GRADUATE DIRECTED PROJECT-2 INTELLIGENT TRAFFIC SIGNALING SYSTEM

November 2023

Northwest Missouri State University,Maryville MO 64468, USA

TEAM MEMBERS
Dheeraj krishna Mekala
Bharath kumar reddy Mopuru
LokeshReddy DonthiReddy
Kovidh Kurra
Siva Bajee Babu Porandla
Sai Bharath Kondepati

Index

Abstract

Traffic management is a key component of contemporary urban infrastructure, with an ever-increasing need for efficient and adaptive solutions. This study proposes an innovative technology that uses image processing and intelligent decision-making to improve traffic management. The system differentiates between emergency and non-emergency cars by taking photos of vehicles at junctions, interpreting text information from these images, and making real-time choices based on the content. In an emergency, the system dynamically changes traffic signal settings to enable safe passage, while non-emergency cars obey conventional traffic norms. This intelligent traffic management system tries to optimize traffic flow while also increasing safety and response times for emergency services.

1 Introduction

Traffic congestion and efficient management of urban transportation networks are challenges faced by cities worldwide. As urbanization continues, the need for innovative solutions to address traffic-related issues becomes increasingly urgent. In response to this demand, our project introduces a novel approach that combines image capture, text processing, and intelligent decision-making to optimize traffic control at intersections.

The basis of this system is its capacity to record photos of automobiles as they approach junctions utilising cameras strategically positioned across the metropolitan area. These photos are then analysed to extract textual information such as licence plate numbers and vehicle marks. This data is used to make real-time determinations regarding whether a vehicle is an emergency vehicle or not.

The decision-making process in our system is critical, since it consults a large database to evaluate the emergency situation of an approaching vehicle. If the car is detected as an emergency vehicle, the system reacts quickly by interrupting conventional traffic signal control and flipping the traffic light to green, allowing the vehicle to proceed through the junction unhindered. In the event that the vehicle is not categorized as an emergency vehicle, the system returns to the picture capturing phase without fail.

By deploying this intelligent traffic management system, we want to improve traffic flow in metropolitan areas, minimise reaction times for emergency services, and, ultimately, increase road safety and efficiency. In the following sections, we will go further into the technical features of our system, discussing the image processing algorithms, decision-making methods, and database architecture. We feel that this novel method has the potential to profoundly alter urban mobility management and emergency response systems.

The website requires users to sign in, and upon successful authentication, it allows them to search for vehicle records based on registration tags. The search results are displayed in a recycler view, and appropriate alerts notify users of the outcome of their search, be it successful or indicating no records found.

2 Project Management Plan

2.1 Team Organization and Schedule

S.NO	TEAM MEMBER	DESIGNATION
1	Dheeraj Krishna Mekala	Team Lead , Developer
2	Bharath Kumar Reddy Mopuru	Developer
3	Lokesh Reddy Donthi Reddy	Database Developer
4	Siva Bajee Babu Porandla	Front-end Developer
5	Kovidh Kurra	Front-end Developer
6	Sai Bharath Kondepati	Back-end Developer

2.2 Risk Management

Technical Risks :

System Malfunction :

The technology might face technical glitches, leading to system failures, and potentially disrupt the flow of regular traffic and emergency vehicle prioritization.

Data Accuracy :

Inaccurate or outdated data used for identifying emergency vehicles could result in wrongful prioritization or non-prioritization, causing potential risks to emergency responses.

Legal and Regulatory Risks

a. Compliance Issues :

The system must comply with local and national traffic regulations. Failure to do so could result in legal consequences or public safety hazards.

b. Privacy Concerns :

Collecting and processing data for the prioritization system might raise privacy issues. Non-compliance with privacy regulations could lead to legal actions or public mistrust.

Ethical Risks

a. Fairness and Equity :

The system might unintentionally favor certain areas or demographics, leading to concerns about fairness and equity in emergency response.

b. Moral Dilemmas :

The system might face moral dilemmas when prioritizing emergencies, especially in situations where multiple emergencies occur simultaneously or where decisions need to be made between different types of emergencies.

