dynamic Visualization for Passenger to Bus Density

- From result_density.json, containing passengers and bus count on a station, create a visual representation of bus density for each route.
- Calculate the passenger-to-bus density for each station along each route by dividing the number of passengers by the number of buses.
- Use the coordinates of stops from all_stations.json to plot the stops on a map.
- Create a visual representation of passenger to bus density for each route by mapping the stations with varying colors or symbols based on the density of passengers and buses.
- Additionally, the visualization will include an alert system to notify users of any instances where the maximum passenger-to-bus density threshold is exceeded.

These instances will be indicated by routes represented in black on the map.

3.4 Color-Coding

- Apply different colors to represent passenger density levels between stops.
- The color scale is designed such that lighter shades, leaning towards green, signify lower density, while darker shades, tending towards red, indicate higher density.



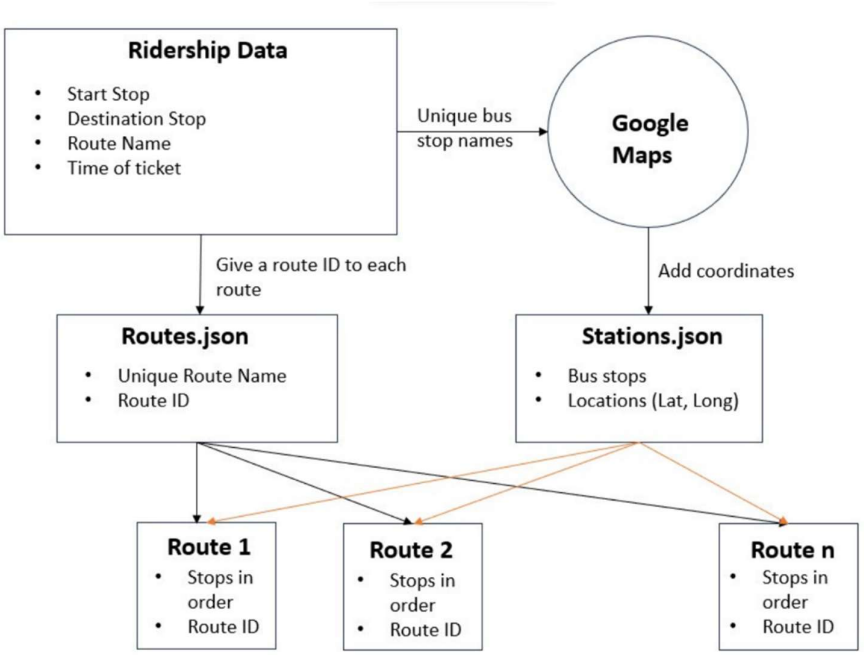
Chapter 6

IMPLEMENTATION

The implementation phase revolves around the development of the system modules and providing their description. The implementation also deals with providing the full logical view of the modules. The chapter involves the modules and package descriptions that are used for implementation.

6.1 Implementation Diagram

Fig. 6.1 Data Flow Diagram



6.2 Code Description

1.Setting up Google Maps API

Step 1: Create a Google Cloud Platform (GCP) Project

Go to the Google Cloud Console:

- https://console.cloud.google.com/
- Create a new project:
    - Click on the project dropdown at the top of the console.
    - Click on "New Project" and fill in the required information.

Step 2: Enable the Google Maps APIs

- Navigate to the API Library:
    - In the Cloud Console, go to the "APIs & Services" > "Library."
- Enable the necessary APIs:
    - Enable at least the following APIs:
        - Google Maps JavaScript API
        - Geocoding API
        - Places API

Step 3: Create API Credentials

- Create API Key:
    - In the Cloud Console, go to "APIs & Services" > "Credentials."
    - Click on "Create Credentials" and select "API Key."
- Configure API Key:
    - You can restrict the API key for security. Set restrictions based on HTTP referrers, IP addresses, or Android/iOS app restrictions.

Step 4: Integrate API Key in Your Application

- Include the API script in your HTML:
    - Add the following script tag to the <head> of your HTML file:
    - html
    - Copy code

<script async defer src="https://maps.googleapis.com/maps/api/js?key=YOUR_API_KEY&callback=initMap" type="text/javascript"></script>

- Replace YOUR_API_KEY with the API key you generated.

- Implement the Map:
  - In your JavaScript, create a function (e.g., initMap) to initialize the map.
  - Use the Google Maps JavaScript API to display a map on your webpage.

Step 5: Billing Setup

- Enable Billing:
  - Google Maps API usage may incur charges. Enable billing on your GCP project.
- Set Up a Billing Account:
  - Go to the "Billing" section in the Cloud Console and follow the instructions to set up billing.

