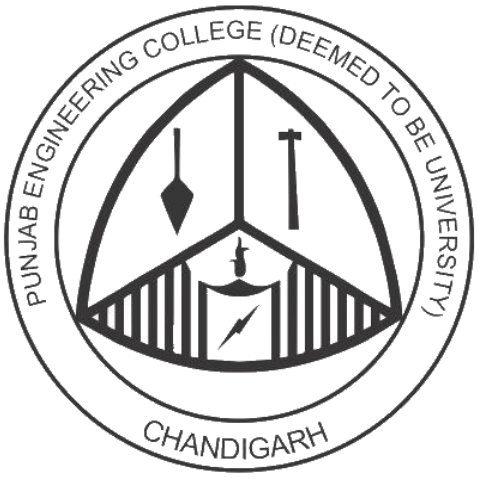

Rush Hour

Major Project Report

Submitted By:

Mayank	19103042
Raman Ailawadhi	19103061
Rahul Jindal	19103033
Deepak Sharma	19103003

Under the Supervision of:

Dr. Sudesh Rani
Faculty
Department of Computer Science and Engineering
Punjab Engineering College
(Deemed to be University)
Chandigarh

DECLARATION

We hereby declare that the project work entitled “Rush Hour” is an authentic record of our own work carried out at Punjab Engineering College (Deemed to be University), as a requirement of Major Project for the award of degree of BTech (Computer Science and Engineering), under the guidance of Dr. Sudesh Rani (Faculty, Department of Computer Science and Engineering), during January 2023 to May 2023.

Dated: 24/05/2023

Mayank	**19103042**
Raman Ailawadhi	**19103061**
Rahul Jindal	**19103033**
Deepak Sharma	**19103003**

Certified that the above statement made by the student is correct to the best of my knowledge and belief.

Dr. Sudesh Rani
Professor
Department of Computer Science
Punjab Engineering College
Chandigarh

CERTIFICATE

This is to certify that the project entitled “Rush Hour” by Mayank, Raman Ailawadhi, Rahul Jindal and Deepak Sharma is an authentic record of the work carried out under the supervision of Dr. Sudesh Rani, Computer Science and Engineering Department, Punjab Engineering College (Deemed to be University), Chandigarh in the partial fulfillment of the requirements as a part of Major Project for the award of 06 credits in semester 8 of the degree of Bachelor of Technology in Computer Science and Engineering. I certify that the above statement made by the students is correct to the best of my knowledge and belief.

Dr. Sudesh Rani

Professor

Department of Computer Science

Punjab Engineering College

Chandigarh

ACKNOWLEDGEMENT

We have taken a lot of deliberations on this venture. But it wouldn't have been possible without the help and backing of **Dr. Sudesh Rani, Dr. Sanjay Batish, Mr. Prabhsimran Singh and Mr. Sushil**. We want to extend our true appreciation and thank them. We take this opportunity to express our profound gratitude and deep regards to our mentor Dr. Sudesh Rani for her exemplary guidance, monitoring and constant encouragement throughout the course of this project.

This project truly wouldn’t have been possible without her mentorship.

ABSTRACT

As urban populations and automobile numbers continue to grow, traffic congestion is becoming a pressing issue. The adverse effects of traffic jams are not limited to causing delays and stress for drivers, but also contributing to increased fuel consumption and air pollution. Megacities are particularly impacted by this problem, and it is crucial to calculate road traffic density in real-time to enable effective traffic management and signal control. Given the continuous rise of traffic congestion, optimizing traffic control has become necessary to meet increasing demand. The traffic controller plays a critical role in ensuring smooth traffic flow. Our proposed system focuses on utilizing live images from traffic junction cameras to calculate traffic density using advanced image processing and AI techniques. The system concentrates on developing an algorithm that can switch traffic lights based on the count of vehicles and provide green-corridor to priority vehicles in order to minimize congestion and reduce travel time, ultimately promoting sustainability by curbing pollution and offering prompt resolutions for urgent circumstances. Furthermore, the system assigns trust scores to various roads based on factors such as congestion and weather conditions, providing passengers with an indication of the reliability of different routes.

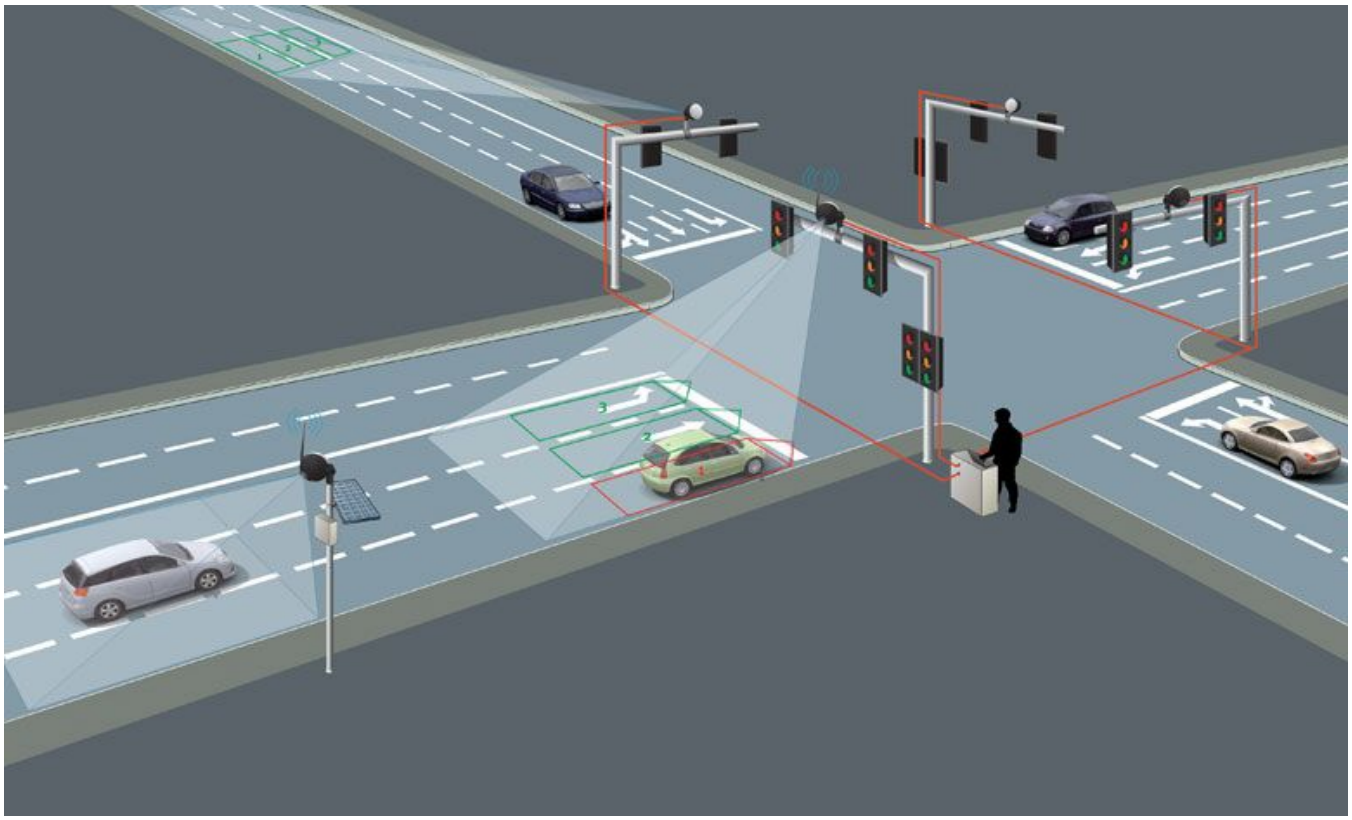

Fig 1: Smart Traffic Management Visualization

TABLE OF CONTENTS

LIST OF ABBREVIATION

ATCS	Adaptive Traffic Control System
AWS	Amazon Web Services
CCTV	Closed Circuit Television
CNN	Convolutional Neural Network
CSS	Cascading Style Sheets
DOM	Document Object Model
EC2	Elastic Compute Cloud
GST	Green Signal Time
HTML	Hypertext Markup Language
I/O	Input Output
ITCS	Improved Traffic Control System
JSON	Javascript Object Notation
SQL	Structured Query Language
YOLO	You Only Look Once

LIST OF FIGURES

CHAPTER 1 : INTRODUCTION

1.1 Introduction

The rise in urban vehicle population has led to challenges in road networks, particularly concerning reduced road capacity and declining level of service. These issues often stem from the use of fixed signal timers in traffic control systems at intersections. These timers follow a repetitive pattern of phase sequences and fixed durations. As the demand for road capacity continues to grow, there is a pressing need for innovative traffic control solutions that fall under the realm of Intelligent Transport Systems.

1.2 ITCS - Improved Traffic Control System

The ITCS (Improved Traffic Control System) is designed to alleviate traffic congestion, reduce delays, and optimize travel time for commuters. Our system incorporates a Signal Switching algorithm that dynamically regulates traffic flow by analyzing the number and types of vehicles present at each arm of a crossroad, taking into account the heterogeneous nature of the traffic.

The Signal Switching Algorithm efficiently allocates green signal time based on real-time traffic density information obtained from the vehicle detection module. It dynamically adjusts the red signal timers of other signals to synchronize traffic movement effectively. The algorithm operates in a cyclic manner, seamlessly transitioning between signals according to the predetermined timers.

To execute the algorithm, it receives data regarding detected vehicles from the detection module as input. This information is processed to calculate the total count of vehicles for each vehicle class. Based on this analysis, the algorithm determines the appropriate green signal duration for the signal in question and adjusts the red signal timings of other signals accordingly. This algorithm can be easily scaled to accommodate any number of signals at an intersection.

1.3 YOLO Model and Simulation Development

The proposed system utilizes YOLO (You Only Look Once) for efficient and accurate vehicle detection. YOLO is a convolutional neural network (CNN) specifically designed for real-time object detection. Unlike traditional methods, YOLO applies a single neural network to the entire image and divides it into regions to predict bounding boxes and probabilities. These bounding boxes are then weighted based on the predicted probabilities. YOLO stands out for its ability to achieve high accuracy while operating in real-time scenarios.

By employing YOLO, our system benefits from a single CNN that can predict multiple bounding boxes and class probabilities simultaneously. To optimize processing time, the backbone CNN used in YOLO can be further simplified. Furthermore, we have developed a simulator using the Pygame library in Python, which incorporates the ITCS (Improved Traffic Control System). This simulator provides a demonstration of the entire model and showcases the dynamic movement of traffic as the green light timers are calculated using the ITCS algorithm. It serves as a valuable tool to visualize the functioning of the system and its impact on traffic flow in real-time scenarios.

1.4 Priority Vehicle Module

We have developed a Priority Vehicle Algorithm which helps provide green-corridor to priority vehicles while making sure that traffic on other lanes does not have to wait very long and embedded it in the Pygame Simulator. Four separate cases of priority vehicle occurrence were developed to make the algorithm more efficient. Furthermore we have compared the waiting times of the ITCS Algorithm with the ITCS+Priority Algorithm.

The objective is to improve the movement of emergency vehicles, public transport, and other priority vehicles through intersections, ensuring their swift and safe passage while minimizing disruption to regular traffic. The proposed algorithm builds upon the foundation of the Improved Traffic Control System (ITCS), which is designed to optimize traffic signal timings based on real-time traffic conditions.

This research project involves a comparative analysis between the ITCS algorithm and the ITCS+Priority Vehicle algorithm. The key performance metric used for comparison is the waiting time experienced by priority vehicles at intersections. By evaluating the waiting times under both algorithms, we can assess the effectiveness of the ITCS+Priority Vehicle algorithm in providing faster and more efficient passage for priority vehicles.

1.5 Trust Score Module

The Trust Score Algorithm provides users with a trust score for each lane, indicating the reliability and safety of the lanes ahead. It incorporates real-time weather conditions, traffic congestion data, and historical accident information to calculate the trust scores.

The algorithm helps drivers make informed decisions by considering factors such as adverse weather conditions, traffic congestion levels, and the safety history of each lane. It serves as a tool to assess the relative trustworthiness of different paths, allowing drivers to choose the most suitable lane based on their preferences and circumstances.

1.6. Hardware Module

In this project, we have undertaken the task of implementing a hardware system to simulate and display all 12 real-time traffic lights for a junction with four lanes. To accomplish this, we have utilized the Pygame simulator, which has been exported to a Raspberry Pi 4 and connected to a breadboard. Additionally, LED lights have been employed as the means of visually representing the traffic lights.

The implementation of this hardware system provides a tangible representation of the simulated traffic lights, allowing for a more immersive and realistic experience. By incorporating physical LED lights, we enhance the usability and practicality of the simulation, enabling users to visually observe the changing states of the traffic lights at the junction.

The Raspberry Pi 4 serves as the central processing unit for the hardware implementation. Its computational capabilities allow for efficient execution of the simulation, ensuring real-time responsiveness and accurate synchronization of the LED lights with the simulated traffic light states. With its compact form factor and versatile GPIO (General Purpose Input/Output) pins, the Raspberry Pi 4 acts as an ideal platform for integrating the hardware components seamlessly.

1.7 Mobile Application Development

Leveraging the power of React Native and Firebase technologies, our mobile application aims to enhance the efficiency and effectiveness of ambulance operations by providing essential functionalities and real-time information. The primary objective of our mobile application is to enable ambulance drivers to navigate through congested traffic more effectively. To achieve this, we have incorporated GPS functionality, allowing us to track the precise location of the ambulances in real-time. By utilizing this information, we can dynamically analyze traffic patterns and provide ambulance drivers with optimal routes to reach their destinations swiftly and safely.

One of the key features of our mobile application is the emergency state notification system. We have implemented a dedicated button that ambulance drivers can easily access and tap to indicate that they are in an emergency situation. This crucial input triggers a chain of events within our system, enabling our Pygame Simulator to generate a green corridor for the ambulance, ensuring a smooth and unhindered passage through traffic. Furthermore, our mobile application provides ambulance drivers and normal users with vital information about the trust scores of lanes ahead of them. This feature assists drivers in making informed decisions regarding their chosen path.

CHAPTER 2: BACKGROUND

Our main objective is to build a system that can regulate the real time heterogeneous traffic so as to let more vehicles cross the crossroad and save time for the commuters. The research done on the traffic control system is mentioned in Table 1.

Table 1. Research Work

S. No.	Research Paper Title	Publication	Remarks
1	De Oliveira, L. F. P., Manera, L. T., & da Luz, P. D. G, Development of a Smart Traffic Light Control System with Real-Time Monitoring [1]	IEEE Internet of Things Journal, Vol. 14, No. 8, 2020	• Development of wireless traffic light control system using XMesh network technology. • Utilization of 2.4 GHz ISM non-licensed frequency band with transfer rate up to 2 Mbps. • Implementation of redundant network technology for reliable communication. • Features include a network coordinator device, initialization routine and synchronization of traffic lights.
2	Kumar, N., Rahman, S. S., & Dhakad, Fuzzy Inference Enabled Deep Reinforcement Learning-Based Traffic Light Control for Intelligent Transportation System [2]	IEEE transactions on intelligent transportation systems, 2020, from IEEE Xplore	• Adjusts the durations based on real-time traffic information. • Combines deep reinforcement learning and fuzzy logic, optimal actions to reduce average waiting time using real-time data. • Adapts to different traffic scenarios, prioritizes emergency vehicles, reduces waiting time, and improves overall traffic management.

3	Karmakar, G., Chowdhury, A., Kamruzzaman, J., & Gondal, I. (2020). A Smart Priority Based Traffic Control System for Emergency Vehicles [3]	IEEE Sensors Journal, Volume: 21, Issue: 14, 15 July 2021	• The smart system assigns priority codes to emergency vehicles based on incident severity and estimates cell clearance time to reduce travel time. • Traffic conditions and clearance times at signals used to optimize the route for emergency vehicles. • Simulation results using SUMO demonstrate that increasing interventions can decrease travel time for emergency vehicles.
4	Sundar, R., Hebbar, S., & Golla, V. (2015). Implementing Intelligent Traffic Control System for Congestion Control, Ambulance Clearance, and Stolen Vehicle Detection [4]	IEEE SENSORS JOURNAL, VOL. 15, NO. 2, FEBRUARY 2015	• The system uses RFID technology to track congestion volume and adjust signal timings accordingly to reduce congestion. • The integration of ZigBee technology allows emergency vehicles to change traffic lights to green, enabling them to reach their destinations quickly and efficiently. • By comparing RFID tags with a list of stolen vehicle RFIDs, the system can identify stolen vehicles and alert the authorities promptly.
5	Qmulus – A Cloud Driven GPS Based Tracking System for Real-Time Traffic Routing [7]	IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) ,Vol:7, No:1, 2013	• The cloud-based GPS system that provides possible path from source to destination with priority/score considering all the adversities like road blockage, weather etc. to driver through mobile app. • Also used unsupervised learning to predict unforeseen accidents to update score of the path.

6	Real Time Traffic Light Control Using Image Processing [8]	Indian Journal of Computer Science and Engineering (IJCSE), ISSN : 0976-5166, Vol:2, No:1, 2011	● The cloud-based GPS system that provides possible path from source to destination with priority/score considering all the adversities like road blockage, weather etc. to driver through mobile app. ● Also used unsupervised learning to predict unforeseen accidents to update score of the path.
7	IoT-based Traffic Signal Control for ambulance [9]	International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020	● Used sound sensors to detect and control traffic lights for that lane. ● Used 2 sound sensors: one at traffic pole itself and another at 120m from traffic light pole. ● When sound detected at 120m, it will send turn the light green and when ambulance crosses traffic pole, sound sensor indicates to return back to normal.
8	Smart Traffic Management System [10]	International Journal of Research Publication and Reviews Vol (2) Issue (2) (2021) Page 226-228	● If RFID reader at traffic pole detects ambulance, it will ask driver through mobile app for acknowledgement to turn the lights green or not. ● 2nd is driver itself is making API call through mobile app and request is passed on to respective traffic light arduino and lights are turned green.
9	Study Of Automatic Traffic Signal System For Chandigarh [26]	International Journal Of Engineering Sciences & Research Technology, (I2or), Publication Impact Factor: 3.785, July, 2015	● Inductive loop used to detect presence and speed of vehicles ● Distance of inductive loop from junction and approach speed of vehicles used to then estimate the addition to green time until max green time is reached or no further vehicles are detected. ● Comparisons made with preset green signal times.

10	Intelligent Traffic Management Systems: A Review [27]	International Journal For Innovative Research In Science & Technology, Volume 2, Issue 09, February 2016	• Different intelligent traffic management systems compared. • VANET is efficient and can provide accident information, but GPS has to be installed on every device. • RFID tags can be used for priority vehicles. • Fuzzy logic takes number of vehicles and average speed of vehicles as input, can become very complex when more parameters involved. • Infrared sensors and Video cameras are expensive.

2.1 Existing Systems

Contact was made with Smart City Chandigarh Project Manager to know about the existing Traffic Control System in Chandigarh and gathered the information regarding the same. We got to know that there is already an existing intelligent traffic control system known as ATCS i.e. Adaptive traffic control system which measures the traffic queue length and predicts the green time.

The average length between two crossroad junctions is 600 meters. For the first 20 meters of an arm of the crossroad, they have employed the traditional inductive loop technology to measure the traffic and after 20m, they have deployed a camera that will measure queue length on the basis of which the system will predict the green time. Then they will add the time to get the resultant green time.
If no vehicle crosses the inductive loop when there is green light for over 5 sec, they will turn the signal to red which will help them to save the time and that saved green signal time will be propagated to other junctions to reduce their respective red signal times.
But this system has few limitations as given below:

1. They are calculating the queue length rather than considering the number of the vehicles which leads to inaccurate approximation of the green time.
2. Heterogeneity of traffic is not considered i.e. type of vehicles like bus, car, truck, etc.
3. They are using traditional inductive loop technology to sense the presence of vehicles in the 20m vicinity of the traffic lights before skipping the green signal time for that junction.
4. Current system uses RFID to find an ambulance in the vicinity of the traffic junction that has a much shorter range.