2.3 Software Requirements

Vscode :

Visual Studio Code (VS Code) is a popular and versatile source-code editor developed by Microsoft. It's known for its flexibility, extensive features, and a large library of extensions that make it suitable for various programming languages and development tasks.

PyCharm :

PyCharm is a robust Integrated Development Environment (IDE) specifically designed for Python development. Developed by JetBrains, PyCharm offers a comprehensive set of tools for software development, making it a popular choice among Python developers.

MongoDb :

MongoDB is a widely-used, open-source, NoSQL database program that stores data in a flexible, document-based model. It is designed to handle unstructured or semi-structured data, making it particularly suitable for modern applications where data formats can vary and evolve rapidly.

Cascading Style Sheets :

Cascading Style Sheets is a key web design technique used to style and structure the visual appearance of HTML and XML content. It defines how elements of a webpage, including as their layout, colors, fonts, and additional visual elements should be viewed.

Hyper Text Markup Language :

Hyper Text Markup Language is a computer language that is widely used to create online pages and apps. It specifies the meaning of its structure and web content, as well as the layout and appearance of the webpages.

JavaScript :

JavaScript is a versatile programming language primarily used in web development to create dynamic and interactive content on websites.It's an essential part of the web ecosystem and has a wide range of functionalities.

Node.js :

Node.js is a powerful runtime environment that facilitates server-side execution of JavaScript code. It's renowned for its versatility and an array of functionalities, allowing developers to build various applications and handle different tasks efficiently.

Turtle :
The Turtle module is an a common Python a package that allows for the development simple graphics and drawings. It is part of the standard library and is included with most Python installations, thus no extra packages are required to use it.

2.4 Hardware Requirements

Windows or Mac Computer :
This project is mostly completed on a Windows PC. Pycharm, Mongo DB, JavaScript, and Node Js require a Windows or Mac machine to execute.We must check that Windows or Mac matches the minimal criteria for using the version of the aforesaid programs.

Processor :
A multicore Intel the central processing unit is recommended for best development performance.

Memory (RAM) :
The amount of RAM you have impacts the speed of Pycharm, Visual Studio Code, MongoDB, and other development tools. Windows recommends at least 8 GB of RAM, however additional RAM is useful for bigger tasks.

Storage :
A certain amount of free disk space is required for the installation of Pycharm, VS code, MongoDB, and other development tools. Windows recommends a minimum of 50 GB of free space for project files and resources, however more is preferred.

Camera :
A high-resolution camera is necessary to collect images or videos of vehicles in traffic in order to detect an emergency vehicle.

3 Functional Requirements :

Capture Registration number :
The project necessitates the installation of a high-resolution camera at the traffic lights in order to record photographs or videos of the vehicles that will pass through the traffic signals. After gathering photographs at traffic lights, we will create an algorithm to recognize the registration plate.

Send Registration number to database :

The project needs detecting the registration number plate from the collected photos. After detecting the registration plate, store the registration plate number to mongo db database.

Compare Registration Tag with Emergency vehicle data :

After recording the registration number plate in the MongoDB database, we will match the emergency vehicle registration number with the recorded picture registration number to determine whether the vehicle is an emergency vehicle.

Allow Emergency vehicle if it matches data :

If the vehicle data matches the emergency vehicle data, the traffic signals changes to green to the side where the emergency vehicle is on.

Detect number of vehicles :

The algorithm is designed to count the number of vehicles in each traffic lane at the traffic signal.