6.3 Code

1. make_json

```
const fs = require('fs');
const data = require('./GetRoutes.json');
var jsonData={};
for (let i = 0; i < data.length; i++) {
    var path = './Routes/'+data[i].RouteId+'.json';
    if (fs.existsSync(path)) {
        var temp=require(path);
        for(let j=0;j<temp.length-1;j++){
            var st=temp[j].stationname+' to ' + temp[j+1].stationname;
            jsonData[st]=0;
        }
    }
}
const jsonDataf = JSON.stringify(jsonData, null, 2); // The second argument adds indentation for readability

// Specify the file path where you want to save the JSON data
const filePath = 'output.json';

// Write the JSON string to the file
fs.writeFile(filePath, jsonDataf, (err) => {
  if (err) {
    console.error('Error writing to JSON file:', err);
  } else {
    console.log('Data has been written to the JSON file:', filePath);
  }
});
```

Explanation:

It takes routes from routes file and create a output json file which contains A-B format stations list which will later contain the number of passengers between A-B

2. Map.json

```
        else if(a==='M-11'){
            temp=temp3;
        }
        else if(a==='M-17'){
            temp=temp4;
        }
        else continue;

        var start=bus.rows[j].fromstopname,end=bus.rows[j].tostopname;
        // console.log(start);
        var startid,endid;
        for(let k=0;k<busdata.features.length;k++){
            if(start===busdata.features[k].properties.name){
                // console.log(1);
                startid=busdata.features[k].properties.stationid;
                break;
            }
        }
        for(let k=0;k<busdata.features.length;k++){
            if(end===busdata.features[k].properties.name) {
                endid=busdata.features[k].properties.stationid;
                break;
            }
        }
        let l=0;
        for(;l<temp.length;l++){
            if(startid===temp[l].stationid) break;
        }
        while(l<(temp.length-1)&&endid!=temp[l+1].stationid){
            // console.log(1);
            var st=temp[l].stationname+' to '+temp[l+1].stationname;
            output[st]+=parseInt(bus.rows[j].passengers);
            l++;
        }
        l++;
        if(l<(temp.length-1)){
            var st=temp[l].stationname+' to '+temp[l+1].stationname;
            output[st]+=parseInt(bus.rows[j].passengers);
        }
    }
}
```

```
for(let i in output){
    let value = output[i][0];
    if(value<mi) mi=value;
    if(value>mx) mx=value;
    // console.log(value);
    if(value>0){
        const myArray = i.split(" to ");
        var obj = {},obj1={};
        for(let j = 0;j<stations.length;j++){
            if(myArray[0]===stations[j].StationName){
                var o = {};
                o['lat']=parseFloat(stations[j].Lat1);
                o['lng']=parseFloat(stations[j].Long1);
                obj['origin']=o;
                obj1['origin']=o;
            }

            if(myArray[1]===stations[j].StationName){
                var o1 = {};
                o1['lat']=parseFloat(stations[j].Lat1);
                o1['lng']=parseFloat(stations[j].Long1);
                obj['destination']=o1;
                obj1['destination']=o1;
            }
        }
        obj1['travelMode']= 'DRIVING';
        co.push(obj1);
        obj['wt']=value;
        coor.push(obj);

    }

}

var map = new google.maps.Map(document.getElementById('map'), {
    center: { lat: 40.7128, lng: -74.0060 },
    zoom: 6
});

var directionsService = new google.maps.DirectionsService();
// Request and render multiple routes
p=0;
co.forEach(function(request, index) {
    directionsService.route(request, function(response, status) {
        if (status === 'OK') {
            renderRoute(map, response, getRandomColor() ); // Get a random color for each route
        } else {
            alert('Directions request failed: ' + status);
        }
    });
});
```

```
function initMap() {
    const jsonString = JSON.stringify(output, null, 2);

    // Create a Blob containing the JSON data
    const blob = new Blob([jsonString], {type: 'application/json'});

    // Create a temporary URL for the Blob
    const url = URL.createObjectURL(blob);

    // Create a link element
    const link = document.createElement('a');
    link.href = url;
    link.download = 'passenger_count_data.json'; // Set the file name
    link.click();
}


function renderRoute(map, response, color) {
    var directionsRenderer = new google.maps.DirectionsRenderer({
        map: map,
        polylineOptions: {
            strokeColor: color ,
            strokeWeight: 6,
        },
        suppressMarkers: true,
    });

    directionsRenderer.setDirections(response);
}

var i=0;
function getRandomColor() {

    const gradientColors = [

        'rgb(0, 128, 0)',       // Green
        'rgb(50, 205, 50)',     // Lime Green
        'rgb(0, 255, 0)',       // Lime
        'rgb(173, 255, 47)',    // Yellow-Lime
        'rgb(255, 255, 224)',   // Light Yellow
        'rgb(255, 255, 192)',   // Pale Yellow
        'rgb(255, 255, 0)',     // Yellow
        'rgb(255, 215, 0)',     // Gold
        'rgb(255, 191, 0)',     // Amber
        'rgb(255, 165, 0)',     // Orange-Yellow
        'rgb(255, 140, 0)',     // Orange
        'rgb(255, 69, 0)',      // Red-Orange
        'rgb(255, 99, 71)',     // Deep Orange
        'rgb(255, 0, 0)',       // Red
        'rgb(178, 34, 34)'      // Fire Red
    ];
    const step=Math.max((mx-mi)/15,1);
    const value1 = Math.abs(Math.ceil((coor[p].wt/step)-1));

    const color = gradientColors[Math.max(value1,0)];

    p+=1;

    return color;
}

window.initMap = initMap;


function docReady() {
        initMap();

    // Call the function to load the API
}
window.addEventListener("load", docReady);
```