CHAPTER 3: PROPOSED WORK

(Algorithm/Model/Approach)

Our proposed system takes an image from the CCTV cameras at traffic junctions as input for real-time traffic density calculation using image processing and object detection. This system can be broken down into 7 modules: Vehicle Detection module, Signal Switching module, Simulation module, Priority Vehicle module, Trust Score module, Hardware module and Mobile Application module. As shown in the Fig 2 below, this image is passed on to the vehicle detection algorithm, which uses YOLO. The number of vehicles of each class, such as car, bike, bus and truck, is detected, which is used to calculate the density of traffic.

The signal switching algorithm uses this density to set the green signal timer for each lane. The red signal times are updated accordingly. The green signal time is restricted to a maximum and minimum value in order to avoid starvation of a particular lane.

A simulation is also developed to demonstrate the system's effectiveness and compare it with the existing static system. In addition to the aforementioned modules, the system incorporates the Trust Score Module, which considers real-time congestion, weather data, and hotspot parameters to assign a trust score to each lane. This feature enables users to make informed decisions based on the trustworthiness of a specific lane.

Furthermore, the Hardware Module enhances the system by providing physical traffic lights that simulate real-world conditions. This integration adds a tangible element to the simulation, creating a more immersive and realistic experience for users.

Additionally, the Mobile Application Module plays a crucial role in the system. It offers priority vehicle drivers the ability to request a green corridor, ensuring a smooth and uninterrupted passage through the traffic. Moreover, the mobile app also integrates the trust functionality, allowing users to access and evaluate the trust scores assigned to different roads, aiding in their decision-making process.
Overall, these modules collectively contribute to an advanced traffic management system that combines real-time data analysis, hardware simulation, and mobile application integration to optimize traffic flow, prioritize emergency vehicles, and provide users with valuable information for navigating the road network effectively.

3.1 Vehicle Detection Module

A pretrained YOLO model is used for vehicle detection which can detect vehicles of different classes like cars, motorbikes, heavy vehicles (buses and trucks).

- In Fig 2, the first step is to capture images and then apply vehicle detection techniques to classify the vehicles.
- The YOLO model processes captured images and predicts bounding boxes and probabilities for vehicles of various classes, including cars, motorbikes, and heavy vehicles like buses and trucks.
- By accurately identifying and classifying vehicles, we can utilize this information for calculating traffic density, analyzing traffic patterns, and making informed decisions for traffic control and optimization in our traffic management system.

3.2 Signal Switching Algorithm

The Signal Switching Algorithm sets the green signal timer according to traffic density returned by the vehicle detection module, and updates the red signal timers of other signals accordingly. It also switches between the signals cyclically according to the timers. The algorithm takes the information about the vehicles that were detected from the detection module, as explained in the previous section, as input.

This is in JSON format, with the label of the object detected as the key and the confidence and coordinates as the values. This input is then parsed to calculate the total number of vehicles of each class. After this, the green signal time for the signal is calculated and assigned to it, and the red signal times of other signals are adjusted accordingly. The algorithm can be scaled up or down to any number of signals at an intersection.

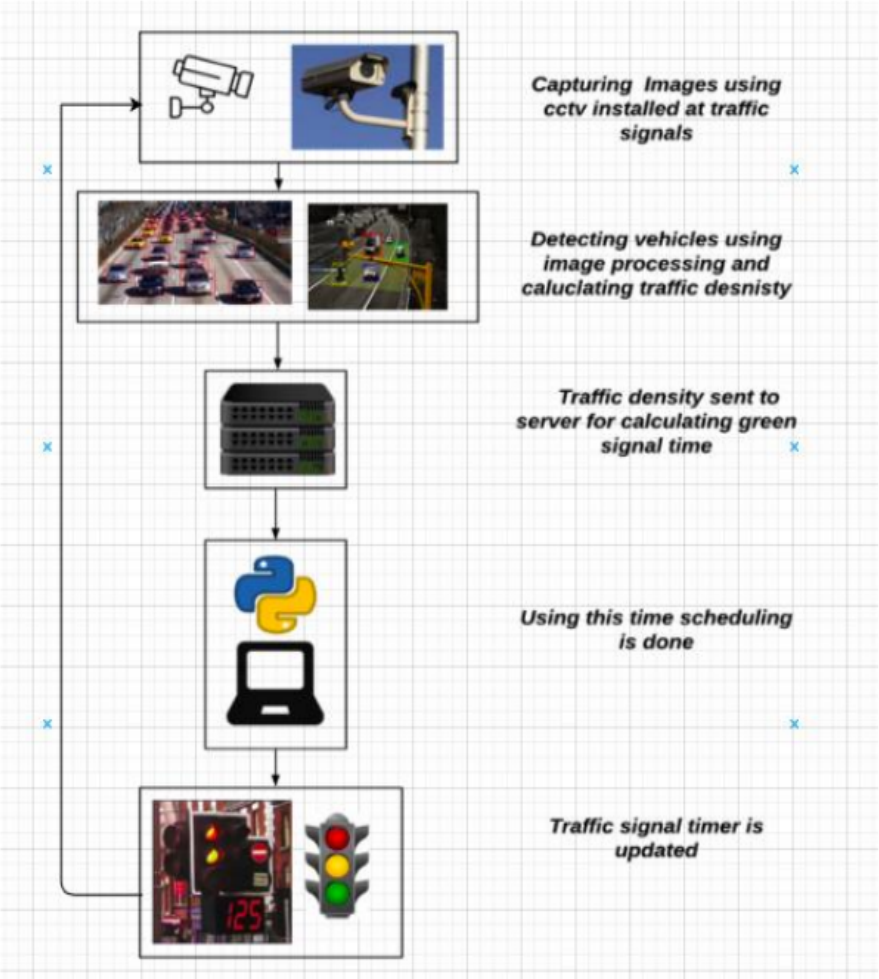

Fig 2: Workflow

3.3 Simulation Module

A simulation was developed from scratch using Pygame to simulate real-life traffic. It assists in visualizing the system and comparing it with the existing static system. The traffic on each lane is simulated based on the real time traffic congestion on the four lanes connected to the selected junction. It contains a 4-way intersection with 3 traffic signals on each lane. Each signal has a timer on top of it, which shows the time remaining for the signal to switch from green to yellow, yellow to red, or red to green.

Each signal also has the number of vehicles that have crossed the intersection displayed beside it. Vehicles such as cars, bikes, buses, trucks, and rickshaws come in from all directions. In order to make the simulation more realistic, some of the vehicles in the rightmost lane turn to cross the intersection.

Whether a vehicle will turn or not is also set using random numbers when the vehicle is generated. It also contains a timer that displays the time elapsed since the start of the simulation.

3.4 Priority Vehicle Module

The priority vehicle algorithm is designed to address different scenarios involving priority vehicles, ensuring efficient traffic management while providing a green corridor for these vehicles. It is divided into four distinct parts, each tailored to handle specific situations.

- Priority Vehicle approaching a current green signal
- Priority Vehicle approaching a current yellow signal
- Priority Vehicle approaching a current red signal
- Priority Vehicle approaching the next green signal

By dividing the priority vehicle algorithm into these four parts, traffic management is optimized, and the green corridor is provided to priority vehicles while ensuring minimal disruption to the regular traffic flow. This comprehensive approach allows for efficient handling of various priority vehicle scenarios, contributing to effective emergency response and overall road safety.

3.5 Trust Score Module

The trust score module consists of the trust score algorithm, which functions by considering several key factors to calculate the trust score for each lane. Firstly, it utilizes real-time congestion data obtained from Google, which provides insights into the current traffic conditions between the starting and end points of the lanes. This data helps determine the level of congestion, allowing users to assess the potential delays and plan their routes accordingly.

Secondly, the algorithm incorporates real-time weather data obtained from OpenWeather between the starting and end points of the lanes. By considering weather conditions such as rainfall, snowfall, visibility, the algorithm can assess the impact of weather on the road conditions and overall safety. This information is particularly valuable during inclement weather, as it allows users to choose safer routes that may be less affected by adverse weather conditions.

Lastly, the algorithm takes into account historical accidents data for the specific junction. By analyzing past accident records, it identifies patterns and trends associated with each lane, allowing for a more accurate assessment of its safety. This historical data provides insights into accident-prone areas, enabling users to make informed decisions to avoid such areas whenever possible. By combining these factors, the trust score algorithm calculates a numerical trust score for each lane connected to the

junction. This score serves as an indication of the relative safety and reliability of the respective lanes, helping users make well-informed decisions about their route selection.

3.6 Hardware Module

The hardware module utilizes a Raspberry Pi 4 to operate the Pygame simulator, which is responsible for simulating a traffic junction. Connected to the Raspberry Pi is a breadboard, which is equipped with 12 LED lights. These LED lights are strategically wired to display the three traffic light signals - red, yellow, and green - for each of the four lanes connected to the simulated junction. The LED lights are synchronized with the Pygame simulator, ensuring that their state changes correspond accurately to the simulation.

3.7 Mobile Application Module

The mobile application incorporates three key features to enhance its functionality and assist priority vehicles:

- GPS Functionality: The application integrates GPS technology to accurately track the location of priority vehicles. This information is crucial for the Rush Hour system to precisely determine the vehicle's position and optimize traffic management accordingly.
- Green Corridor Button: The application includes a dedicated button for priority vehicle drivers. When activated by clicking this button, a signal is transmitted to the Rush Hour system. This prompt notifies the system to establish a green corridor, ensuring a smooth passage for the priority vehicle. The green corridor remains active until the ambulance successfully traverses the targeted junction.
- Trust Score: The application features a dedicated page that provides users with access to the trust scores of all lanes connected to any given junction. By default, the page displays the trust scores for the junction that the vehicle is currently heading towards. This feature empowers users with valuable insights into the reliability and safety of different lanes, enabling informed decision-making for route selection.

Overall, these three features within the mobile application contribute to improved navigation and traffic management for priority vehicles. The integration of GPS functionality, the ability to request a green corridor, and the provision of trust scores enhance the efficiency and safety of emergency response operations.

CHAPTER 4: IMPLEMENTATION

4.1 Vehicle Detection Module

YOLO is popular because it achieves high accuracy while also being able to run in real-time.

- The algorithm applies a single neural network to the full image for ex. Fig 3, and then divides the image into regions and predicts bounding boxes and probabilities for each region. These bounding boxes are weighted by the predicted probabilities like in Fig 4.
- It requires only one forward propagation pass through the neural network to make predictions. After non max suppression (which makes sure the object detection algorithm only detects each object once), it then outputs recognized objects together with the bounding boxes.
- The model runs on pre-trained weights downloaded from the YOLO website. Further configuration of the .cfg file used for training can be changed in accordance with the specifications of our model. The number of output neurons in the last layer can be set to be equal to the number of classes the model is supposed to detect by changing the 'classes' variable. In our system, this was 5 viz. Car, Bike, Bus,Truck, and Rickshaw. Further enhancements to the model can be done by training on high priority vehicles like ambulances, fire brigades and police vans.

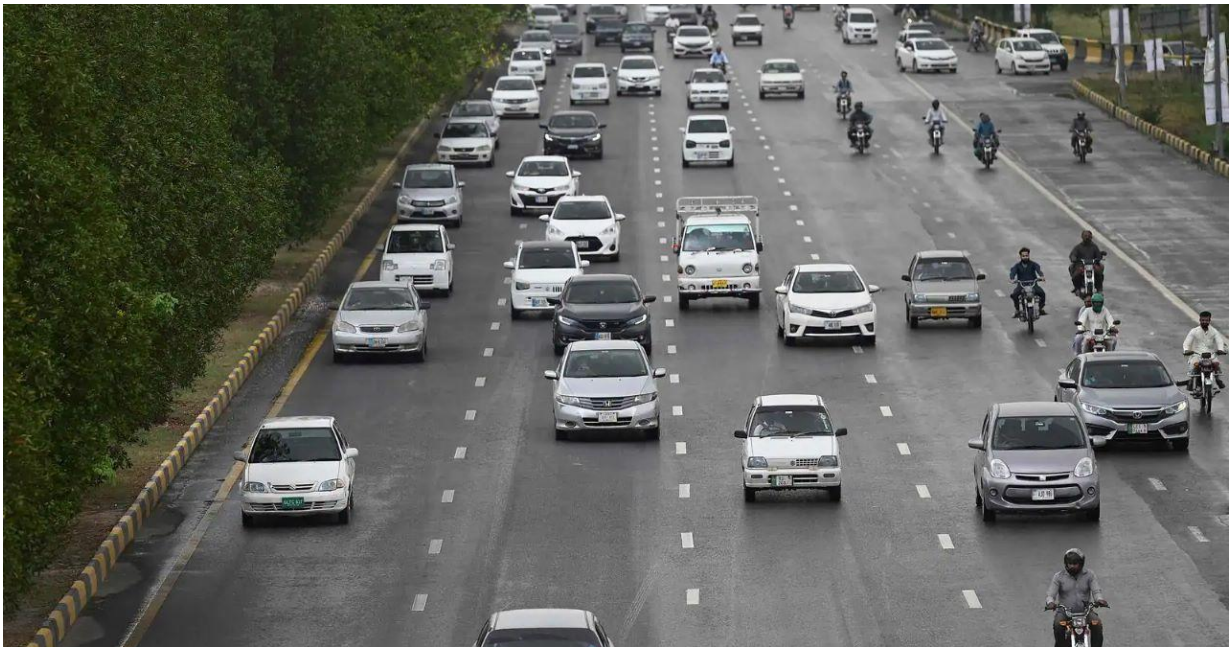

Fig 3: Traffic Example

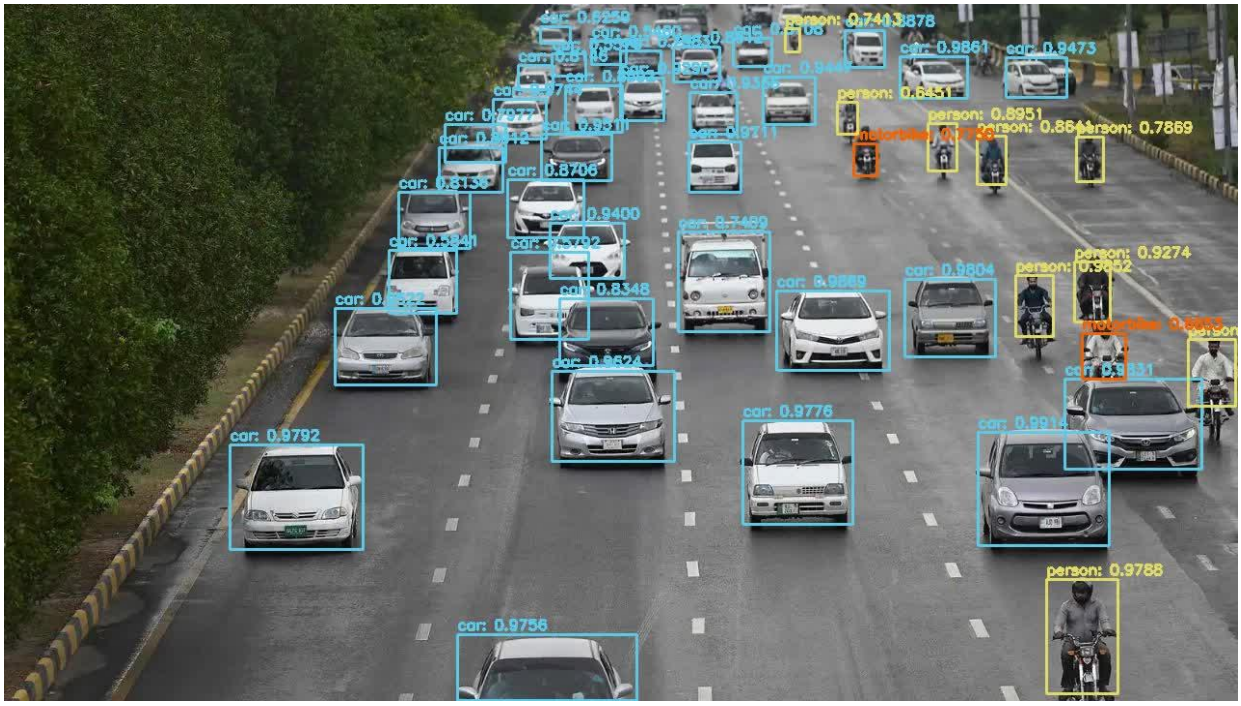

Fig 4: Traffic Detection with YOLO

4.2 Signal Switching Algorithm

Signal Switching algorithm dynamically changes the green light time for an arm of the crossroad.
The following factors were considered while developing the algorithm:

1. The processing time of the algorithm to calculate traffic density and then the green light duration, this decides at what time the image needs to be acquired
2. Number of lanes.
3. Total count of vehicles of each class like cars, trucks, motorcycles, etc.
4. Traffic density calculated using the above factors.
5. Time added due to lag each vehicle suffers during start-up and the non-linear increase in lag suffered by the vehicles which are at the back.
6. The average speed of each class of vehicle when the green light starts i.e. the average time required to cross the signal by each class of vehicle
7. The minimum and maximum time limit for the green light duration - to prevent starvation.

Working of the algorithm

When the algorithm is first run, the default time is set for the first signal of the first cycle and the times for all other signals of the first cycle and all signals of the subsequent cycles are set by the algorithm. A separate thread is started which handles the detection of vehicles for each direction and the main thread

handles the timer of the current signal. When the green light timer of the current signal (or the red light timer of the next green signal) reaches 0 seconds, the detection threads take the snapshot of the next direction. The result is then parsed and the timer of the next green signal is set. All this happens in the background while the main thread is counting down the timer of the current green signal. This allows the assignment of the timer to be seamless and hence prevents any lag. Once the green timer of the current signal becomes zero, the next signal becomes green for the amount of time set by the algorithm.

The image is captured when the time of the signal that is to turn green next is 0 seconds. This gives the system a total of 5 seconds (equal to value of yellow signal timer) to process the image, to detect the number of vehicles of each class present in the image, calculate the green signal time, and accordingly set the times of this signal as well as the red signal time of the next signal. The green signal time is then calculated using the formula below.

$$\textbf{GST} = \frac{\sum_{vehicleClass} (NoOfVehicles)*(AverageTime)}{(NoOfLanes + 1)}$$

where:

- GST is green signal time
- NoOfVehicles is the number of vehicles of each class of vehicle at the signal as detected by the vehicle detection module.
- averageTime is the average time the vehicles of that class take to cross an intersection.
- NoOfLanes is the number of lanes at the intersection.

The average time each class of vehicle takes to cross an intersection can be set according to the location, i.e., region-wise, city-wise, locality-wise, or even intersection-wise based on the characteristics of the intersection, to make traffic management more effective. Data from the respective transport authorities can be analyzed for the same.

The signals switch in a cyclic fashion and not according to the densest direction first. This is in accordance with the current system where the signals turn green one after the other in a fixed pattern and does not need the people to alter their ways or cause any confusion. The order of signals is also the same as the current system **(Red → Green → Yellow → Red)**, and the yellow signals have been accounted for as well.