4 Architecture for Emergency Vehicle Prioritization

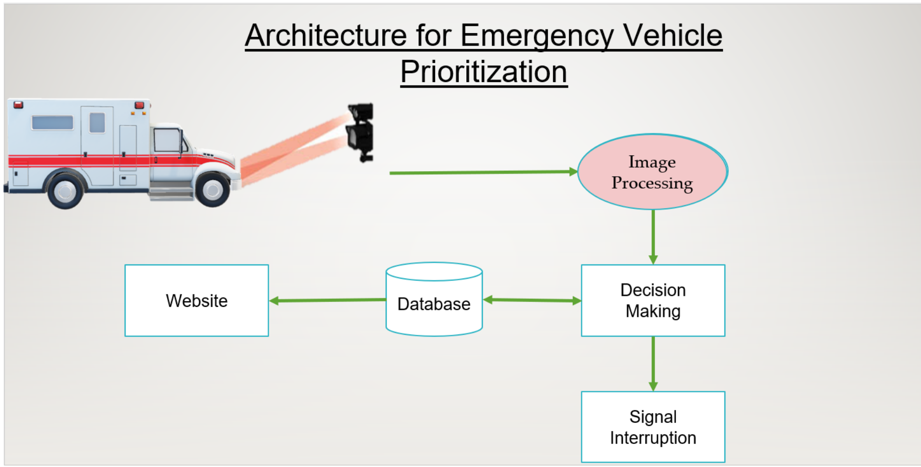

Architecture for Emergency Vehicle Prioritization

Figure 1: Architecture for Emergency Vehicle Prioritization

The image of the vehicle is captured at the traffic signal using camera. The captured image will be given to the RSU(Road Side Unit) for processing the image. In the RSU(Road Side Unit) by using the Haar -cascade Russian plate detection model, registration tag of the vehicle is detected and by using the emergency vehicle detection model, we can find whether the vehicle is emergency vehicle or not.

We are going to compare the registration tag with the data in the database. If the vehicle is found to be a emergency vehicle then the RSU(Road Side Unit) will interrupt the traffic signal from regular operation and the traffic line with emergency vehicle is given green and the remaining all the traffic lines are given red light. After the emergency vehicle is passed through the signal the traffic signal operation will be regular. If the vehicle is not found to be an emergency vehicle, then the traffic signal is not interrupted from its regular operation.

5 Image Processing

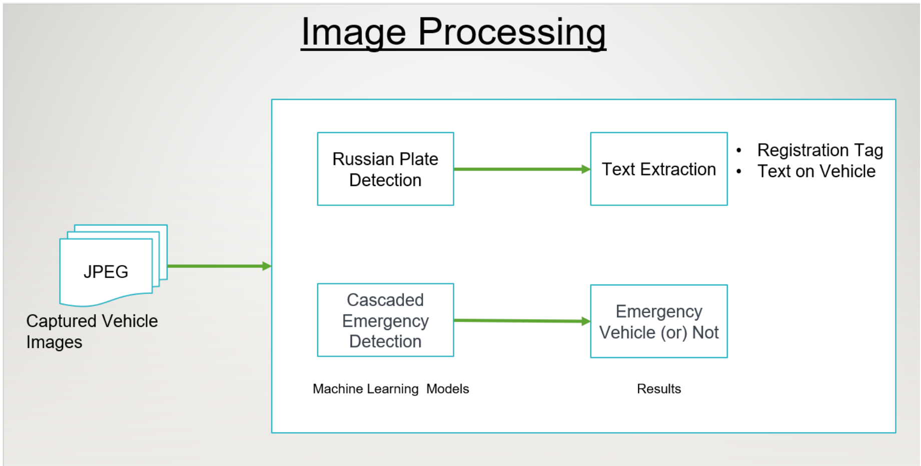


Figure 2: Image Processing Algorithm

The images of the vehicle captured at the traffic signal are received from the camera in the jpeg format. The jpeg images are then given to the two machine learning models namely Haar-cascade Russian plate detection model and cascaded emergency vehicle detection model.

Haar-cascade Russian plate detection model is used to detect the alphanumerical characters from the images. Haar-cascade Russian detection model is a cascade function is learned using many both positive and negative pictures. The notion of an integral image, also known as a summed-area table, is used to quickly extract numerical values for features such as edges and lines. This method is more efficient than the traditional method of subtracting sums of pixels over different parts of a picture, which requires a lot of computing power.