File Description for passengers' count:

1. Data file:
    - Coordinates
    - Stop names.
2. Bus data:
    - From Stop Name: The starting bus stops.
    - To Stop Name: The destination bus stop.
    - Passengers: The count of passengers on the bus.
    - Ticket Time: The timestamp when the ticket was issued.
    - Bus Number: The unique identifier for the bus.
    - Trip Number: The specific trip the bus is on.
    - Route Name: The name of the bus route.
3. Routes/routeId:
    - Stations: List of stations in the ordered form of the route, indicating the sequence of stops.
    - Station Id: A unique identifier for each station along the route.

Code Explanation

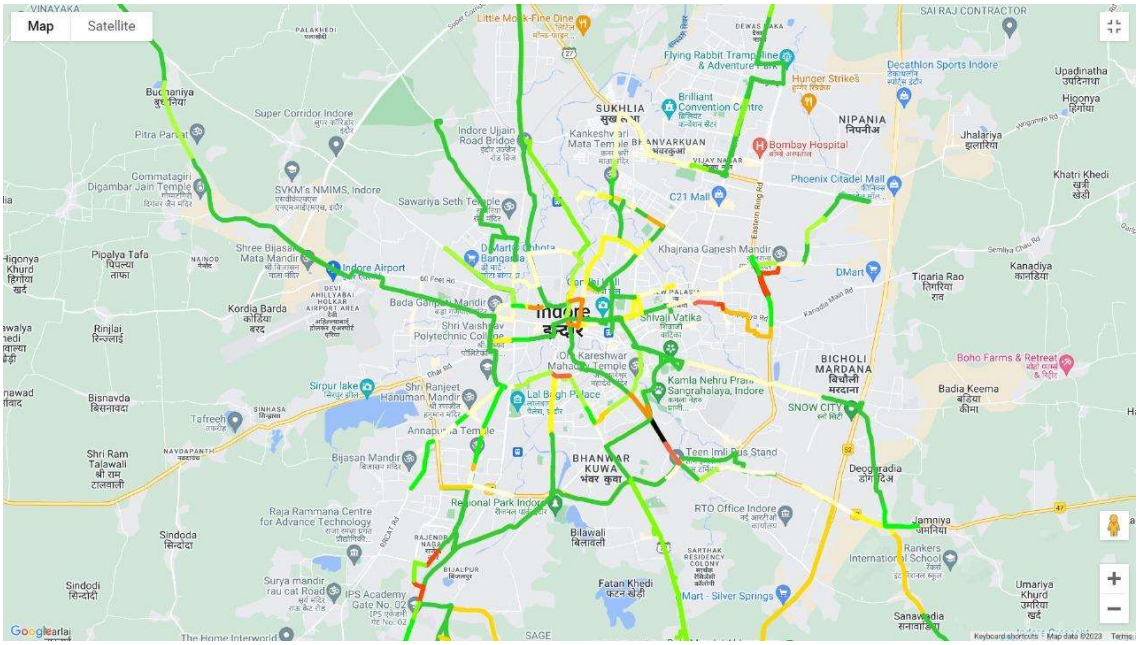
Extracting Route Name, FromStopName, and ToStopName IDs from Bus Data: We need to process the bus data file and extract the route name, from-stop ID, and to-stop ID for each row. The IDs will be useful for mapping coordinates and passenger counts later

Aggregating Passenger Counts in Route File and Preparing Output: We will use the information extracted to update the output dictionary with passenger counts for each route segment.

Adding the coordinates in output for each stop. This updated output will then be sent to the Google Maps API for plotting the routes on google map.