4.3 Simulation Module

Key Steps in development of simulation:

1. Below Fig 5 and 6 shows the background image considered for intersection.

3-way intersection

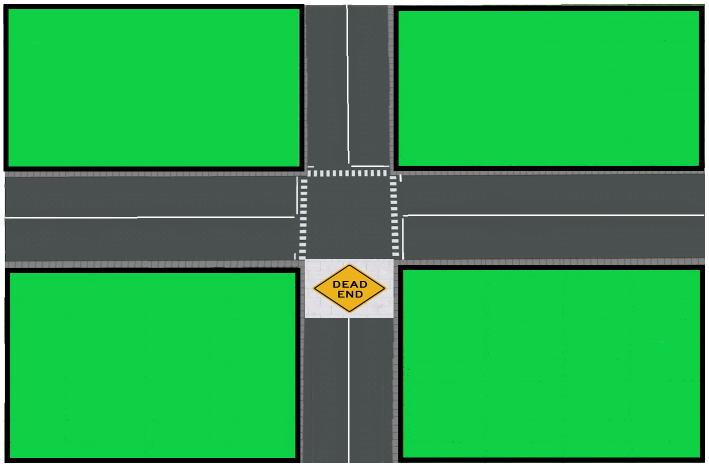

Fig 5: Pygame Simulator Background for 3 Lanes

4-way intersection

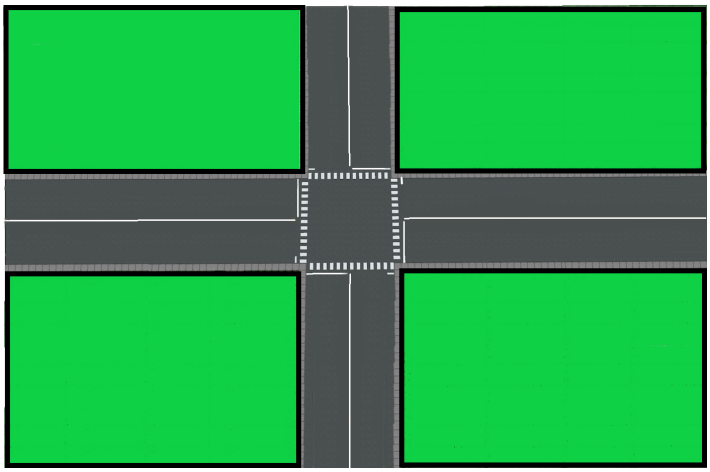

Fig 6: Pygame Simulator Background for 4 Lanes

2. Top-view images of car, truck, bus, rickshaw and motorcycle were shown in Fig 7.
3. Aforementioned images were resized.

Fig 7: Pygame Simulator Vehicles

4. Gathered images of traffic signals.

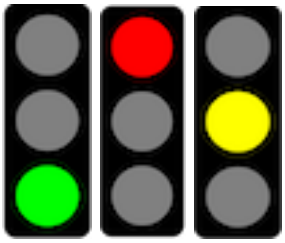

Fig 8: Pygame Simulator Traffic Lights

5. Code developed for rendering the appropriate image of the signal depending on whether it is red, green, or yellow.
6. Worked on displaying the current signal time i.e. the time left for a green signal to turn yellow or a red signal to turn green or a yellow signal to turn red. The green time of the signals is set according to the algorithm, by taking into consideration the number of vehicles at the signal. The red signal times of the other signals are updated accordingly.
7. Generation of vehicles according to direction, lane, vehicle class, and whether it will turn or not all set by random variables. Distribution of vehicles among the 4 directions can be controlled. A new vehicle is generated and added to the simulation after every 1 second.
8. Configured how the vehicles move, each class of vehicle has different speed, there is a gap between 2 vehicles, if a car is following a bus, then its speed is reduced so that it does not crash into the bus.
9. Code developed for how vehicles react to traffic signals i.e. stop for yellow and red, move for green. If they have passed the stop line, then continue to move if the signal turns yellow.
10. Displaying the number of vehicles that have crossed the signal and the total time elapsed since the start of the simulation.
11. Updating the time elapsed as simulation progresses and exiting when the time elapsed equals the desired simulation time, then printing the data that will be used for comparison and data analysis.
12. Worked on vehicle turning and crossing the intersection in the simulation to make it look more realistic.

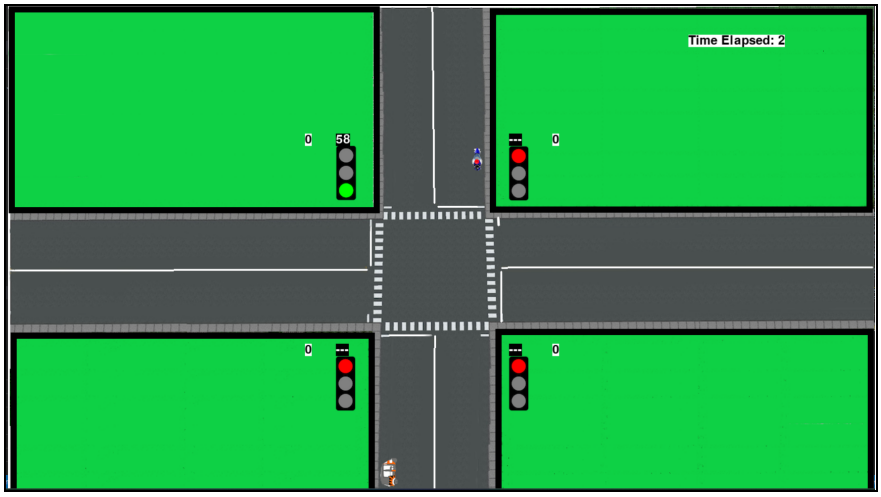

Fig 9: Pygame Simulator Initiation

13. Above Fig 9 shows the simulation just after it starts showing red and green lights, green signal time counting down from a default of 60 and red time of next signal is blank (--). When the signal is red, we display a blank value till it reaches 15 seconds. The number of vehicles that have crossed can be seen beside the signal, which are all 0 initially. The time elapsed since the start of simulation can be seen on top right.

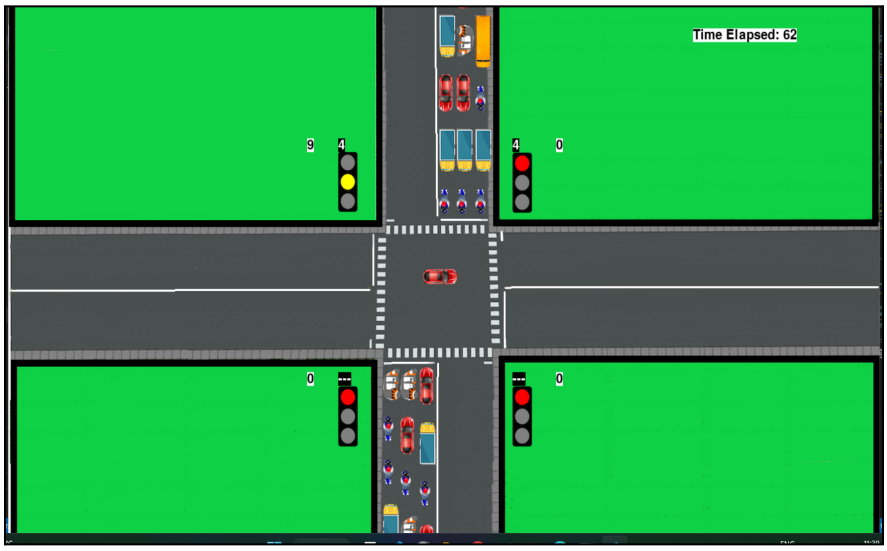

Fig 10: Simulation showing change of lights

14. Above Fig 10 shows the simulation showing yellow light and red time for next signal. When red signal time is less than 15 seconds, we show the countdown timer so that vehicles can start up and be prepared to move once the signal turns green.

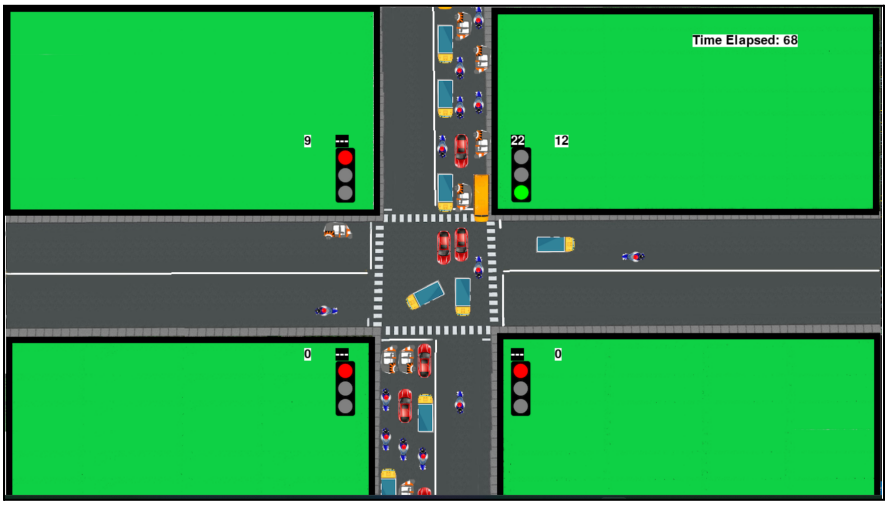

Fig 11: Simulation showing predicted green timer for a light

15. Above Fig 11 shows the instance when green time predicted by YOLO model is less than 60s on the basis of the number and type of vehicles and hence saving time of the commuters which is going to propagate for other arms of the crossroads too.

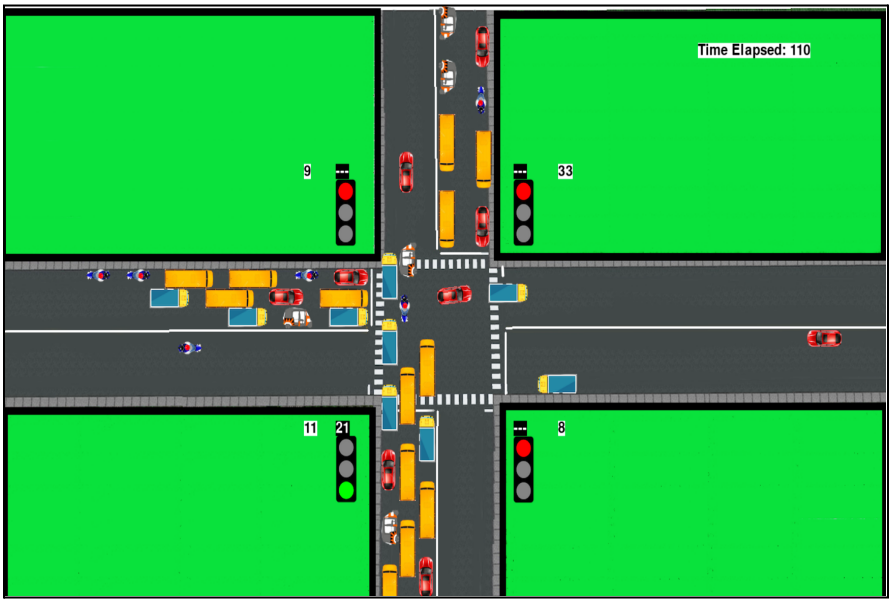

Fig 12: An intermediate state of a crossroad

16. Above fig 12 shows an intermediate state of a 4-Way crossroad and below fig 13 shows the intermediate state of a 3-Way Crossroad

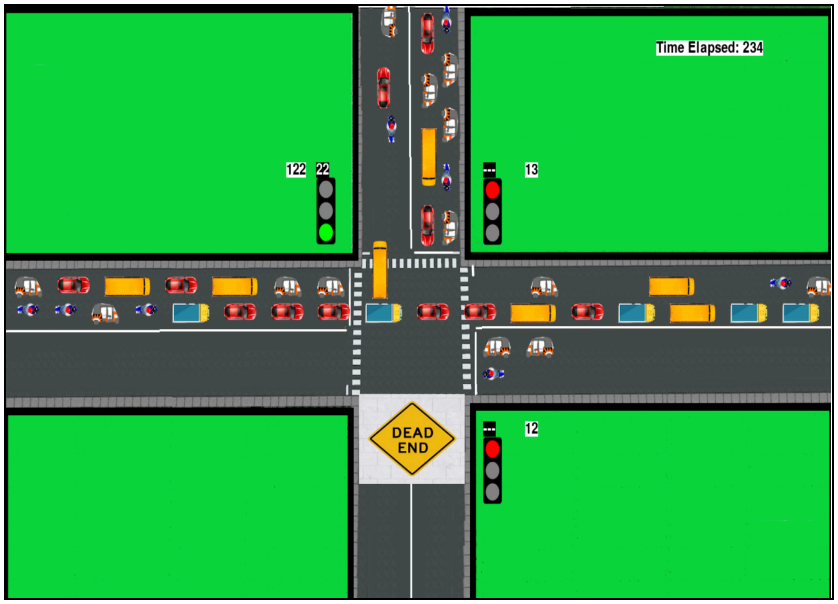

Fig 13: Intermediate Simulation for 3-Way traffic system

4.4 Priority Vehicle Algorithm

The algorithm is designed to respond to Priority Vehicles which is approaching a particular signal for green corridor clearance. The algorithm uses GPS to get the current location of the approaching Priority Vehicle. The algorithm is divided into four cases and the flow is as follows:

Case 1: Priority Vehicle at a Yellow Signal

1. Considering the yellow signal as ith signal, as soon as we receive the request for priority vehicle clearance we set the ith signal to green and set the green time to the Maximum Limit.
2. Then we loop till the green time reaches the Minimum Limit and keep on checking the location of the priority vehicle after every 5 sec and find out if the priority vehicle has cleared the signal or not.
3. Once the priority vehicle has cleared the signal we exhaust the rest of the Minimum Limit green time for the ith signal and set the red time of the (i+1)th, (i+2)th and (i+3)th signals accordingly.

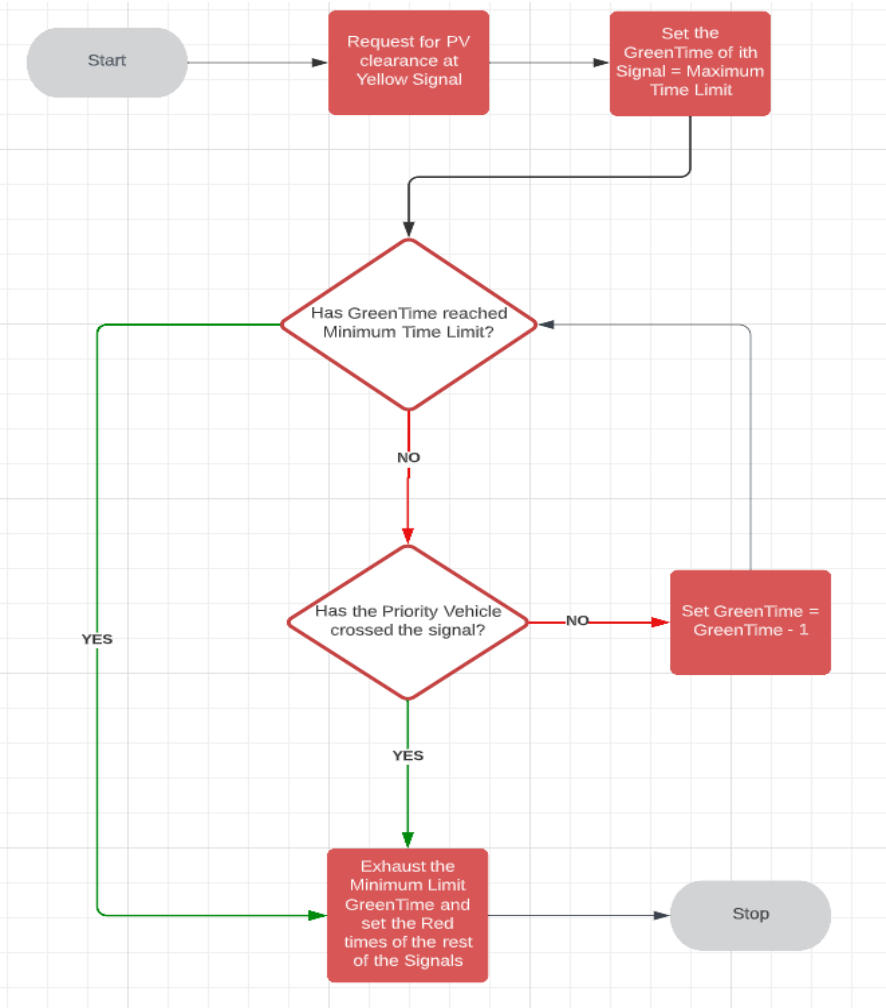

Fig 14: Flowchart showing the algorithmic flow for Priority Vehicle at a Yellow Signal

Case 2: Priority Vehicle at a Green Signal

1. The green signal is the ith signal, so as soon as we receive the request for priority vehicle clearance we set the ith signal timer to the Maximum limit.
2. It then repeats the logic of the yellow signal from step 2.

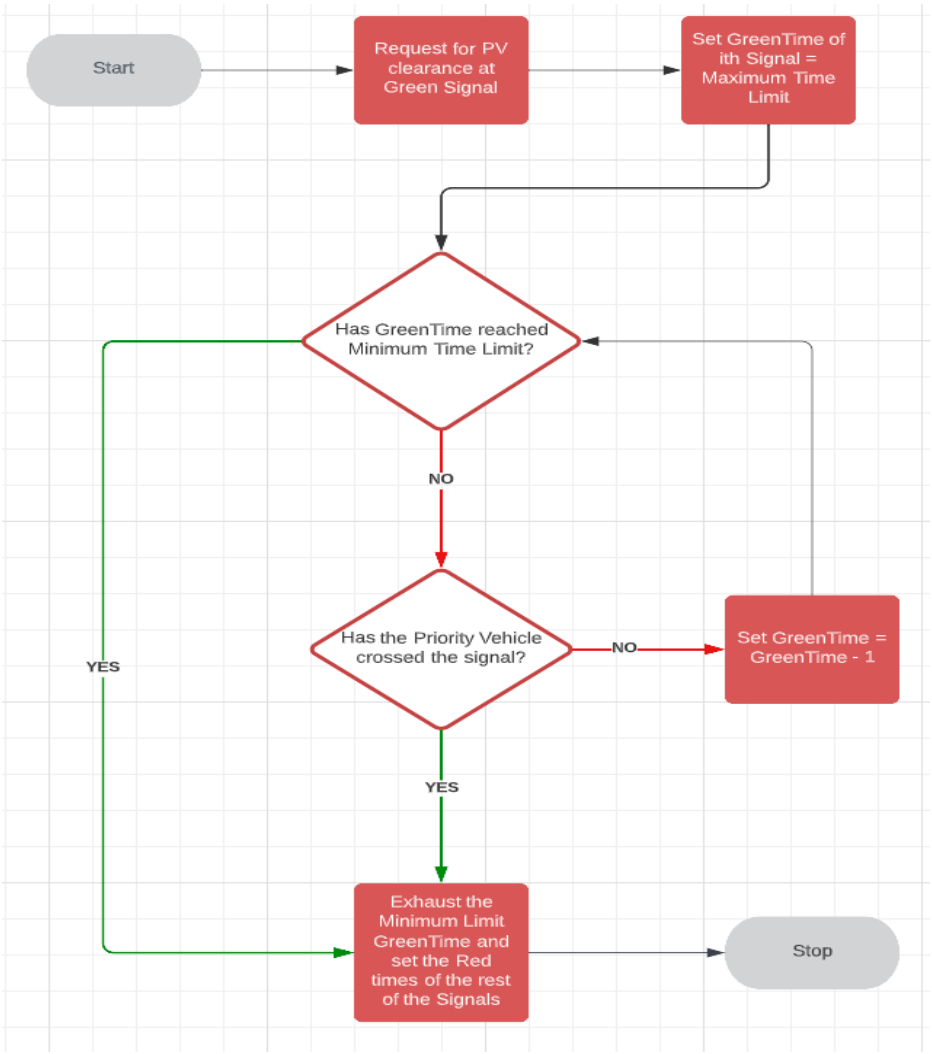

Fig 15: Flowchart showing the algorithmic flow for Priority Vehicle at a Green Signal

Case 3: Priority Vehicle at Next Green Signal

1. Considering the green signal as ith signal, as soon as we receive the request for priority vehicle clearance we set the green signal timer of the ith signal to Minimum Time limit, if it is greater than the Minimum Time Limit. We set the red timer of the (i+1)th or next green signal as the sum of the green and the yellow timer of the ith signal.
2. Then we loop to exhaust the green and yellow timers of the ith signal and red timer of the (i+1)th signal where the priority vehicle is present.
3. Then we set the default values for the ith signal and change the current green to (i+1)th signal and the next green to (i+2)th signals respectively.
4. It then repeats the logic of the yellow signal from step 2 for the current green signal.