The results of this model not only include the registration tag but also contains the text which is in similar size to that of the registration tag polygon. That means the alphanumeric characters that are in the similar size of the registration tag are also extracted from the image.

The cascaded Russian plate detection model is pretrained by using different emergency vehicle images from different angles and different distances. The model will be used to directly detect whether the vehicle in the image is an emergency vehicle or not.

6 Decision Making

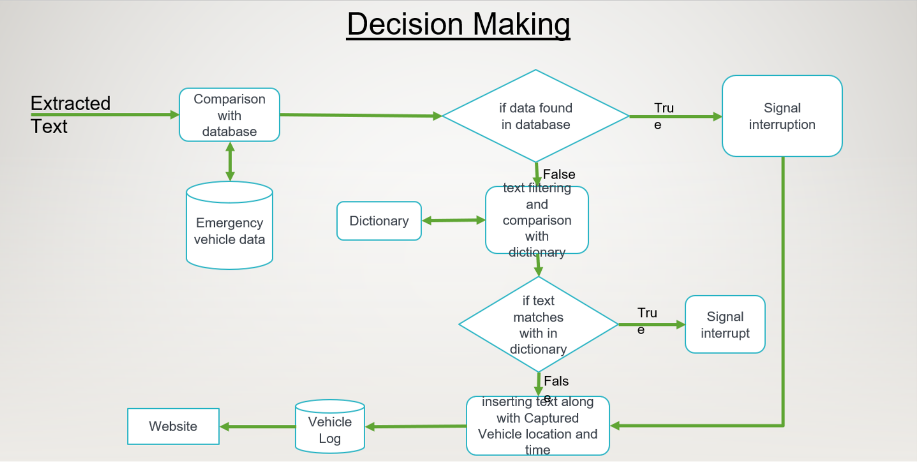

Decision Making
Extracted Text
Comparison with database
Emergency vehicle data
if data found in database
True
Signal interruption
False
Dictionary
text filtering and comparison with dictionary
if text matches with in dictionary
True
Signal interrupt
False
inserting text along with Captured Vehicle location and time
Vehicle Log
Website

Figure 3: Decision Making Algorithm

The database contains two collections namely “Emergency vehicle data” and “vehicle log”. The “Emergency vehicle data” collection contains all the registration tag of the of all the emergency vehicles.

The “vehicle log” collection contains the registration tag of all the vehicles that passed through the junction along with the time and location where the image is captured. The text which is detected by the machine learning model is compared with the values in the “Emergency vehicle data” collection. If the text detected is found in the collection, then the vehicle is concluded as the emergency vehicle and the traffic signal is interrupted. The registration tag is then stored in the “vehicle log” collection along with time and location information.

If the text detected is not found in the “Emergency vehicle data” collection then the text undergo filtration and the value is compared with the Emergency vehicle python dictionary. The Emergency vehicle python dictionary contains all the possible strings like the string present on the emergency vehicles. If the filtered text matches with the string in the dictionary, then the vehicle is concluded as the emergency vehicle and the traffic signal is interrupted. The traffic line where the emergency vehicle detected is found is given a green signal and all the rest of the traffic lines are given red signal. In this case the text is not stored in the “vehicle log” collection because it is not the registration tag, it is just the text on the vehicle. If the text does not match the dictionary, then the value is stored in the “vehicle log” collection along with the time and location. It is stored in the collection because it does not match with the dictionary and “Emergency vehicle data” so the vehicle is concluded to be a non-emergency vehicle.