Output –

Fig. 6.3.1 Passenger Density map



Code to count Bus Density:

```
function initMap() {
    var set5492 = new Set();
    var set5488 = new Set();
    var set5471 = new Set();
    var set5537 = new Set();
    var set5463 = new Set();
    var set5457 = new Set();
    var set5499 = new Set();
    var set5479 = new Set();
    var set5446 = new Set();
    var set5497 = new Set();
    var set5609 = new Set();
```

```
if(formattedTime>=starttime&&formattedTime<=endtime&&(bus.rows[j].trip_no%2)==0){

    var str = bus.rows[j].bus_number + bus.rows[j].trip_no;

    if(bus.rows[j].route_name==='C-1'){
        set5492.add(str);
    }
    else if(bus.rows[j].route_name==='C-3'){
        set5488.add(str);
    }
    else if(bus.rows[j].route_name==='M-11'){
        set5471.add(str);
    }
    else if(bus.rows[j].route_name==='M-17'){
        set5537.add(str);
    }
    else continue;
}


for(var i=0;i<temp1.length-1;i++){

    var str = temp1[i].stationname + ' to '+temp1[i+1].stationname;
    output[str][1]+=parseInt(set5492.size);
}
for(var i=0;i<temp2.length-1;i++){
    var str = temp2[i].stationname + ' to '+temp2[i+1].stationname;
    output[str][1]+=parseInt(set5488.size);
}
for(var i=0;i<temp3.length-1;i++){
    var str = temp3[i].stationname + ' to '+temp3[i+1].stationname;
    output[str][1]+=parseInt(set5471.size);
}
for(var i=0;i<temp4.length-1;i++){
    var str = temp4[i].stationname + ' to '+temp4[i+1].stationname;
    output[str][1]+=parseInt(set5537.size);

}



  for(let i in output){
      let value = output[i][1];
      if(value<mi) mi=value;
      if(value>mx) mx=value;
      console.log(value);
      if(value>0){
          const myArray = i.split(" to ");
          var obj = {},obj1={};
          for(let j = 0;j<stations.length;j++){
              if(myArray[0]===stations[j].StationName){
                  var o = {};
                  o['lat']=parseFloat(stations[j].Lat1);
                  o['lng']=parseFloat(stations[j].Long1);
                  obj['origin']=o;
                  obj1['origin']=o;
              }

              if(myArray[1]===stations[j].StationName){
                  var o1 = {};
                  o1['lat']=parseFloat(stations[j].Lat1);
                  o1['lng']=parseFloat(stations[j].Long1);
                  obj['destination']=o1;
                  obj1['destination']=o1;
              }
          }
          obj1['travelMode']= 'DRIVING';
          co.push(obj1);
          obj['wt']=value;
          coor.push(obj);

      }
  }
```

```
var map = new google.maps.Map(document.getElementById('map'), {
    center: { lat: 40.7128, lng: -74.0060 },
    zoom: 6
});

var directionsService = new google.maps.DirectionsService();


// Request and render multiple routes
p=0;
co.forEach(function(request, index) {
    directionsService.route(request, function(response, status) {
        if (status === 'OK') {
            renderRoute(map, response, getRandomColor()); // Get a random color for each route
        } else {
            alert('Directions request failed: ' + status);
        }
    });
});
const paral = document.getElementById("minscale");
const parar = document.getElementById("maxscale");
const node = document.createTextNode(mi);
const node2 = document.createTextNode(mx);

paral.appendChild(node);
parar.appendChild(node2);
document.body.appendChild(paral);
document.body.appendChild(parar);
// console.log(JSON.stringify(output));
const jsonString = JSON.stringify(output, null, 3);

// Create a Blob containing the JSON data
const blob = new Blob([jsonString], {type: 'application/json'});
```

File Description for bus count:

1. Data file:
    - Coordinates
    - Stop names.
2. Bus data:
    - From Stop Name: The starting bus stop.
    - To Stop Name: The destination bus stop.
    - Passengers: The count of passengers on the bus.
    - Ticket Time: The timestamp when the ticket was issued.
    - Bus Number: The unique identifier for the bus.
    - Trip Number: The specific trip the bus is on.
    - Route Name: The name of the bus route.
3. Routes/routeId:
    - Stations: List of stations in the ordered form of the route, indicating the sequence of stops.
    - Station Id: A unique identifier for each station along the route.