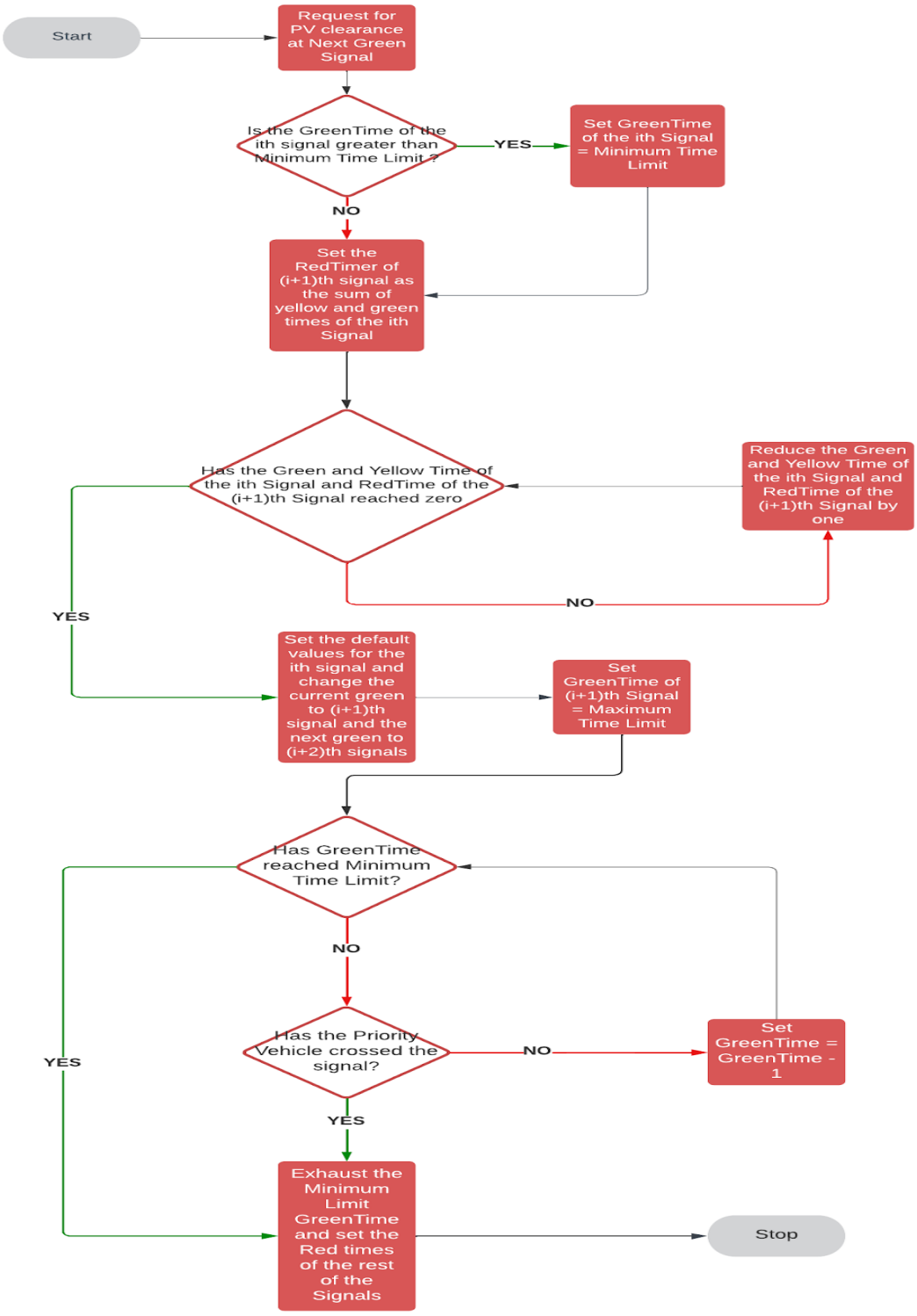

Fig 16: Flowchart showing the algorithmic flow for Priority Vehicle at Next Green Signal

Case 4: Priority Vehicle at a Red Signal

1. Considering the green signal as ith signal, as soon as we receive the request for priority vehicle clearance we set the green signal timer of the ith signal to Minimum Time limit, if it is greater than the Minimum Time Limit. Here we define the signal as a Priority Signal(PS) which can be any of (i+1), (i+2) or (i+3). We set the red timer of the PS as the sum of the green and the yellow timer of the ith signal.
2. Then we loop to exhaust the green and yellow timers of the ith signal and red timer of the PS where the priority vehicle is coming.
3. Then we set the default values for the ith signal and change the current green to PS and set the green timer of the PS as the Maximum Time Limit.
4. Then we loop till the green time of the PS reaches the Minimum Time Limit and keep on checking the location of the priority vehicle after every 5 sec and find out if the priority vehicle has cleared the signal or not.
5. Once the priority vehicle has cleared the signal we exhaust the rest of the Minimum Limit green time and the yellow time for PS and set the red time of the (i+1)th, (i+2)th and (i+3)th signals accordingly.
6. Then we set the (i+1)th signal as the current green signal and sets the green time of the (i+1)th signal using YOLO prediction and set the (i+2)th signal as the next green signal. It sets the red time of the next green signal as:
   RedTime for (i+2)th signal = greenTime for (i+1)th signal + yellowTime for (i+1)th signal.

4.5 Trust Score Module

The algorithm is designed to display the trust score by utilizing the real time traffic congestion information from Google Maps API, real time weather from OpenWeather API and hotspot data taken from the Chandigarh administration.

Flow of the algorithm is as follows:

1. Data Integration: Setting up a data pipeline to fetch real-time congestion data from Google Maps API and using OpenWeather API to retrieve real-time weather data for the specified locations. Cleaning and transforming the weather data to obtain key weather parameters like rainfall, snowfall, visibility, thunderstorm etc.
2. Historical Accidents Analysis: Analyzing the accident data to identify accident-prone areas, hotspots, and patterns associated with each lane.
3. Trust Score Calculation: Calculating a trust score value that takes into account congestion, weather, and historical accident data and applying a decay function to set the range of trust score from 0 to 1, while giving equal weightage to each factor.
4. Real-time Updates: Setting up a mechanism to continuously fetch and update the congestion and weather data.

5. Mobile App Integration: Integrating the trust score functionality into the existing mobile app module. Designing and implementing user interfaces to display the trust scores for each lane. Enabling users to access and utilize the trust scores when selecting a route or lane.

Weather	Trust Score
Tornado	0
Thunderstorm	0
Sand	0.1
Dust	0.1
Fog	0.2
Smoke	0.2
Drizzle	0.3
Haze	0.3
Snow	0.3
Overcast clouds	0.3
Rain	0.4
Mist	0.4
Broken clouds	0.5
Scattered Clouds	0.7
Few Clouds	0.8
Clear Sky	1

Google API Congestion Data in minutes	Trust Score
0	1
1	0.983
2	0.967
3	0.95
4	0.933
5	0.917
6	0.9
7	0.883
8	0.867
9	0.85
10	0.833

Fig 17: Mapping of trust score outputs for different Weather and Congestion Inputs

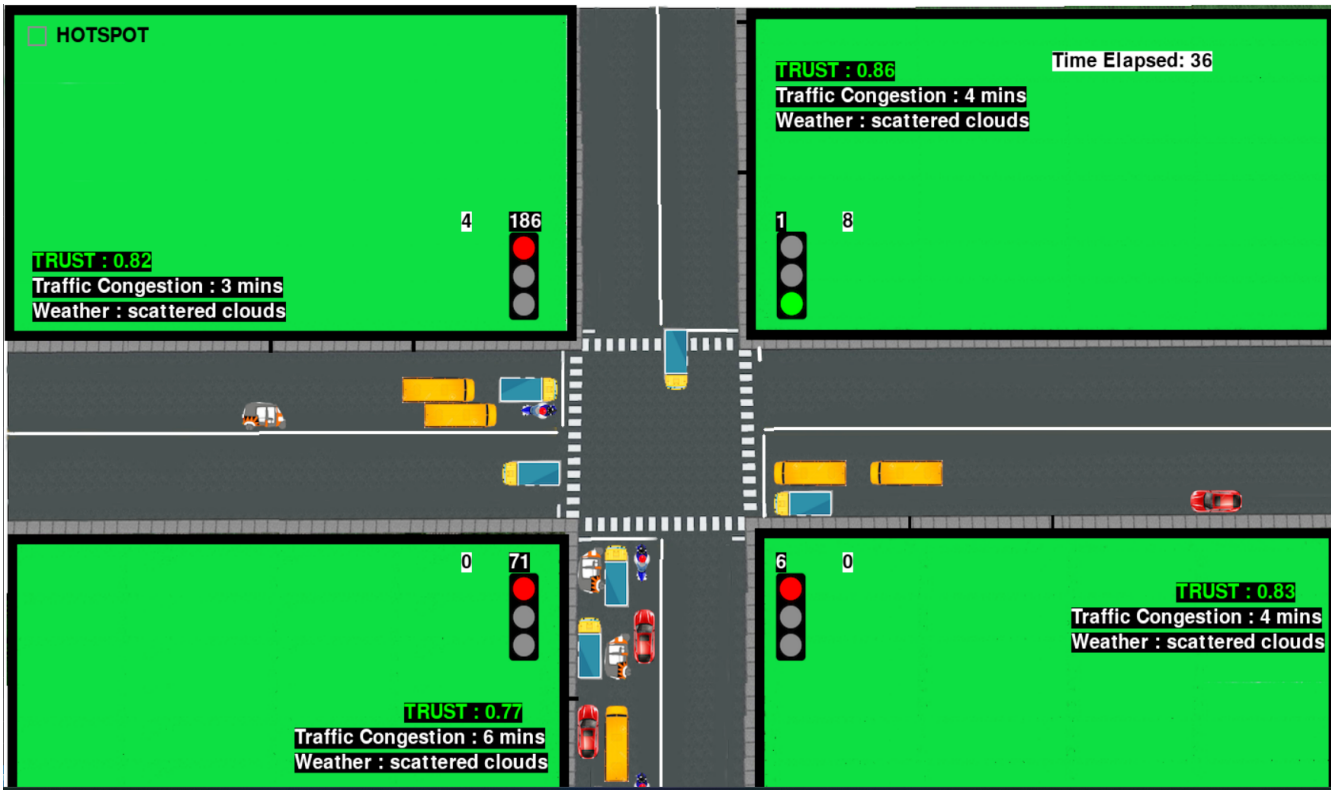

Fig 18: Trust, Traffic Congestion and Weather data shown in real time

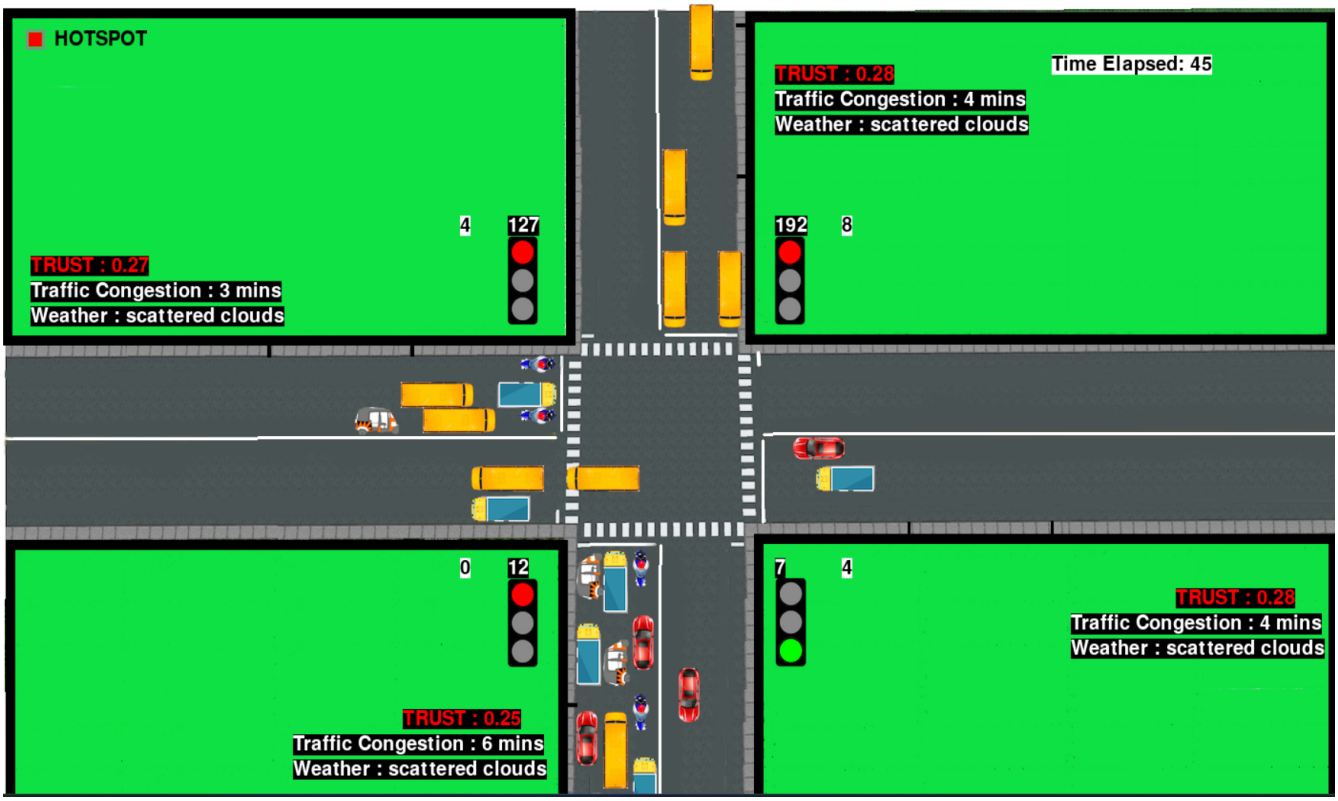

Fig 19: Hotspot region with Trust, Traffic Congestion and Weather data in real time

We have conducted a comprehensive mapping of all the junctions in Chandigarh and established an AWS EC2 instance to gather weather and congestion data for every lane connected to each junction. This data is collected at three-hour intervals over a period of 10 days. The purpose of this data collection is to utilize machine learning models in predicting trust scores for the lanes in future scenarios. These trust scores can aid in better traffic management and decision-making processes.

4.6 Hardware Module

The hardware implementation involves the setup of LED lights connected to a breadboard. For each lane, there are three lights: red, green, and yellow. To regulate the current flowing through the LEDs, 220 ohm resistors are connected between the positive and negative terminals of the breadboard. The GPIO pins of a Raspberry Pi 4 are then connected to the breadboard using male-female wires, establishing the necessary electrical connections. Whenever the traffic lights change in the Pygame simulator, the corresponding LED lights on the breadboard reflect these changes, providing a visual representation of the traffic signal status for each lane.

In addition, a camera module is attached to the Raspberry Pi 4 to capture real-time images for YOLO (You Only Look Once), an object detection algorithm. YOLO enables the detection and identification of various objects within the camera's field of view. Furthermore, the Intelligent Traffic Control System (ITCS), Priority algorithm and Trust Score are deployed on the Raspberry Pi 4. These algorithms facilitate intelligent decision-making processes based on the collected data, allowing for efficient traffic management and optimization.

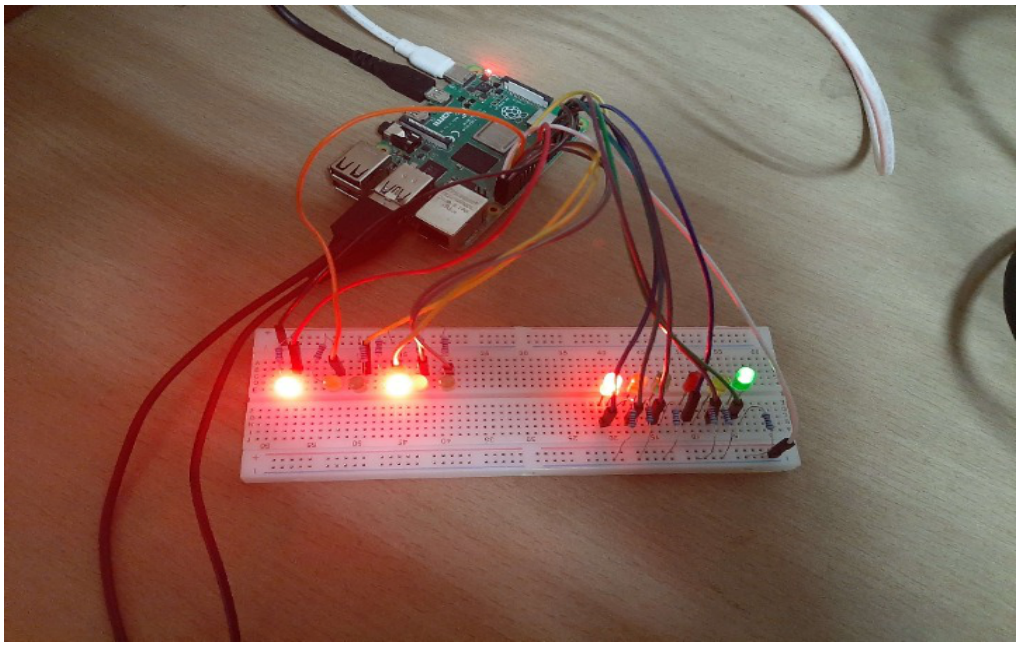

Fig 20: Hardware Implementation

4.7 Mobile Application Module

Mobile Application Module deals with the implementation of the Emergency and Trust Score for the mobile application. It includes features such as mapping the priority vehicle to the nearest junction and calculating the trust score for that junction. It also includes user authentication and profile for the app users.

The Emergency component handles fetching the current location and rendering a map view with the current location and nearest junction. Bearing angle formula is used to find the junctions with the least angle with the user's heading direction and calculations of the distances is done between the user's location and the least bearing junction to find the nearest junction.

When the user hits the request for the green corridor then it updates the flag variable in the Firebase Realtime Database and a priority vehicle appears on the Pygame simulation.

The Trust Score component is similar to the Emergency component but focuses on calculating and displaying the trust scores for the connected junctions. Users can even click on the map to find the nearest junction from the selected pin on the map and see the trust scores for the respective connected junctions.