7 Website

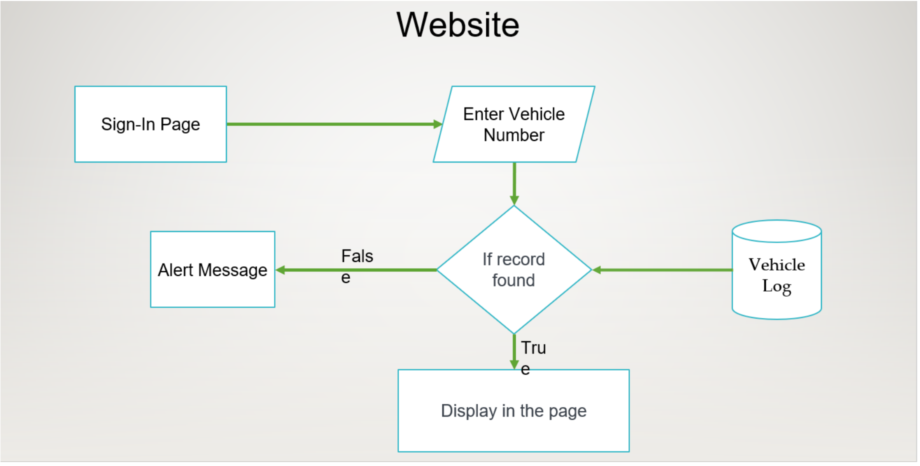
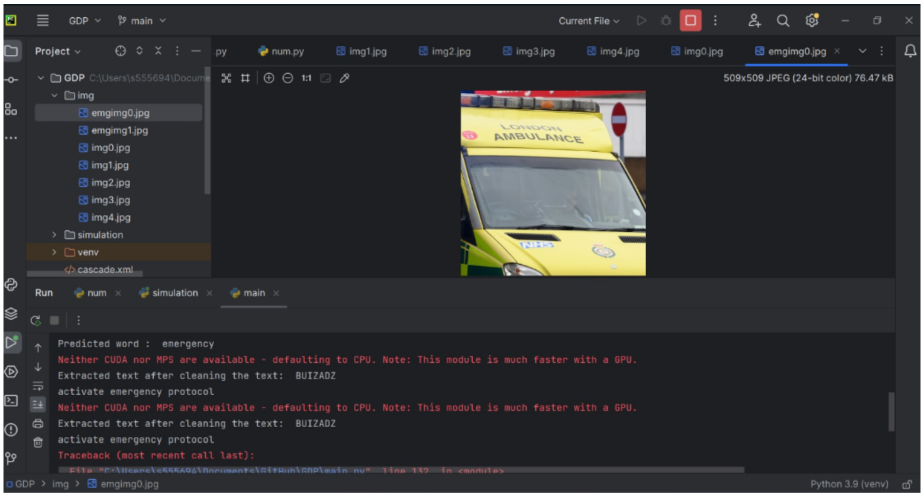


Figure 4: Website Algorithm

There will be sign in page and search page on the website. In the search page there will be a text box, search button, recycler view. The website is developed using html,css, javascript and react js.

The functionality of the website is explained in the rest of the document. We must sign in with the credentials, then we will be redirected to the search page. If the credentials are incorrect, then an alert will pop up with the message ‘credentials incorrect’. After successfully entering the search page, we have to enter the registration tag which we are going to search for in the text box present in the webpage. After clicking the search button, the registration tag will be searched in the “vehicle log” collection and all the records which matches the entered registration tag will be displayed in the recycler view of the website. If no records are found in the “vehicle log” collection, then an alert will be popped up with the message ‘no record found’.