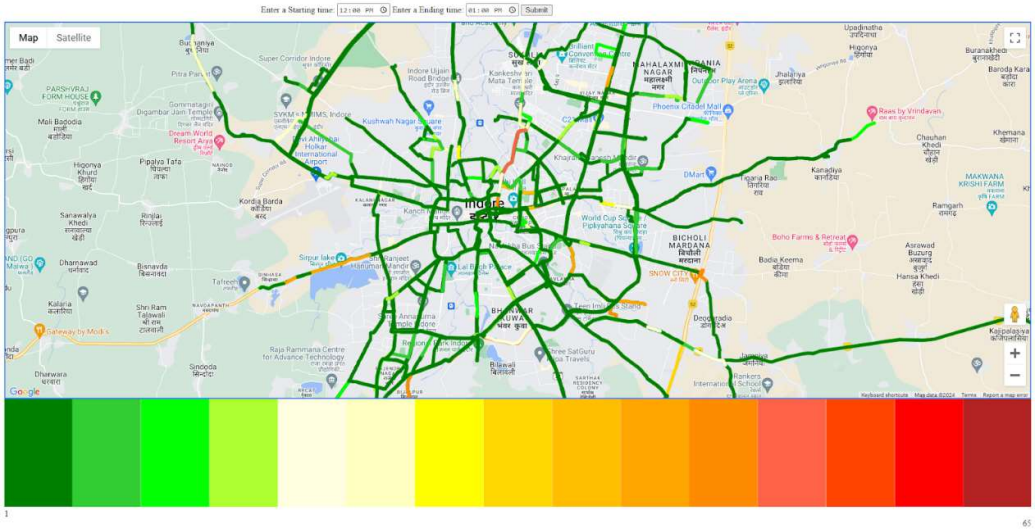
Code Explanation

Generating a collection containing the number of buses assigned to each route. Establishing the number of buses allocated for each stop along the route.
Consolidating bus counts in the route file and preparing the output: Utilizing the gathered data to revise the output dictionary, incorporating passenger and bus tallies for every segment of the route.

Adding the coordinates in output for each stop.This updated output will then be sent to the Google Maps API for plotting the routes on google map.

Output –

Fig. 6.3.2 Bus Density map



Code to count Passenger to Bus density

```
function initMap() {

    starttime = document.getElementById("numbstr").value;
    endtime = document.getElementById("numbend").value;
    console.log(starttime);
    var listOfExceededStops = []
    for(let i in output){
        let value = Math.ceil(output[i][0]/output[i][1]);
        if(value>40) listOfExceededStops.push("\nRoute "+i+" has exceeded passenger limit \t passenger count: " + output[i][0]);
        console.log(value, output[i][1]);
        output[i][0] = value;
        // console.log(output[i][1]);
        if(value<mi) mi=value;
        if(value>mx) mx=value;
        console.log(value);
        if(value>0){

            const myArray = i.split(" to ");
            var obj = {},obj1={};
            for(let j = 0;j<stations.length;j++){
                if(myArray[0]===stations[j].StationName){
                    var o = {};
                    o['lat']=parseFloat(stations[j].Lat1);
                    o['lng']=parseFloat(stations[j].Long1);
                    obj['origin']=o;
                    obj1['origin']=o;
                }

                if(myArray[1]===stations[j].StationName){
                    var o1 = {};
                    o1['lat']=parseFloat(stations[j].Lat1);
                    o1['lng']=parseFloat(stations[j].Long1);
                    obj['destination']=o1;
                    obj1['destination']=o1;
                }
            }
            obj1['travelMode']= 'DRIVING';
            co.push(obj1);
            obj['wt']=value;
            coor.push(obj);

        }
    }
    alert(listOfExceededStops);

    var map = new google.maps.Map(document.getElementById('map'), {
        center: { lat: 40.7128, lng: -74.0060 },
        zoom: 6
    });
```

File Description for passenger to bus count:

1. Data file:
    - GetAllStations - File containing all stations along with coordinates
    - Input file containing count of passengers and bus

Code Explanation

Generating a collection containing the number of buses assigned and passengers count to each route. Establishing the number of passengers to buses traveling for each stop along the route.

Consolidating number of passengers to bus counts in the route file and preparing the output: Utilizing the gathered data to revise the output dictionary, incorporating passenger and bus tallies for every segment of the route.

Adding the coordinates in output for each stops.This updated output will then be sent to the Google Maps API for plotting the routes on google map.

The maximum passenger limit per bus has been established at 40 individuals. In the event that this threshold is surpassed, an alert mechanism is triggered, notifying relevant parties. The alert message will include a comprehensive list of stops along the route, accompanied by their respective passenger counts, enabling prompt action to address the situation.

Output:

Fig. 6.3.3 Alert containing list of routes with exceeded passenger count.