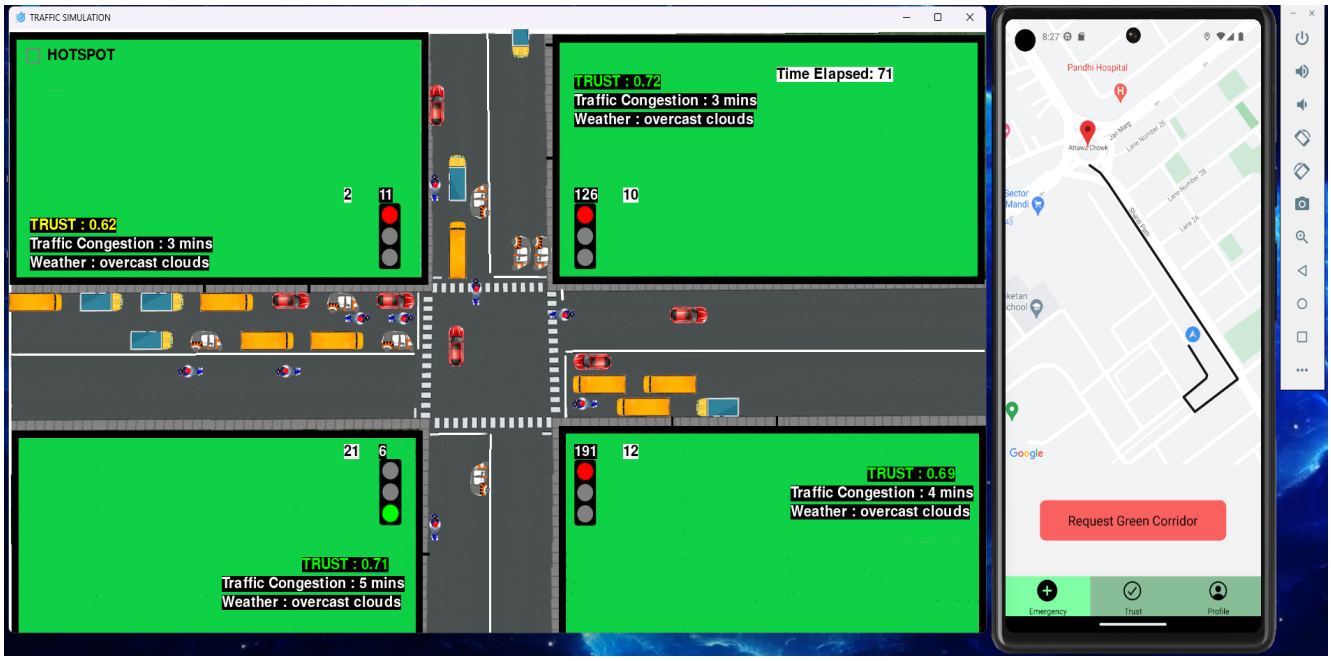

Fig 21: Emergency Screen with Pygame Simulation

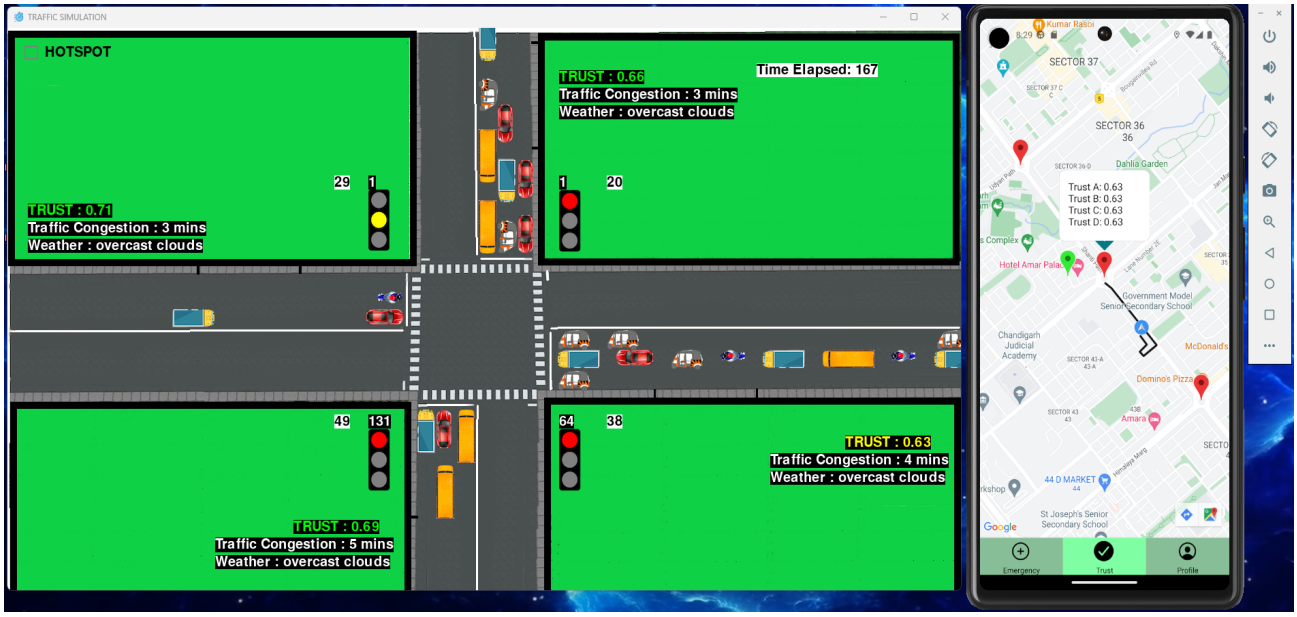

Fig 22: Trust Score Screen displaying the trust scores for the adjoining junctions

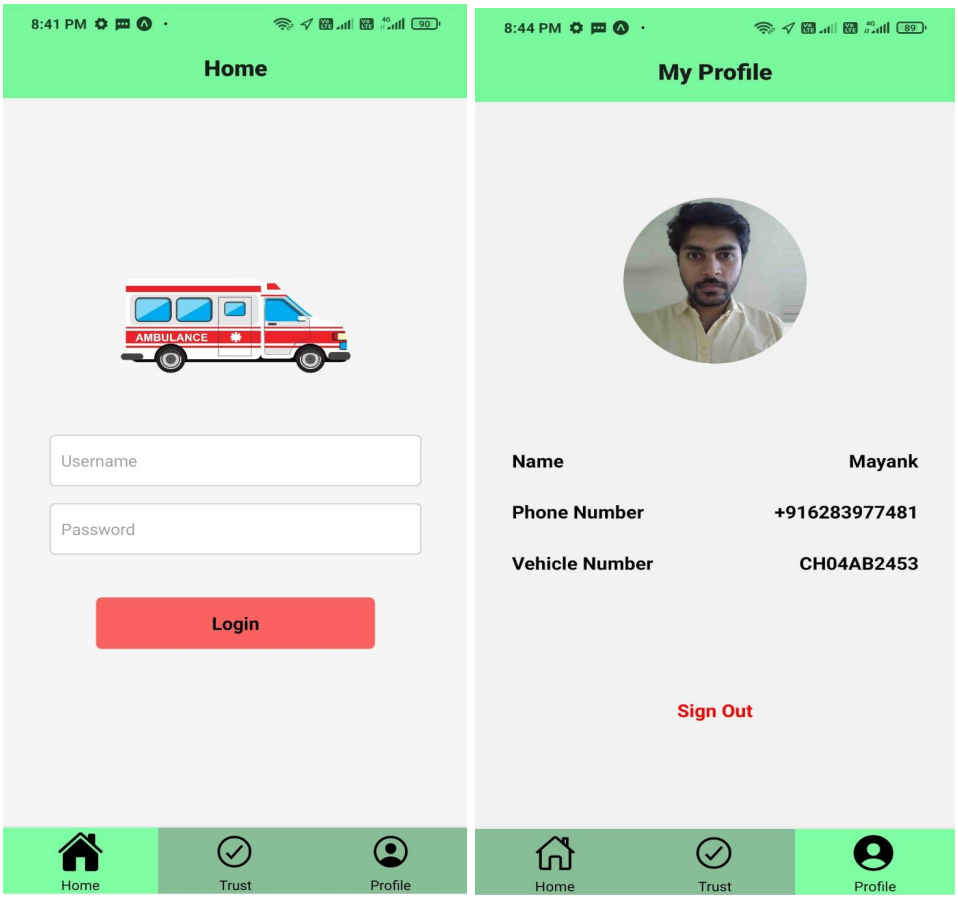

Fig 23: Home and Profile Screen for the user of the Mobile App

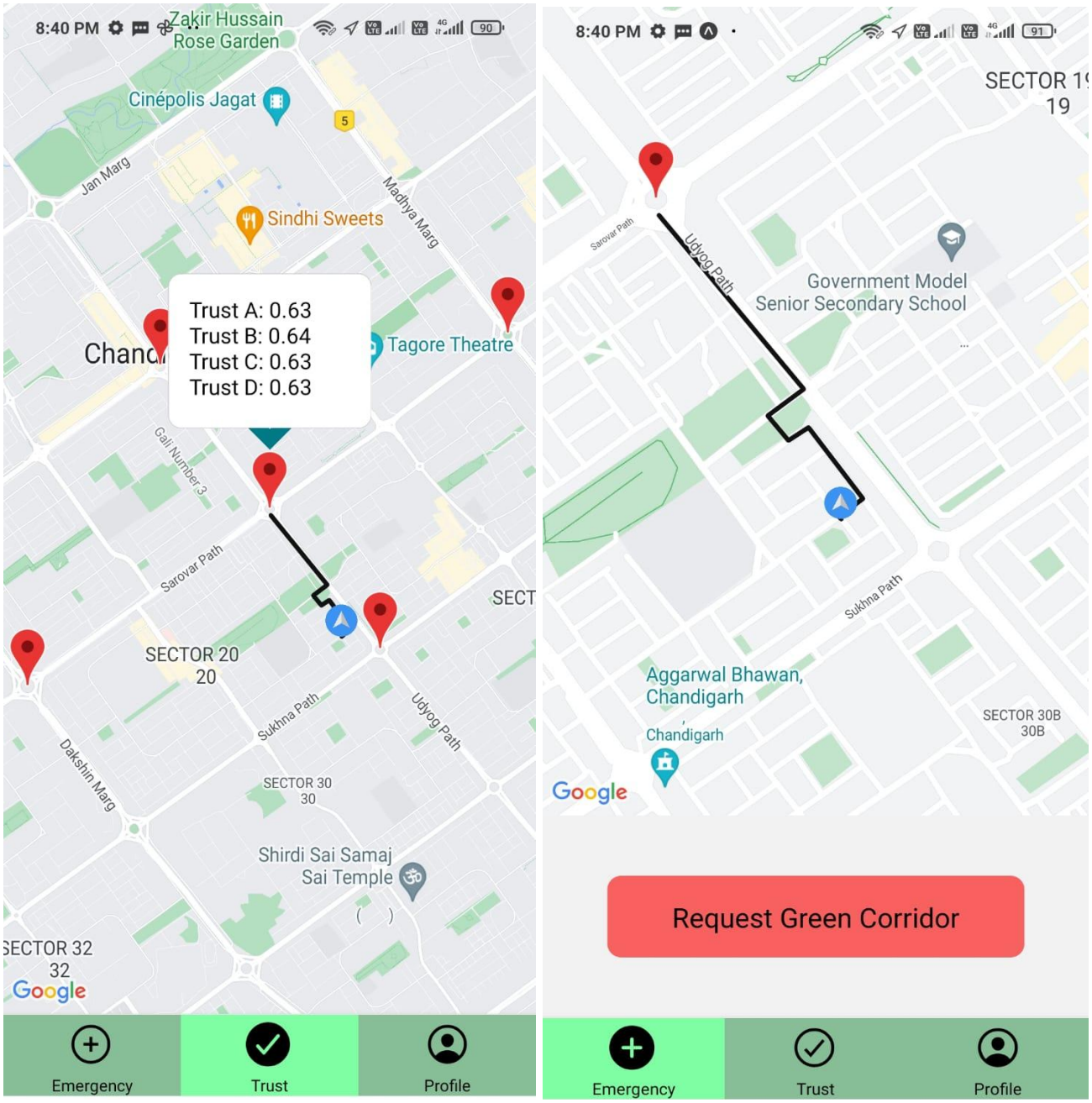

Fig 24: Trust Score and Emergency Screens of the App

4.8 Technologies and Tools used

4.8.1 React Native

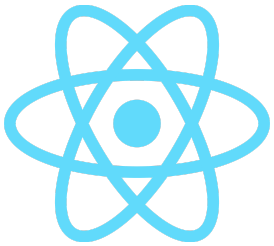

Fig 25: React Native Logo

React Native is a powerful cross-platform framework that enables developers to create mobile applications using JavaScript, providing the ability to write code once and deploy it on both Android and iOS platforms. With React Native, developers can leverage native components and APIs, resulting in high-performance applications that closely resemble their native counterparts. The framework's "learn once, write anywhere" approach allows for faster development cycles and reduced effort in building applications for multiple platforms.

Additionally, React Native boasts a vibrant ecosystem of third-party libraries and a strong community support, offering developers a wide range of tools and resources to enhance their productivity. Its hot-reloading feature further streamlines the development process, enabling instant visualization of changes and facilitating efficient iteration.

4.8.2 Javascript

Fig 26: Java Script Logo

JavaScript is a lightweight, interpreted programming language. It is designed for creating network-centric applications. It is complementary to and integrated with Java. JavaScript is very easy to implement because it is integrated with HTML. It is open and cross-platform.

4.8.3 Firebase

Fig 27: Firebase Logo

Firebase is a powerful backend platform that provides a range of services to support mobile and web application development. It offers a comprehensive suite of tools and features such as real-time database, authentication, cloud storage, and hosting. With Firebase, developers can easily store and sync data in real-time, authenticate users, and implement user-friendly features like push notifications. Its scalability and reliability make it an ideal choice for applications requiring real-time updates and collaborative functionality. Additionally, Firebase offers seamless integration with other Google Cloud services, providing developers with a robust and flexible solution for building high-quality and scalable applications.

4.8.4 Python

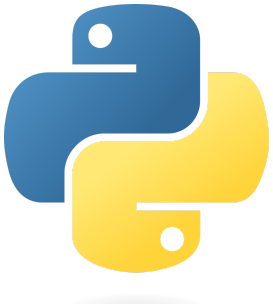

Fig 28: Python Logo

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

4.8.5 Pygame

Fig 29: Pygame Logo

Pygame is a cross-platform set of Python modules which is used to create video games. It consists of computer graphics and sound libraries designed to be used with the Python programming language. Pygame was officially written by Pete Shinners to replace PySDL. Pygame is suitable to create client-side applications that can be potentially wrapped in a standalone executable.

4.8.6 Amazon Web Services

Fig 30: AWS Logo

Amazon Web Services (AWS) is a comprehensive and widely adopted cloud computing platform that offers a vast array of services and solutions for businesses and developers. With AWS, organizations can leverage scalable and flexible infrastructure resources, including virtual servers, storage, databases, and networking capabilities, to build and deploy their applications and services. AWS provides a global network of data centers, ensuring high availability and low latency for users worldwide.

Additionally, AWS offers a wide range of managed services, such as machine learning, analytics, and serverless computing, empowering developers to focus on their core business logic without worrying about underlying infrastructure management. The platform's pay-as-you-go pricing model enables cost optimization by only paying for the resources used, making AWS a scalable and cost-effective solution for businesses of all sizes. Overall, AWS provides a robust and reliable cloud computing platform that enables organizations to innovate, scale, and transform their digital infrastructure with ease.

4.8.7 Raspberry Pi 4

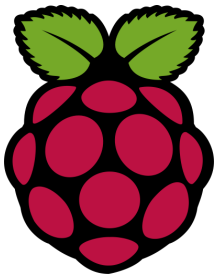

Fig 31: Raspberry Pi Logo

Raspberry Pi 4 is a versatile and affordable single-board computer that has gained significant popularity among developers, hobbyists, and educators. Powered by a quad-core ARM Cortex-A72 processor and with options for different RAM configurations, Raspberry Pi 4 offers improved performance compared to its predecessors. It provides a range of I/O ports, including USB, HDMI, Ethernet, GPIO, and wireless connectivity options, allowing for seamless integration with various peripherals and expansion boards.

Raspberry Pi 4 is widely used for diverse applications, ranging from home automation and robotics to media centers and IoT projects. Its small form factor and low power consumption make it an ideal choice for embedded systems and portable devices. The Pi 4 supports multiple operating systems, including Linux distributions like Raspbian, making it accessible for developers with different backgrounds and skill levels.

4.8.8 Visual Studio Code

Fig 32: Visual Studio Code Logo

Visual Studio Code is a source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git.

4.8.9 Jupyter Notebook

Fig 33: Jupyter Notebook Logo

The Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, statistical modeling, data visualization, machine learning, and much more. We used Jupyter Notebook for writing code, compiling it and getting the results in modern fashion.

CHAPTER 5: RESULTS AND DISCUSSION

The proposed system sets the green signal time adaptively according to the traffic density at the signal and ensures that the direction with more traffic is allotted a green signal for a longer duration of time as compared to the direction with lesser traffic which also took **heterogeneity** of vehicles into consideration. This will lower the unwanted delays and reduce congestion and waiting time, which in turn will reduce fuel consumption and pollution.

According to simulation results, the system shows about **24.52%** improvement for 4 lane roads and **13.61%** for 3 lane roads over the current system in terms of the number of vehicles crossing the intersection in a given time frame referenced in Fig 34 and Fig 35 respectively, which is a significant improvement. With further calibration using real time CCTV data for training the model, this system can be improved to perform even better.

The system shows about **44.75%** improvement when the priority vehicle arrives every 30 seconds and about **47.08%** improvement for 60 seconds over the current ITCS system, so it can significantly reduce the time taken for emergency vehicles to reach their destination, potentially saving many lives.

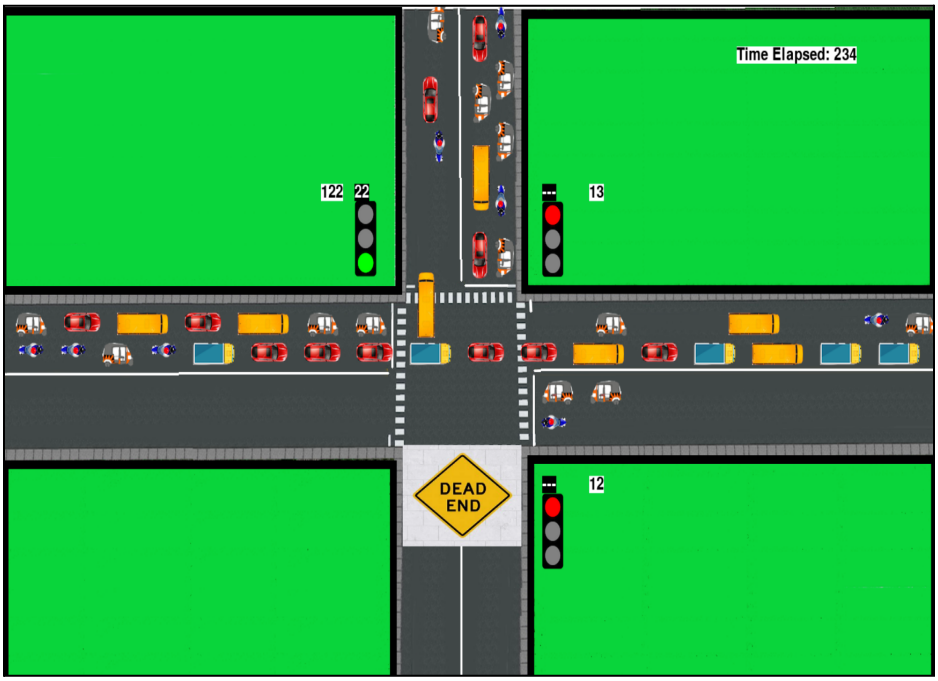

Fig 34: 3-Way System Simulation

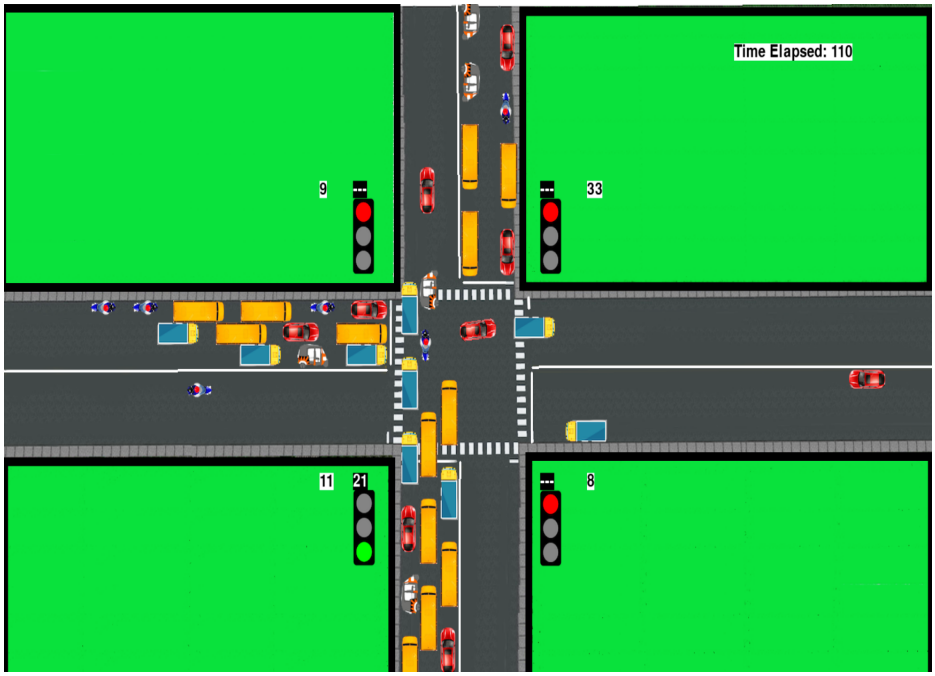

Fig 35: 4-way System Simulation

Fig 36: YOLO Model Detection Case 1

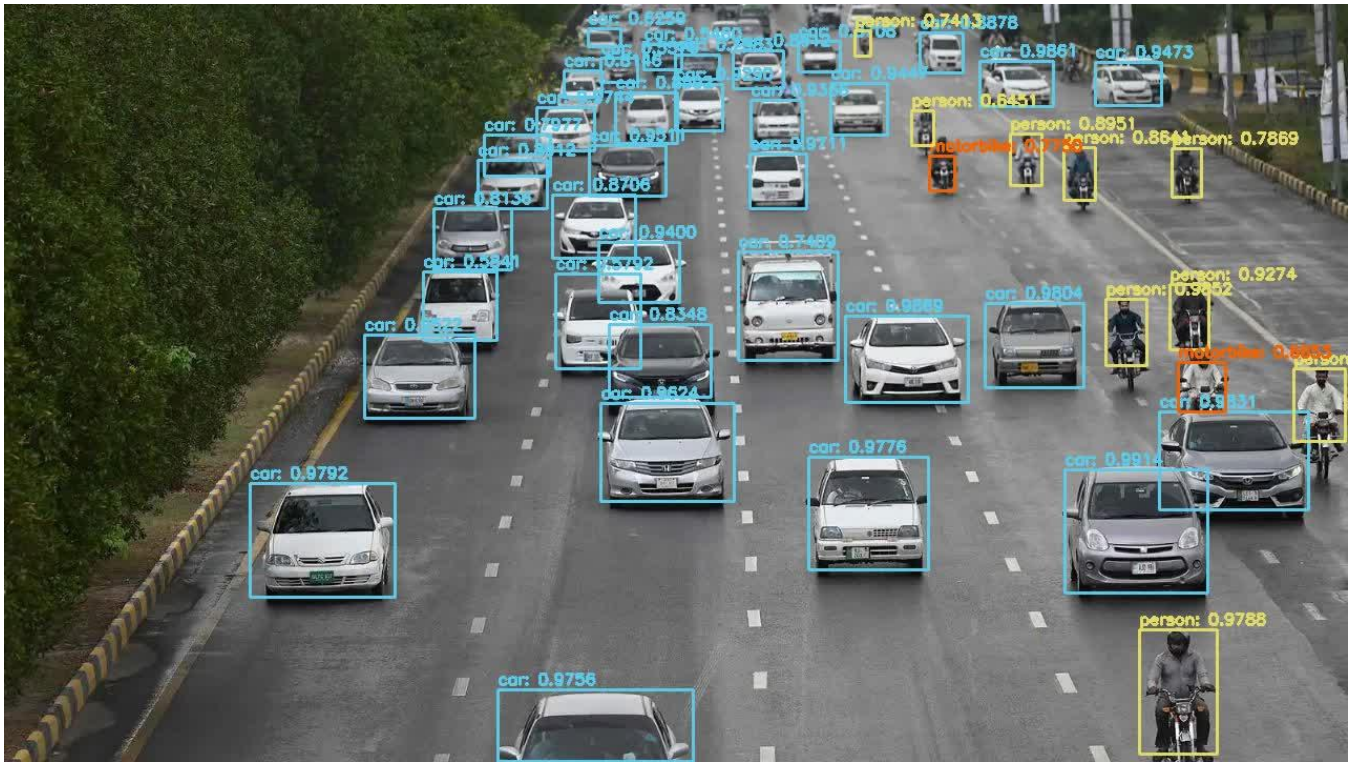

Fig 37: YOLO Model Detection Case 2

The below figures contain the following variables as columns:

- P(i) indicates the traffic density, where the sum of all the P(i) is equal to one.
- Lane(i) indicates the number of vehicles crossed from that lane.
- Total(S) indicates the total number of vehicles crossed from all the lanes considering the static system.
- Total(D) indicates the total number of vehicles crossed from all the lanes considering the dynamic system.
- Improvement indicates the percentage improvement considering the dynamic system over the static system. The red circled regions indicate that in these cases the improvement has been less than 10%.
- WT indicates the total waiting time of the priority vehicles and PV indicates the number of priority vehicles.

	P1	P2	P3	Lane1	Lane2	Lane3	Total(D)	P1	P2	P3	Lane1	Lane2	Lane3	Total(S)	Improvement(%Age)
1	0.000	0.000	1.000	0	0	257	257	0.000	0.000	1.000	0	0	133	133	93.233083
2	0.000	0.500	0.500	0	113	113	226	0.000	0.500	0.500	0	126	76	202	11.881188
3	0.000	0.600	0.400	0	119	97	216	0.000	0.600	0.400	0	134	66	200	8.000000
4	0.333	0.333	0.334	74	93	61	228	0.333	0.333	0.334	68	90	55	213	7.042254
5	0.300	0.500	0.200	65	126	42	233	0.300	0.500	0.200	74	126	28	228	2.192982
6	0.500	0.400	0.100	100	113	29	242	0.500	0.400	0.100	100	109	14	223	8.520179
7	0.500	0.300	0.200	107	80	40	227	0.500	0.300	0.200	108	81	32	221	2.714932
8	0.700	0.200	0.100	185	38	29	252	0.700	0.200	0.100	142	51	13	206	22.330097
9	0.500	0.450	0.050	123	111	11	245	0.500	0.450	0.050	109	115	10	234	4.700855
10	0.300	0.600	0.100	73	161	18	252	0.300	0.600	0.100	66	144	16	226	11.504425
11	0.200	0.550	0.250	53	109	59	221	0.200	0.550	0.250	48	131	43	222	-0.450450
12	0.350	0.150	0.500	80	51	89	220	0.350	0.150	0.500	79	37	83	199	10.552764
13	0.350	0.500	0.150	85	134	33	252	0.350	0.500	0.150	79	123	25	227	11.013216
14	0.940	0.040	0.020	200	13	5	218	0.940	0.040	0.020	168	13	7	188	15.957447
15	0.850	0.100	0.050	193	18	17	228	0.850	0.100	0.050	194	31	15	240	-5.000000

Fig 38: Data points for 3-way System

	P1	P2	P3	P4	Lane1	Lane2	Lane3	Lane4	Total(D)	P1	P2	P3	P4	Lane1	Lane2	Lane3	Lane4	Total(S)	Improvement(%Age)
1	0.00	0.00	0.00	1.00	0	0	0	269	269	0.00	0.00	0.00	1.00	0	0	0	188	188	43.085106
2	0.00	0.00	0.50	0.50	0	0	146	158	304	0.00	0.00	0.50	0.50	0	0	109	117	226	34.513274
3	0.00	0.30	0.30	0.40	0	102	80	120	302	0.00	0.30	0.30	0.40	0	120	69	81	270	11.851852
4	0.25	0.25	0.25	0.25	94	86	74	82	336	0.25	0.25	0.25	0.25	72	92	58	60	282	19.148936
5	0.30	0.30	0.20	0.20	101	114	57	66	338	0.30	0.30	0.20	0.20	88	99	45	55	287	17.770035
6	0.50	0.20	0.20	0.10	177	77	76	27	357	0.50	0.20	0.20	0.10	139	75	53	24	291	22.680412
7	0.30	0.20	0.30	0.20	94	76	111	65	346	0.30	0.20	0.30	0.20	76	75	84	52	287	20.557491
8	0.70	0.10	0.10	0.10	229	32	37	32	330	0.70	0.10	0.10	0.10	180	46	36	17	279	18.279570
9	0.50	0.40	0.05	0.05	148	130	26	17	321	0.50	0.40	0.05	0.05	133	131	20	16	300	7.000000
10	0.30	0.30	0.30	0.10	99	99	90	38	326	0.30	0.30	0.30	0.10	87	110	71	32	300	8.666667
11	0.20	0.50	0.05	0.25	69	150	17	92	328	0.20	0.50	0.05	0.25	63	157	20	61	301	8.970100
12	0.35	0.15	0.35	0.15	124	49	107	53	333	0.35	0.15	0.35	0.15	93	57	59	38	247	34.817814
13	0.35	0.35	0.15	0.15	122	134	49	39	344	0.35	0.35	0.15	0.15	91	129	39	31	290	18.620690
14	0.94	0.02	0.02	0.02	274	4	14	6	298	0.94	0.02	0.02	0.02	165	8	8	7	188	58.510638
15	0.85	0.05	0.05	0.05	231	21	27	22	301	0.85	0.05	0.05	0.05	163	18	21	8	210	43.333333

Fig 39: Data points for 4-way System

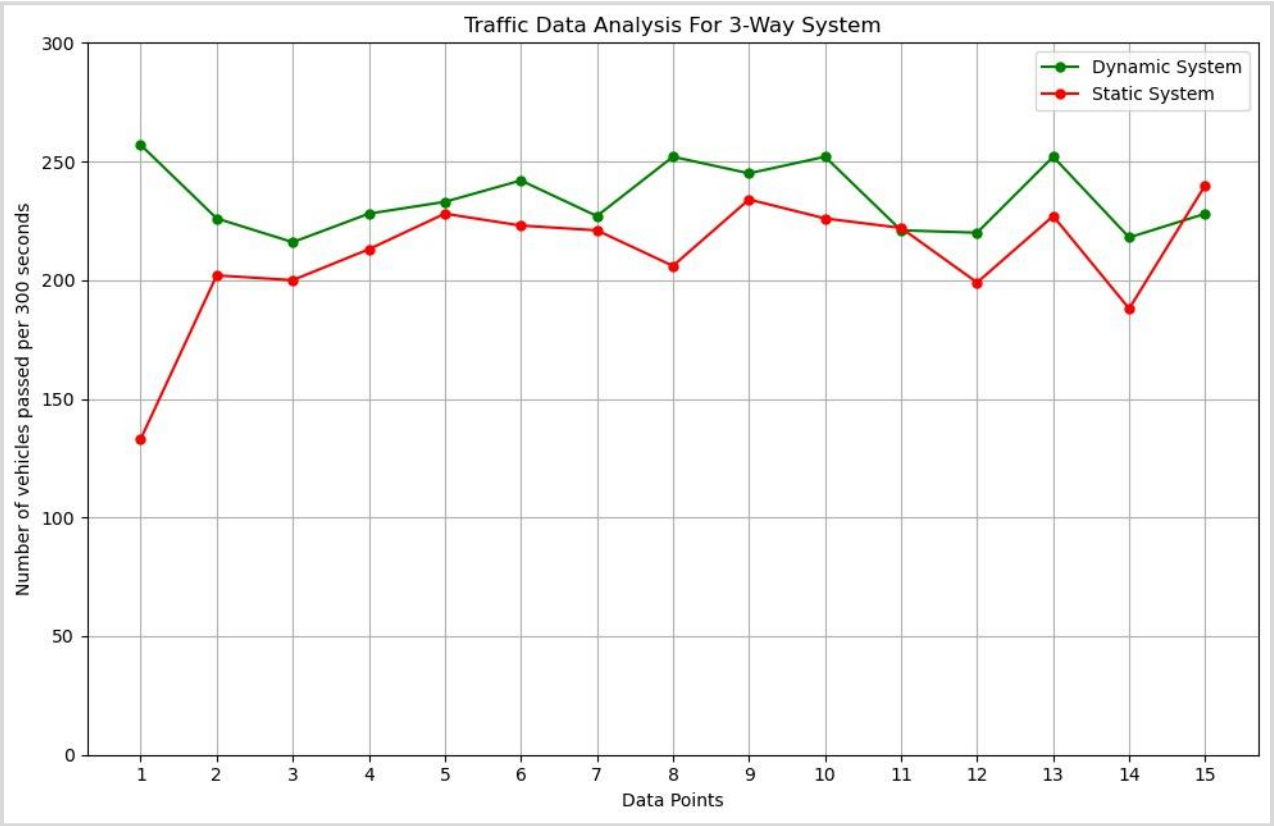

Fig 40: Analysis of Traffic for 15 data points in 3-Way System(Static v/s Dynamic)

Plot in Fig 40 shows the traffic data analysis for 3-Way System for both dynamic and static systems. The x-axis indicates the test case number and the y-axis indicates the number of vehicles passed per 300 seconds. There is a small amount of gap between the static and the dynamic case indicating that the dynamic and the static system give near to the same results for 3-Way System.

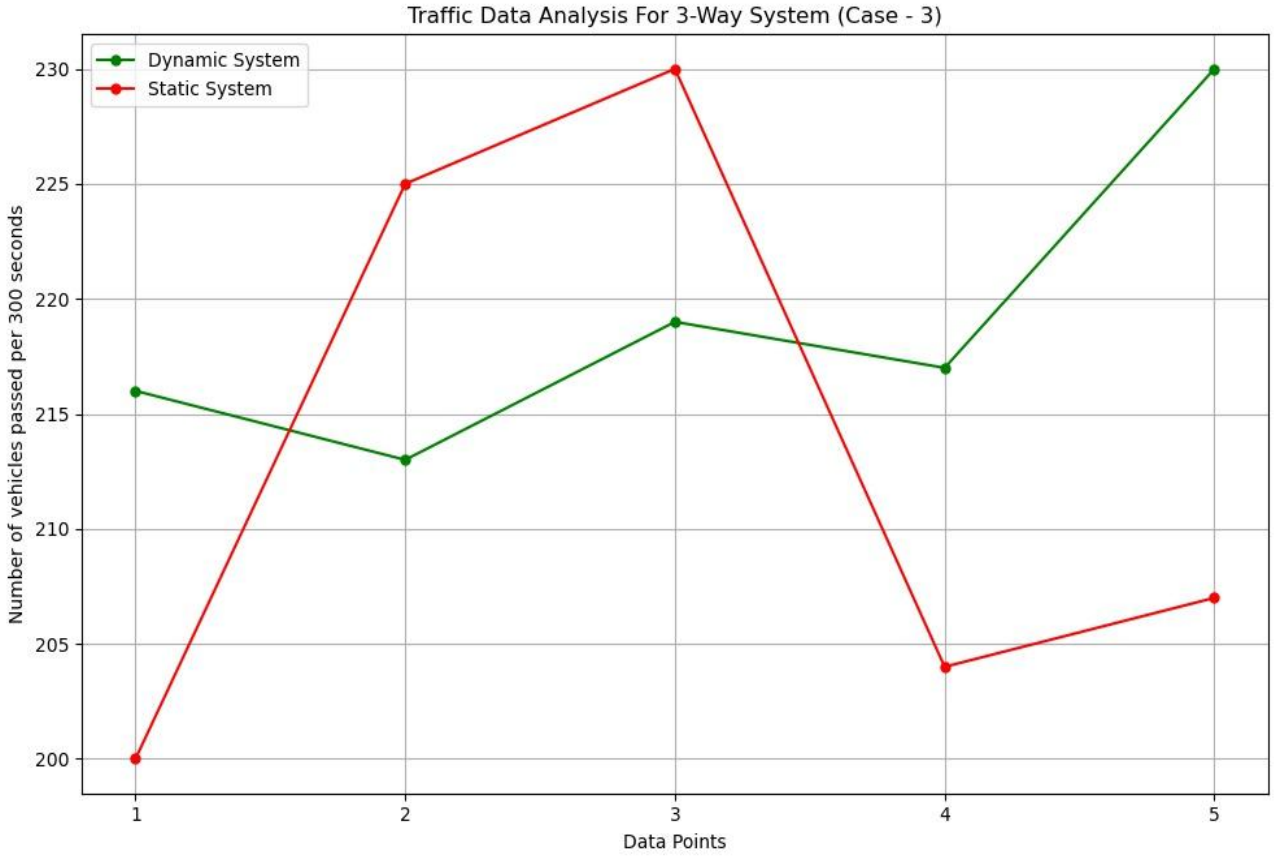

Fig 41: Analysis of Traffic for Case no. 3 in 3-Way System

Plot in Fig 41 shows the Traffic data analysis for 3-Way System for test case number 3 for both dynamic and static systems. Dynamic system gives almost uniform results and the static system is having abrupt changes.

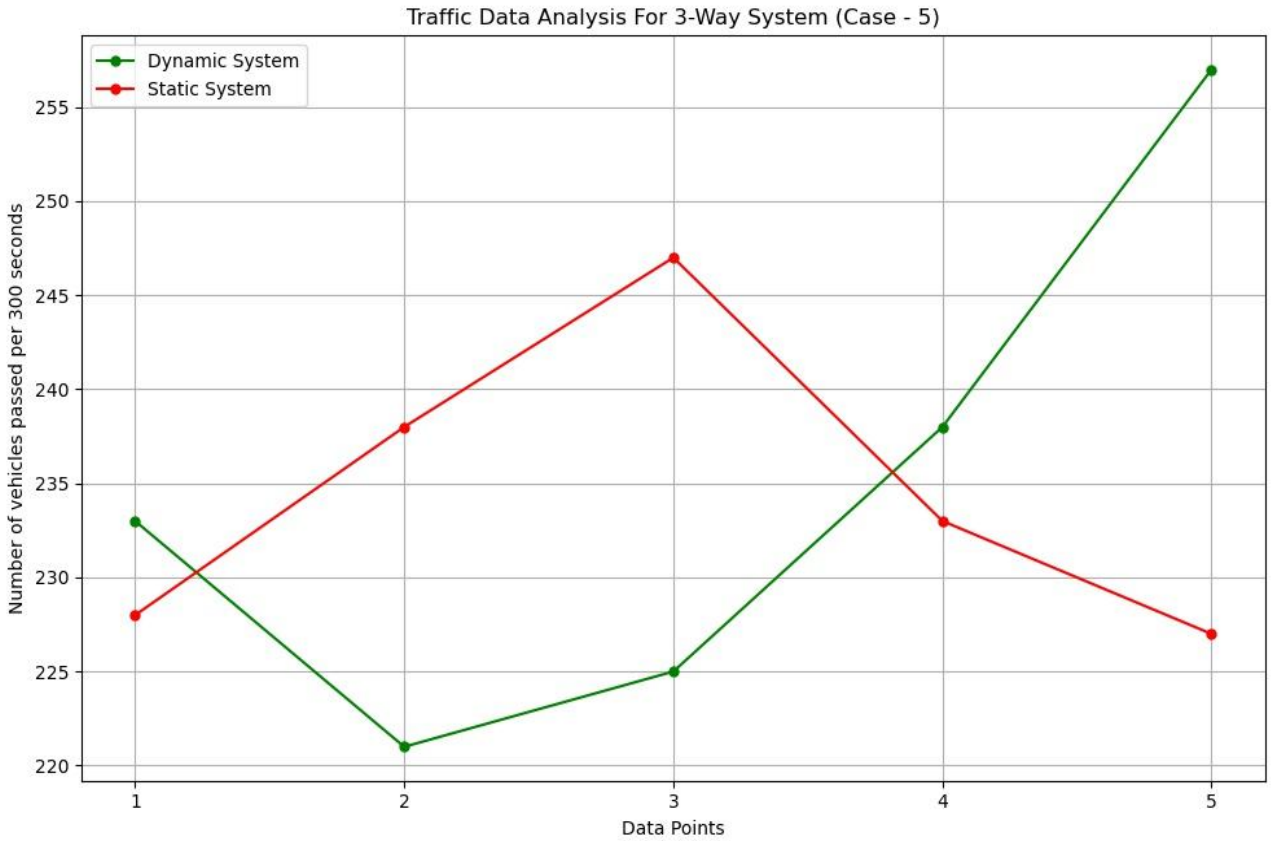

Fig 42: Analysis of Traffic for Case no. 5 in 3-Way System

Plot in Fig 42 shows the Traffic data analysis for 3-Way System for test case number 5 for both dynamic and static systems. Dynamic system first decreases and then increases at a faster rate and Static system first increases and then decreases.