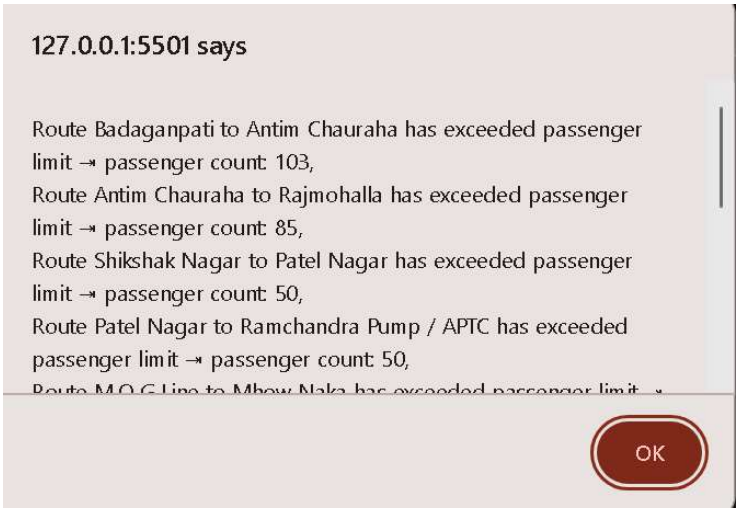
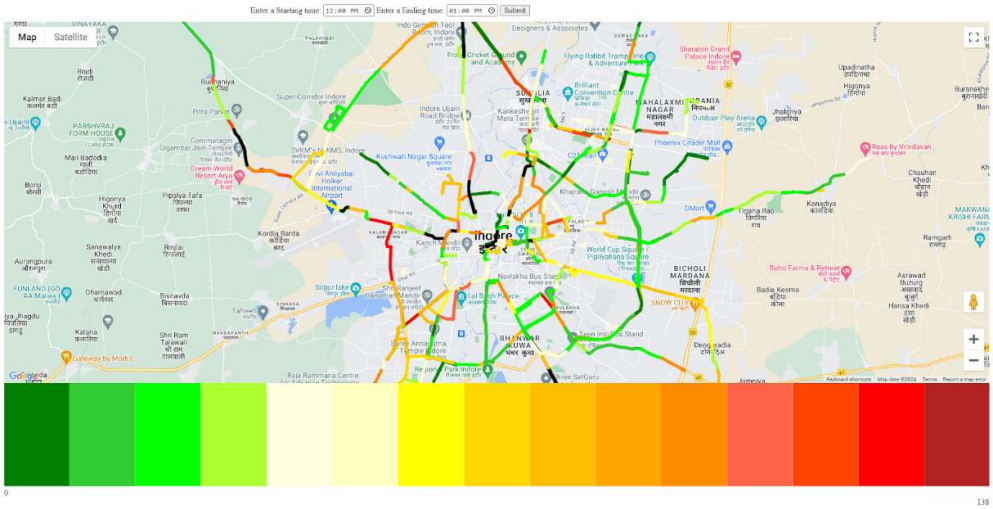


Fig. 6.3.4 Passenger to Bus Density map



Heatmap:

A heatmap is a visual representation of data on google map where values are depicted as colors. In this context, we are utilizing a heatmap to represent the passenger count on various bus stops. The color intensity on the map corresponds to the number of passengers at each stop during a specific time period

File Description:

Data file:

- Coordinates
- Stop names.

Bus data:

- From Stop Name: The starting bus stops.
- To Stop Name: The destination bus stop.
- Passengers: The count of passengers on the bus.
- Ticket Time: The timestamp when the ticket was issued.
- Bus Number: The unique identifier for the bus.
- Trip Number: The specific trip the bus is on.
- Route Name: The name of the bus route.

Code:

```
let map, heatmap;
var starttime=7;
var endtime=8;

import data from './data1.json' assert { type: 'json' };
import bus from './bus_data.json' assert { type: 'json' };

function initMap() {
  starttime = document.getElementById("numbstr").value;
  endtime = document.getElementById("numbend").value;

  map = new google.maps.Map(document.getElementById("map"), {
    zoom: 12,
    center: { lat: 22.719568, lng: 75.857727 },
    mapTypeId: "terrain",
  });

  // Create a <script> tag and set the USGS URL as the source.
  const script = document.createElement("script");

  // This example uses a local copy of the GeoJSON stored at
  // http://earthquake.usgs.gov/earthquakes/feed/v1.0/summary/2.5_week.geojsonp
  script.src =
    "https://developers.google.com/maps/documentation/javascript/examples/json/earthquake_GeoJSONP.js";
  document.getElementsByTagName("head")[0].appendChild(script);
}
function formatTimeWithLeadingZeros(hours, minutes) {
  const formattedHours = String(hours).padStart(2, '0');
  const formattedMinutes = String(minutes).padStart(2, '0');
  return `${formattedHours}:${formattedMinutes}`;
}
```

```
starttime = document.getElementById("numbstr").value;
endtime = document.getElementById("numbend").value;


const heatmapData = [{location: new google.maps.LatLng(22.719871002409814, 75.90613042465931), weight: 6},
  new google.maps.LatLng(22.74938585235775, 75.90352517167092),
  {location: new google.maps.LatLng(22.751096060846255, 75.89520524699515), weight: 2},];
if(starttime<=endtime){
for (let i = 0; i < data.features.length; i++) {
  const coords = data.features[i].geometry.coordinates;
  const stop_name = data.features[i].properties.name;
  var w=0;
  for (let j = 0; j < bus.rows.length-10000; j++) {
      const t = new Date(bus.rows[j].ticket_time);

      const currentHour = t.getHours();
      const currentMinute = t.getMinutes();
      const formattedTime = formatTimeWithLeadingZeros(currentHour, currentMinute);
      if(bus.rows[j].fromstopname==stop_name&&formattedTime>=starttime&&formattedTime<=endtime){
          w+=parseInt(bus.rows[j].passengers);
      }
  }
  const latLng = {location: new google.maps.LatLng(coords[0], coords[1]), weight: w};
  heatmapData.push(latLng);
}
}
else{
  console.log("end time is smaller than start time");
  alert("Enter time correctly")
}
```