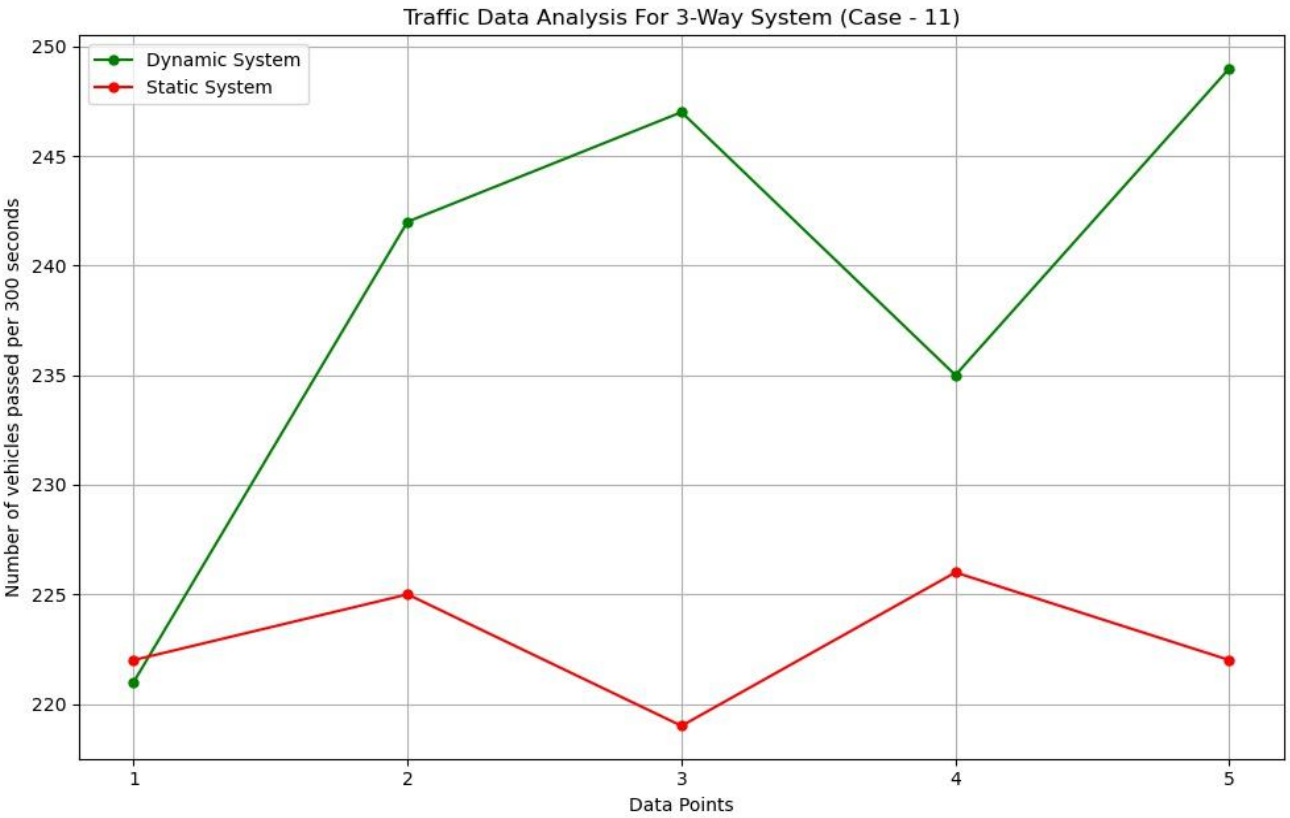

Fig 43: Analysis of Traffic for Case no. 11 in 3-Way System

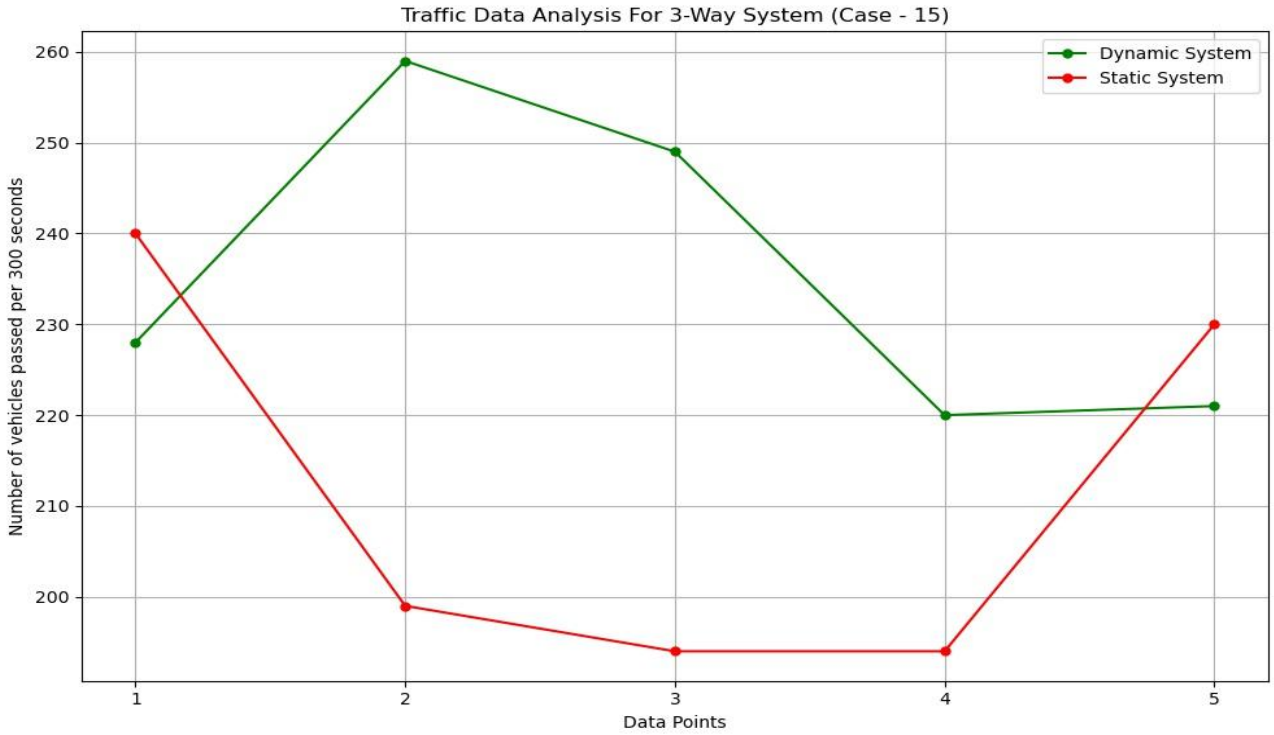

Fig 44: Analysis of Traffic for Case no. 15 in 3-Way System

Plot in Fig 43 shows the Traffic data analysis for 3-Way System for test case number 11 for both dynamic and static systems. Dynamic and static systems are having a large gap between them with dynamic systems showing good results.

Plot in Fig 44 shows the Traffic data analysis for 3-Way System for test case number 15 for both dynamic and static systems. Both the systems are having a large gap between them with the dynamic system having a good response time for the vehicles to pass.

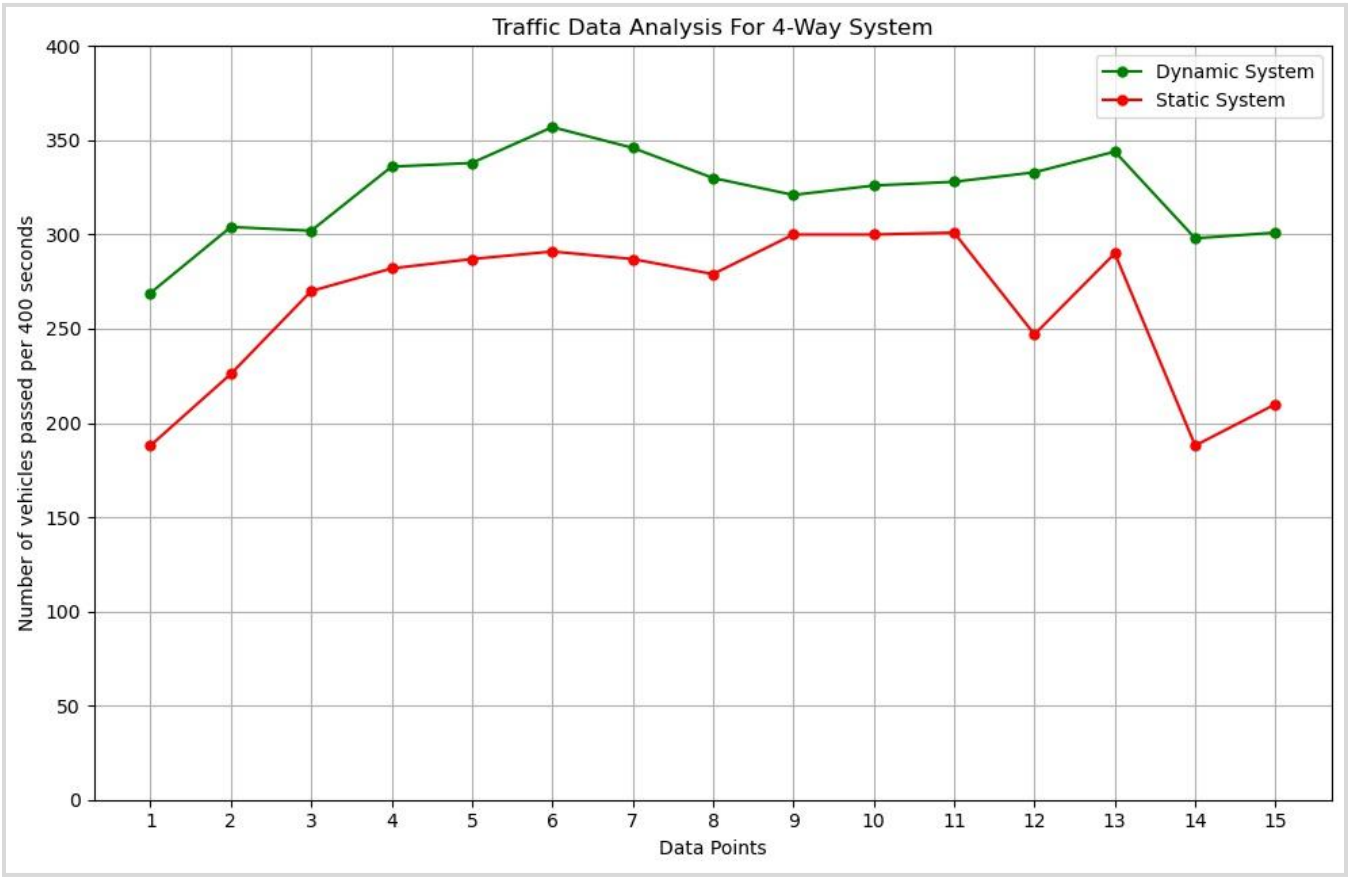

Fig 45: Analysis of Traffic for 15 data points in 4-Way (Static v/s Dynamic)

Plot in Fig 45 shows the traffic data analysis for 4-Way System for both dynamic and static systems. The x-axis indicates the test case number and the y-axis indicates the number of vehicles passed per 400 seconds. There is a significant amount of gap between the static and the dynamic case indicating that the dynamic cases respond well to the random input traffic density.

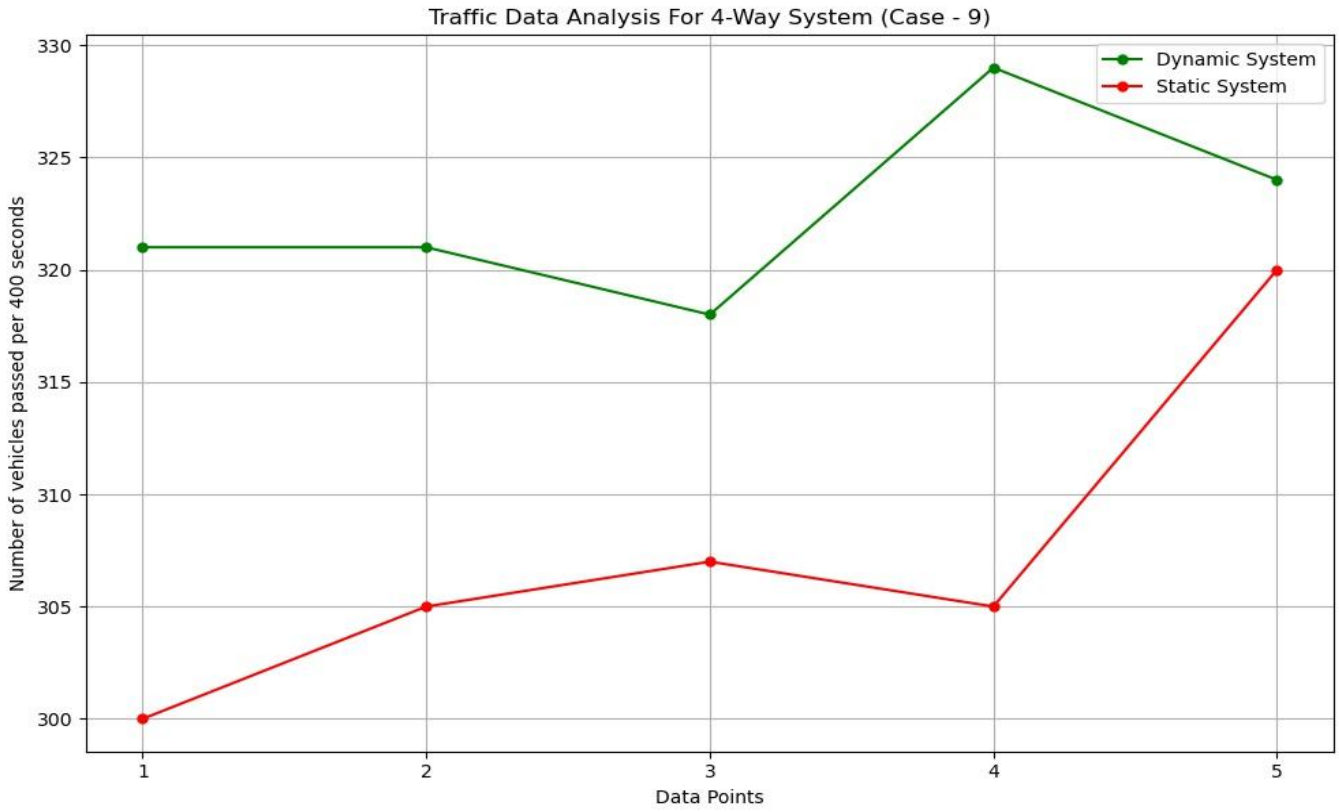

Fig 46: Analysis of Traffic for Case no. 9 in 4-Way System

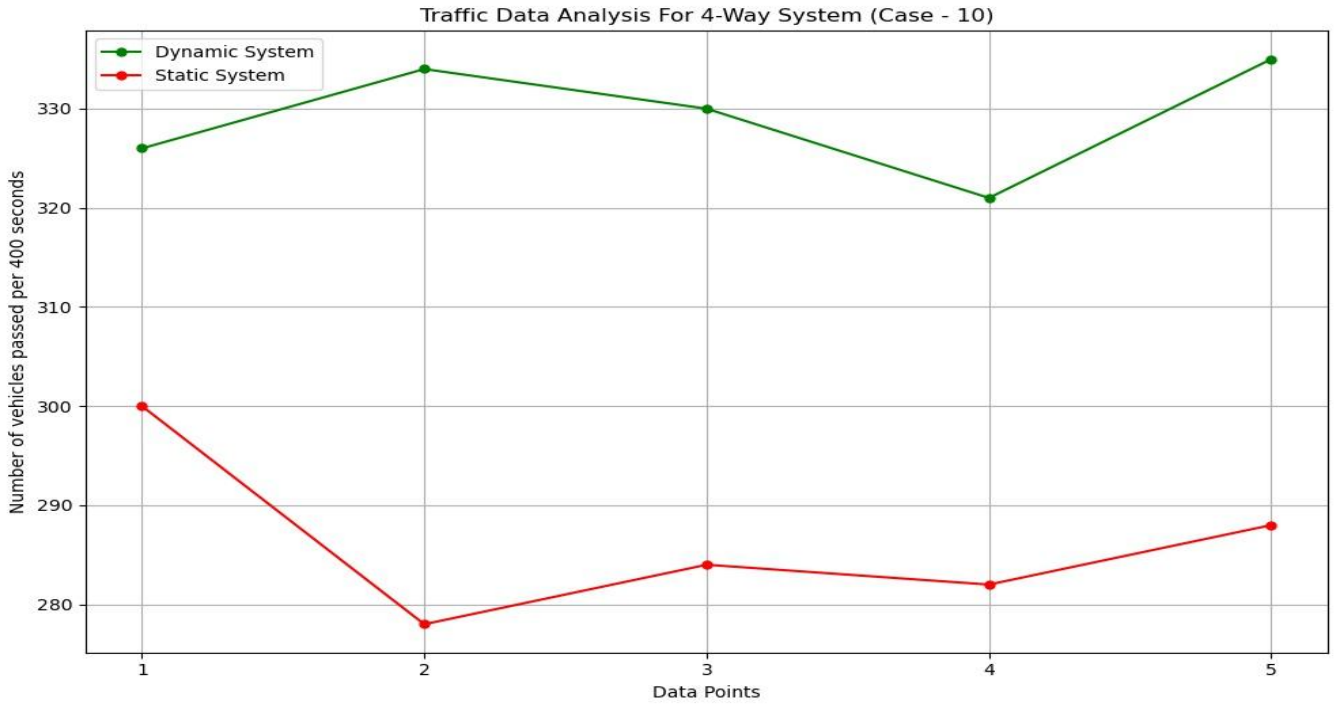

Fig 47: Analysis of Traffic for Case no. 10 in 4-Way System

Plot in Fig 46 shows the Traffic data analysis for 4-Way System for test case number 9 for both dynamic and static systems. Dynamic system shows good results with a large gap between both the systems.

Plot in Fig 47 shows the Traffic data analysis for 4-Way System for test case number 10 for both dynamic and static systems. Dynamic system shows good results with a large gap between both the systems.

Fig 48: Analysis of Traffic for Case no. 11 in 4-way System

Plot in Fig 48 shows the Traffic data analysis for 4-Way System for test case number 11 for both dynamic and static systems. Dynamic system shows good results with a large gap between both the systems.

Through this project, we have made a system that has the potential to give more accurate green time dynamically by analyzing the number and type of vehicles using the YOLO model thus helping commuters to save their time and saving nature by the reduction of fuel consumption and air pollution.