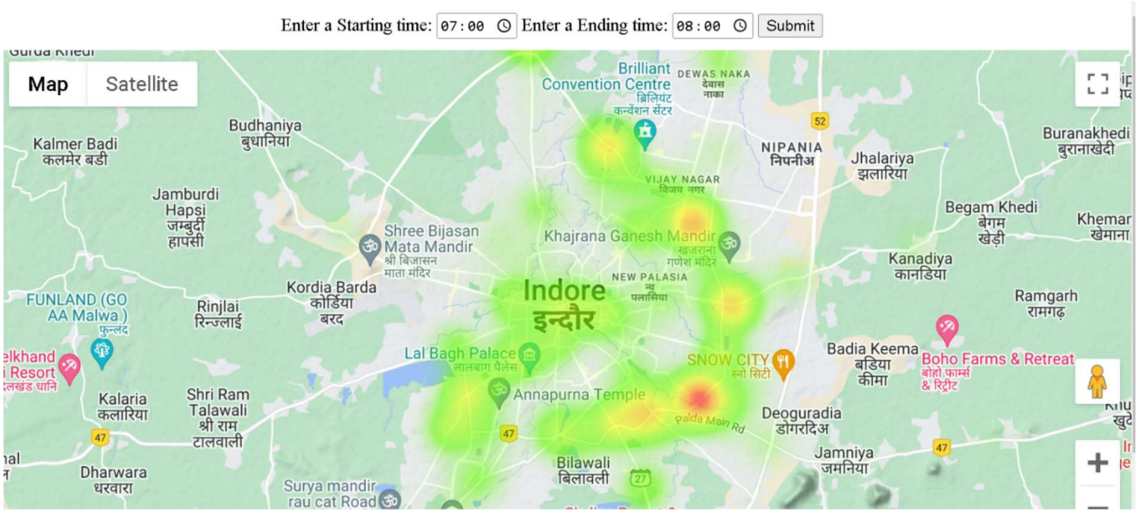
```
  heatmap = new google.maps.visualization.HeatmapLayer({

    data: heatmapData,
    radius:40,
    dissipating: true,
    map: map,
  });

}
```

Code Explanation:

- User Input:
  - The analysis begins with user input for the start and end time.
- Passenger Count Calculation:
  - For each bus stop in the data file, the system counts the number of passengers during the specified time period. This involves filtering the bus data based on the provided time range and summing the passenger counts for each stop.
- Vector Creation:
  - A vector is constructed named "heatmapData" for each bus stop, comprising the following components:
    - Latitude and Longitude: The geographical coordinates of the bus stop.
    - Weight: The counted passenger count during the specified time period.
- Google Maps API Integration:
  - The generated vectors serve as input data for the Google Maps API to plot a heatmap. The coordinates and weights are utilized to visualize the distribution of passenger counts across different bus stops.

Output-

Fig. 6.3.5 HeatMap



6.4 Action Suggestions:

Our proposed approach involves a comprehensive analysis of passenger distribution per bus between each stop. Subsequently, our project provides an alert if a route is exceeding the capacity of passengers-to-bus ratio which helps ensure an equitable spread, effectively maintaining the balance between supply and demand.

How is this solution helpful to the AiCTSL?

The above-mentioned solution offers significant benefits to AICTSL by providing them with an analysis of the existing passenger-to-bus ratio data. Moreover, it sends an alert to them when capacity of bus increases and helps them take action and provide potential enhancements to system efficiency, along with potential improvements in passenger experience, contingent upon the implementation of recommended measures. Our project endeavours to empower the bus transport authority with actionable insights, enabling them to make informed decisions and undertake appropriate measures based on our assistance.

Our Algorithm:

Our algorithm employs a multi-step process to optimize bus routes effectively:

- **Segmentation:** We begin by dividing the map and routes into the smallest feasible segments, typically the paths between consecutive stops. For instance, if the route AE includes intermediate stops B, C, and D (in sequence), we segment it into AB, BC, CD, and DE segments to facilitate more streamlined analysis.
- **Passenger Analysis:** Next, we compute the number of passengers traveling between each pair of stops along these segments.
- **Bus Allocation:** Subsequently, we determine the number of buses currently operating along each segment.
- **Ratio Calculation and Current Status Map Generation:** Utilizing the passenger and bus data, we calculate the ratio of passengers per bus for each

segment, thereby generating a comprehensive map reflecting the current status of the system.

- **Redistribution Strategy:** At this stage, we send alerts if the passenger-to-bus ratio increases the threshold value which is set at 40 (passengers per bus) to guide the redistribution process.
- **Transport authority takes action:** This allows for the transport authority to take action accordingly and ensure better utilization of resources.

Chapter 7

TESTING

We have performed the following types of testing:

- Unit Testing
- Functional Testing

They are as follows:

S.No	Action	Inputs Required	Expected Output	Actual Output	Test Result
1.	Observing a particular route at various time intervals	Route and time interval	The route's passenger densities change according to the input time interval	The passenger densities along the route fluctuate based on the specified time interval	Pass
2.	Verify route data	Route and any time interval	The route shown on map is integrated with the city map and depicts the correct data	The map displays a seamlessly integrated route that aligns with the city map and accurately represents the relevant data	Pass

3.	Observe multiple routes together	Multiple routes at a particular time interval	All routes' data is taken into consideration and map auto adjusts to all overlapping passenger densities, providing a city map with all routes co-existing together at a given time interval	All route data is accounted for, and the map automatically adjusts to accommodate overlapping passenger densities, resulting in a comprehensive city map where all routes coexist harmoniously	Pass
4.	Color testing	Multiple routes and multiple time intervals	Map is checked on various time intervals with multiple routes and the passenger density is shown according to the color schema defined	The map undergoes checks at various time intervals, incorporating multiple routes, and displays passenger density according to the predefined color scheme	Pass

5.	Time interval testing	Time interval and route	The map is observed with changing time interval slots (i.e. 15 min, 30 min, 1 hour, 2 hour) and the map adjusts accordingly	The map is monitored across different time intervals (e.g., 15 minutes, 30 minutes, 1 hour, 2 hours), and it adjusts accordingly to reflect the changes	Pass
6.	Route data change testing	Route and time	The map reflects changes in any route data in the backend and adjusts to it accordingly	The map dynamically updates to reflect any changes in route data from the backend	Pass
7.	Data validation	Route Data	Validate the input data to ensure it meets the expected format and quality. Check for missing values, outliers, or inconsistencies in the passenger data, to ensure that our	Verified the input data to confirm it adheres to the anticipated format and quality standards. Scrutinized for absent values, anomalies, or discrepancies within the passenger data,	Pass

			visualization accurately reflects the underlying data	ensured that our visualization faithfully represents the underlying information	
8.	Alert verification	Multiple routes at a particular time interval	Alert pops up if a route is getting overcrowded at any time interval	A list showing all the overcrowded routes at that particular time interval shows up on the screen	Pass

Conclusion/Summary:

In conclusion, our testing endeavors have substantiated the robustness and effectiveness of our transportation route visualization system across a spectrum of scenarios. From observing individual routes to validating data integrity and assessing dynamic adjustments, each test has affirmed the system's capability to accurately represent and adapt to real-world transportation dynamics. With successful outcomes across all testing criteria, we stand assured of delivering a reliable tool that empowers transportation authorities to make informed decisions and enhance commuter experiences.

Chapter 8

CONCLUSION

8.1 Conclusion

- Ultimately, through a diverse range of maps and the alert mechanism triggered by bus route overcrowding, transportation authority administrators gain access to a dynamic dashboard. This tool enables them to promptly respond and optimize resource utilization for greater efficiency.

- Moreover, the data analysis of the bus routes has provided valuable insights into passenger patterns and fluctuations throughout the day. The identification of specific high-frequency stops, such as Rajwada and Teen Imli, as well as the recognition of time-dependent variations, allows for a strategic reconsideration of the existing bus route plan. The analysis suggests that optimizing the route by eliminating stops with consistently low passenger counts and adjusting bus frequency dynamically based on temporal variations can enhance overall operational efficiency.

8.2 Future Enhancement

1. Developing a user-friendly mobile app for real-time bus tracking, providing passengers with accurate arrival times and dynamically updated routes.

2. Integrating smart technologies at bus stops to relay real-time information on bus arrivals, optimizing passenger flow and reducing wait times.

3. Implementing predictive analytics algorithms to forecast peak passenger times, enabling proactive adjustments to bus schedules and frequencies.

REFERENCES

1. AiCTSL official website - https://aictslportal.infinium.management/
2. Google Maps routes API - https://developers.google.com/maps/documentation/routes
3. Heat Map API - https://developers.google.com/maps/documentation/javascript/examples/layer-heatmap
4. Smart City Office ridership data
5. JavaScript - https://www.w3schools.com/js/
6. Pandas - https://www.w3schools.com/python/pandas/default.asp