	P1	P2	P3	P4	WT	PV	Total	WT per PV	P1	P2	P3	P4	WT	PV	Total	WT per PV	Improvement(%Age)
1	0.00	0.00	0.00	1.00	153	12	382	12.750000	0.00	0.00	0.00	1.00	273	13	387	21.000000	39.285714
2	0.00	0.00	0.50	0.50	151	13	384	11.615385	0.00	0.00	0.50	0.50	252	11	352	22.909091	49.297924
3	0.00	0.30	0.30	0.40	167	12	370	13.916667	0.00	0.30	0.30	0.40	227	12	366	18.916667	26.431718
4	0.25	0.25	0.25	0.25	126	13	365	9.692308	0.25	0.25	0.25	0.25	287	12	371	23.916667	59.474672
5	0.30	0.30	0.20	0.20	146	13	359	11.230769	0.30	0.30	0.20	0.20	189	11	366	17.181818	34.635735
6	0.50	0.20	0.20	0.10	124	12	351	10.333333	0.50	0.20	0.20	0.10	262	11	370	23.818182	56.615776
7	0.30	0.20	0.30	0.20	150	13	364	11.538462	0.30	0.20	0.30	0.20	255	12	365	21.250000	45.701357
8	0.70	0.10	0.10	0.10	156	12	370	13.000000	0.70	0.10	0.10	0.10	290	12	373	24.166667	46.206897
9	0.50	0.40	0.05	0.05	158	12	375	13.166667	0.50	0.40	0.05	0.05	285	12	364	23.750000	44.561404
10	0.30	0.30	0.30	0.10	144	12	355	12.000000	0.30	0.30	0.30	0.10	259	11	372	23.545455	49.034749
11	0.20	0.50	0.05	0.25	149	12	365	12.416667	0.20	0.50	0.05	0.25	261	13	370	20.076923	38.154534
12	0.35	0.15	0.35	0.15	155	12	353	12.916667	0.35	0.15	0.35	0.15	269	12	358	22.416667	42.379182
13	0.35	0.35	0.15	0.15	113	12	365	9.416667	0.35	0.35	0.15	0.15	273	11	373	24.818182	62.057387
14	0.94	0.02	0.02	0.02	169	12	382	14.083333	0.94	0.02	0.02	0.02	292	12	349	24.333333	42.123288
15	0.85	0.05	0.05	0.05	175	12	374	14.583333	0.85	0.05	0.05	0.05	293	13	382	22.538462	35.295791

Fig 49: Data points for comparison between ITCS+Priority System and ITCS Algorithm (T = 30 sec)

	P1	P2	P3	P4	WT	PV	Total	WT per PV	P1	P2	P3	P4	WT	PV	Total	WT per PV	Improvement(%Age)
1	0.00	0.00	0.00	1.00	76	5	380	15.200000	0.00	0.00	0.00	1.00	133	6	380	22.166667	31.428571
2	0.00	0.00	0.50	0.50	79	6	390	13.166667	0.00	0.00	0.50	0.50	126	4	340	31.500000	58.201058
3	0.00	0.30	0.30	0.40	63	5	380	12.600000	0.00	0.30	0.30	0.40	114	5	360	22.800000	44.736842
4	0.25	0.25	0.25	0.25	71	6	360	11.833333	0.25	0.25	0.25	0.25	144	5	375	28.800000	58.912037
5	0.30	0.30	0.20	0.20	71	6	355	11.833333	0.30	0.30	0.20	0.20	95	4	356	23.750000	50.175439
6	0.50	0.20	0.20	0.10	70	5	350	14.000000	0.50	0.20	0.20	0.10	131	4	375	32.750000	57.251908
7	0.30	0.20	0.30	0.20	73	6	360	12.166667	0.30	0.20	0.30	0.20	128	5	370	25.600000	52.473958
8	0.70	0.10	0.10	0.10	79	5	372	15.800000	0.70	0.10	0.10	0.10	145	5	370	29.000000	45.517241
9	0.50	0.40	0.05	0.05	85	5	377	17.000000	0.50	0.40	0.05	0.05	146	5	360	29.200000	41.780822
10	0.30	0.30	0.30	0.10	74	5	358	14.800000	0.30	0.30	0.30	0.10	130	4	366	32.500000	54.461538
11	0.20	0.50	0.05	0.25	69	5	362	13.800000	0.20	0.50	0.05	0.25	148	6	380	24.666667	44.054054
12	0.35	0.15	0.35	0.15	75	5	350	15.000000	0.35	0.15	0.35	0.15	135	5	350	27.000000	44.444444
13	0.35	0.35	0.15	0.15	68	5	369	13.600000	0.35	0.35	0.15	0.15	117	4	375	29.250000	53.504274
14	0.94	0.02	0.02	0.02	86	5	380	17.200000	0.94	0.02	0.02	0.02	146	5	350	29.200000	41.095890
15	0.85	0.05	0.05	0.05	88	5	370	17.600000	0.85	0.05	0.05	0.05	147	6	372	24.500000	28.163265

Fig 50: Data points for comparison between ITCS+Priority System and ITCS Algorithm (T = 60 sec)

The above tables in Fig 49 and 50 show the data points comparison between the ITCS+Priority System and ITCS Algorithm based upon waiting times for priority vehicles appearing on the simulation after every 30 and 60 sec respectively.

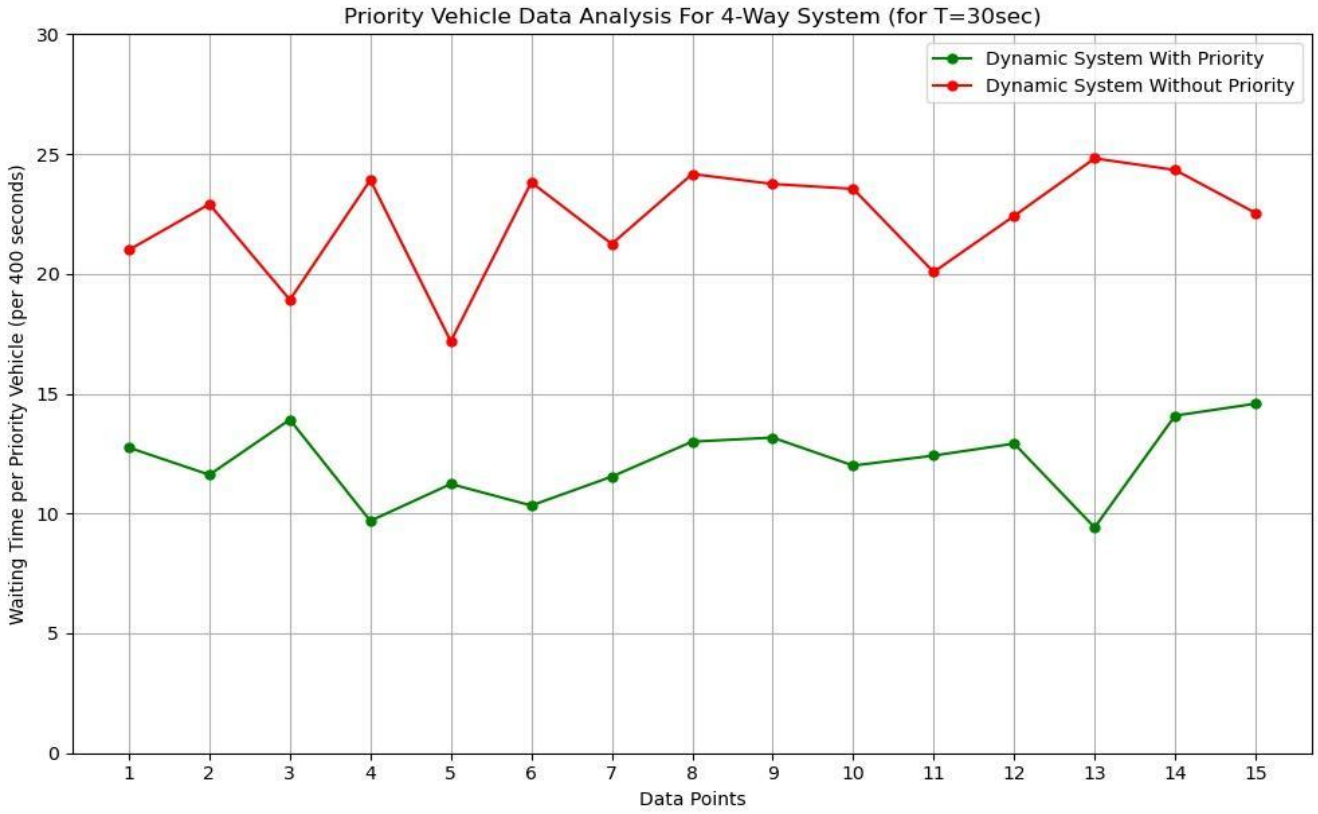

Fig 51: Analysis of Priority Vehicle Waiting Times for T = 30 sec

Plot in Fig 51 shows the analysis of Priority Vehicle waiting times when the priority vehicle arrives after every 30 sec for 4-Way junction system for ITCS+Priority System and ITCS System. The ITCS+Priority System shows good results with a large gap between both the systems.

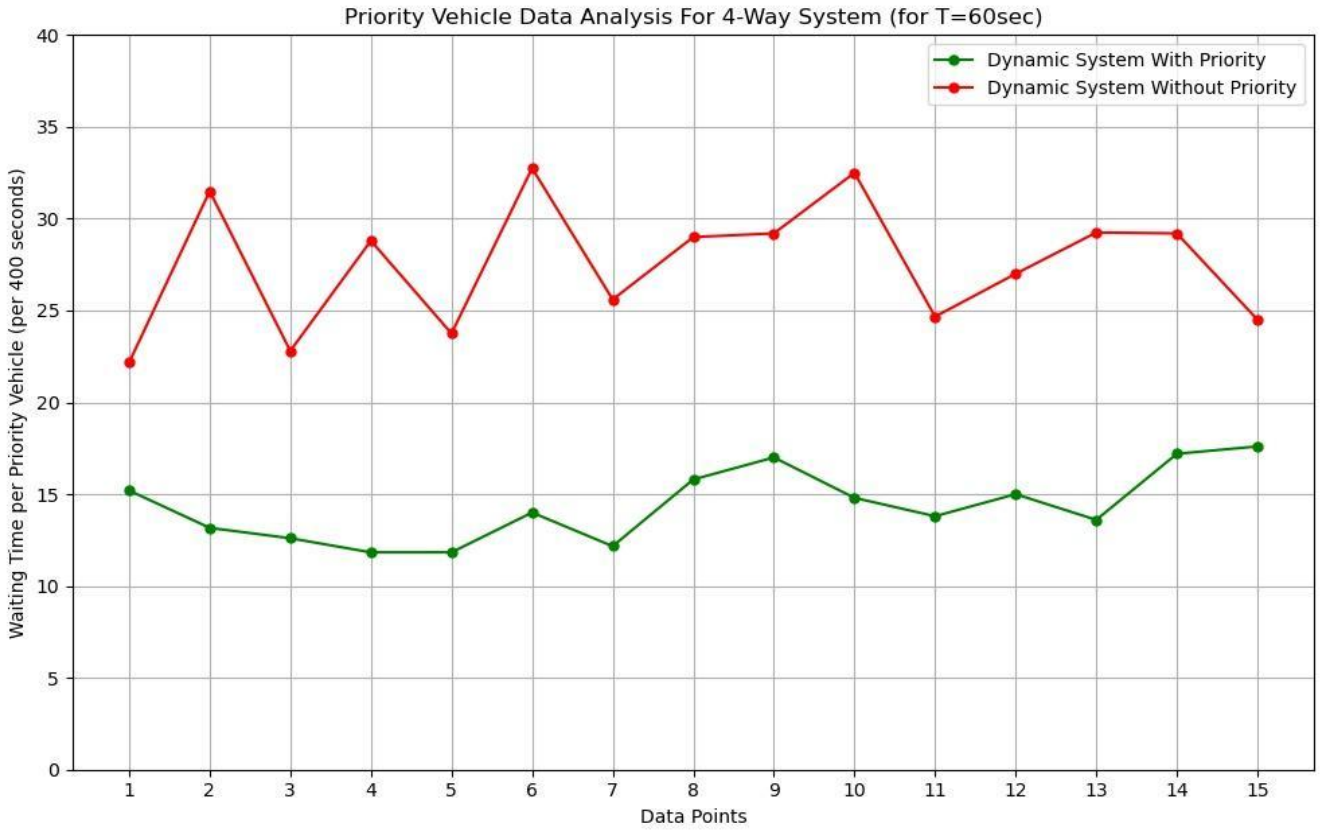

Fig 52: Analysis of Priority Vehicle Waiting Times for T = 60 sec

Plot in Fig 52 shows the analysis of Priority Vehicle waiting times when the priority vehicle arrives after every 60 sec for 4-Way junction system for ITCS+Priority System and ITCS System. The ITCS+Priority System shows good results with a large gap between both the systems.

CHAPTER 6: CONCLUSION AND FUTURE WORK

7.1 Conclusion

In conclusion, our comprehensive traffic management system encompasses multiple key components that work together seamlessly to improve the efficiency and safety of road networks. The Improved Traffic Control System (ITCS) effectively regulates traffic flow at intersections, reducing congestion and optimizing the level of service for drivers. Additionally, our Priority Vehicle Algorithm prioritizes emergency vehicles, enabling them to navigate through traffic more efficiently and enhancing the overall emergency response system.

The Trust Score feature provides drivers with real-time information about the reliability and safety of different lanes. By considering factors such as weather conditions, traffic congestion, and historical accident data, our algorithm calculates trust scores for each lane, empowering drivers to make informed decisions about their route choices. This feature has been seamlessly integrated into our mobile application built with React Native, ensuring convenient access to real-time emergency information and trust scores directly on users' smartphones.

To support our data-driven approach, we have established a robust data infrastructure on AWS EC2 instances. This infrastructure enables efficient storage and retrieval of trust score-related information, facilitating future predictions using machine learning algorithms. By continuously analyzing traffic patterns and improving the accuracy of trust scores, we can make proactive decisions for traffic management, ultimately leading to optimized traffic flow and improved overall travel experience.

In summary, our integrated system leverages advanced technologies, real-time data, and user-friendly applications to create a comprehensive approach to traffic management. Through the synergy of the ITCS, Priority Vehicle Algorithm, Trust Score feature, and mobile application, we aim to enhance traffic flow, emergency response, and driver decision-making. By embracing innovation and data analysis, we strive to create safer, more efficient road networks for the benefit of all users.

7.2 Future Scope

More advancements could be done in Rush Hour which are listed below

- By leveraging machine learning techniques on congestion and weather data collected from AWS, we aim to predict trust scores for Chandigarh's road network, enhancing route planning for drivers.
- Conducting junction mapping and collecting weather and congestion data for neighboring cities of Chandigarh to provide a broader context and more comprehensive insights into traffic conditions.
- Integration of parameters such as pollution levels in addition to congestion and weather as an additional variable in the trust score algorithm, considering the impact on travel experience and health risks for drivers.

- Publishing of the developed mobile application on the App Store and Google Play Store, providing widespread access to real-time trust scores and efficient route planning for users.
- Drafting and publishing of a research paper to contribute to the existing body of knowledge, highlighting the methodology, insights, and findings from our trust score prediction system based on machine learning and data analysis.

REFERENCES

[1] De Oliveira, L. F. P., Manera, L. T., & da Luz, P. D. G. (2020). Development of a Smart Traffic Light Control System with Real-Time Monitoring. IEEE Internet of Things Journal, 1–1. doi:10.1109/jiot.2020.3022392

[2] Kumar, N., Rahman, S. S., & Dhakad, N. (2020). Fuzzy Inference Enabled Deep Reinforcement Learning-Based Traffic Light Control for Intelligent Transportation System. IEEE Transactions on Intelligent Transportation Systems, 1–10. doi:10.1109/tits.2020.2984033

[3] Karmakar, G., Chowdhury, A., Kamruzzaman, J., & Gondal, I. (2020). A Smart Priority Based Traffic Control System for Emergency Vehicles. IEEE Sensors Journal, 1–1. doi:10.1109/jsen.2020.3023149

[4] Sundar, R., Hebbar, S., & Golla, V. (2015). Implementing Intelligent Traffic Control System for Congestion Control, Ambulance Clearance, and Stolen Vehicle Detection. IEEE Sensors Journal, 15(2), 1109–1113. doi:10.1109/jsen.2014.2360288

[5] Application of Image Processing In Road Traffic Control, International Journal of Advanced Research Trends in Engineering and Technology (IJARTET), Vol. 2, Issue 4, April 2015

[6] Traffic Light Control and Violation Detection Using Image Processing, IOSR Journal of Engineering (IOSRJEN), vol. 08, no. 4, 2018, pp. 23-27

[7] Qmulus – A Cloud Driven GPS Based Tracking System for Real-Time Traffic Routing, IOSR Journal of Electronics and Communication Engineering (IOSR-JECE), Vol:1, No:1, 2013

[8] Real Time Traffic Light Control Using Image Processing, Indian Journal of Computer Science and Engineering (IJCSE), Vol:2, No:1, 2011

[9] IoT-based Traffic Signal Control for ambulance, International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249-8958 (Online), Volume-9 Issue-3, February 2020

[10] Smart Traffic Management System, International Journal of Research Publication and Reviews Vol (2) Issue (2) (2021) Page 226-228

[11] Design of Intelligent Ambulance and Traffic Control, International Journal of Innovative Technology and Exploring Engineering (IJITEE) ISSN: 2278-3075 (Online), Volume-2 Issue-5, April 2013

[12] Smart Traffic Control System Using RFID, International Research Journal of Engineering and Technology (IRJET), Volume: 07 Issue: 03, Mar 2020

[13] Automatic Traffic Estimation Using Image Processing, International Journal of Signal Processing, Image Processing and Pattern Recognition Vol. 5, No. 4, December, 2012

[14] Image processing based Adaptive Traffic Control System, IOSR Journal of Electronics and Communication Engineering (IOSR-JECE) ISSN: 2278-2834, ISBN: 2278-8735, PP: 33-37

[15] Controlling Traffic Jam using Feature Detection and Object Detection Technique, IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)e-ISSN: 2278-1684, p-ISSN: 2320-334XPP 07-13

[16] Adaptive Traffic Management For Life Saving Services In Smart Cities, International Journal of Creative Research Thoughts (IJCRT), Volume 10, Issue 11 November 2022

[17] Smart Ambulance with Traffic Control, Journal of Emerging Technologies and Innovative Research (JETIR), June 2020, Volume 7, Issue 6

[18] Advance Alert for Ambulance Pass by using IOT for Smart City, International Journal of Engineering Science and Computing, June 2017

[19] A. Vogel, I. Oremović, R. Šimić and E. Ivanjko, "Improving Traffic Light Control by Means of Fuzzy Logic," 2018 International Symposium ELMAR, Zadar, 2018, pp. 51-56, doi: 10.23919/ELMAR.2018.8534692.

[20] A. A. Zaid, Y. Suhweil and M. A. Yaman, "Smart controlling for traffic light time," 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), Aqaba, 2017, pp. 1-5, doi: 10.1109/AEECT.2017.8257768.

[21] A. Kanungo, A. Sharma and C. Singla, "Smart traffic lights switching and traffic density calculation using video processing," 2014 Recent Advances in Engineering and Computational Sciences (RAECS), Chandigarh, 2014, pp. 1-6, doi: 10.1109/RAECS.2014.6799542.

[22] Siddharth Srivastava, Subhadeep Chakraborty, Raj Kamal, Rahil, Minocha, “Adaptive traffic light timer controller” , IIT KANPUR, NERD MAGAZINE.

[23] TomTom.com, 'Tom Tom World Traffic Index', 2019. [Online]. Available: https://www.tomtom.com/en_gb/traffic-index/ranking/

[24] Pygame Library, 2019. [Online]. Available: https://www.pygame.org/wiki/about

[25] Open Data Science, 'Overview of the YOLO Object Detection Algorithm', 2018. [Online]. Available:https://medium.com/@ODSC/overview-of-the-yolo-object-detection-algorithm-7b52a745d30

[26] Study Of Automatic Traffic Signal System For Chandigarh, International Journal Of Engineering Sciences & Research Technology, (I2or), Publication Impact Factor: 3.785, July, 2015

[27] Intelligent Traffic Management Systems: A Review, International Journal For Innovative Research In Science & Technology, Volume 2, Issue 09, February 2016