8 Results and Testing :

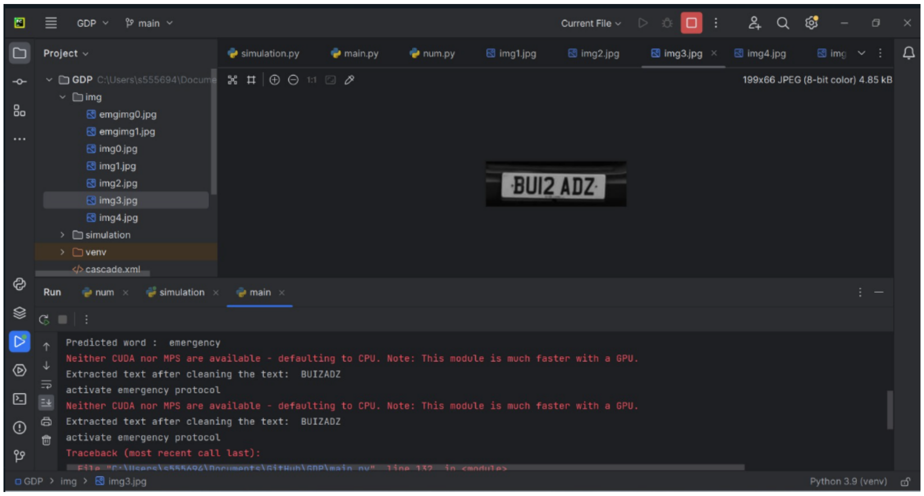


```
Predicted word :  emergency
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
Extracted text after cleaning the text:  BUIZADZ
activate emergency protocol
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
Extracted text after cleaning the text:  BUIZADZ
activate emergency protocol
Traceback (most recent call last):
  File "C:\Users\s555694\Documents\GitHub\GDP\main.py", line 132, in <module>
```

GDP > img > emgimg0.jpg

Figure 5:

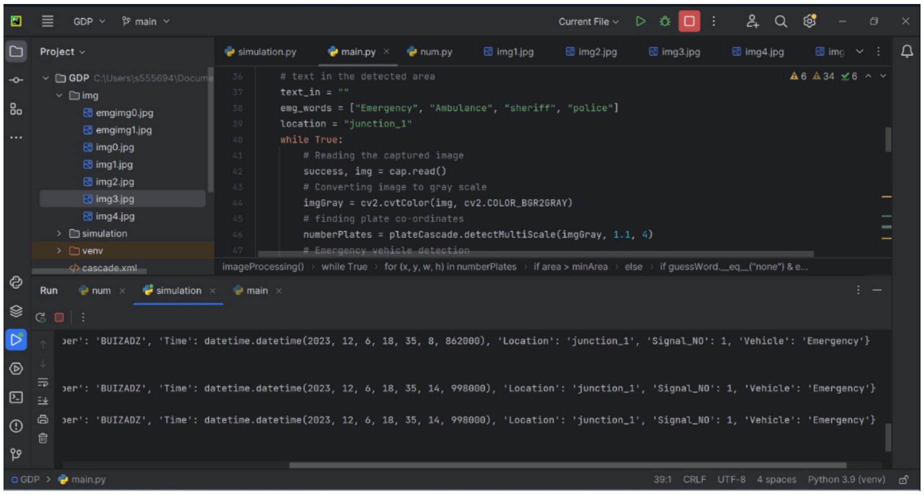
In the figure 5, an image is given to the program and the machine learning model, and the model started detecting the polygons of specified size across the image. The text is then extracted from the polygons and the results are displayed in the console panel.



```
Predicted word :  emergency
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
Extracted text after cleaning the text:  BUIZADZ
activate emergency protocol
Neither CUDA nor MPS are available - defaulting to CPU. Note: This module is much faster with a GPU.
Extracted text after cleaning the text:  BUIZADZ
activate emergency protocol
Traceback (most recent call last):
  File "C:\Users\s555694\Documents\GitHub\GDP\main.py", line 132, in <module>
```

Figure 6:

In the figure 6, the machine learning models extracted the polygon from the image which is similar to the size of a registration tag and contains strings similar to that of the registration tag.



```
    # text in the detected area
    text_in = ""
    emg_words = ["Emergency", "Ambulance", "sheriff", "police"]
    location = "junction_1"
    while True:
        # Reading the captured image
        success, img = cap.read()
        # Converting image to gray scale
        imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        # finding plate co-ordinates
        numberPlates = plateCascade.detectMultiScale(imgGray, 1.1, 4)
        # Emergency vehicle detection
```

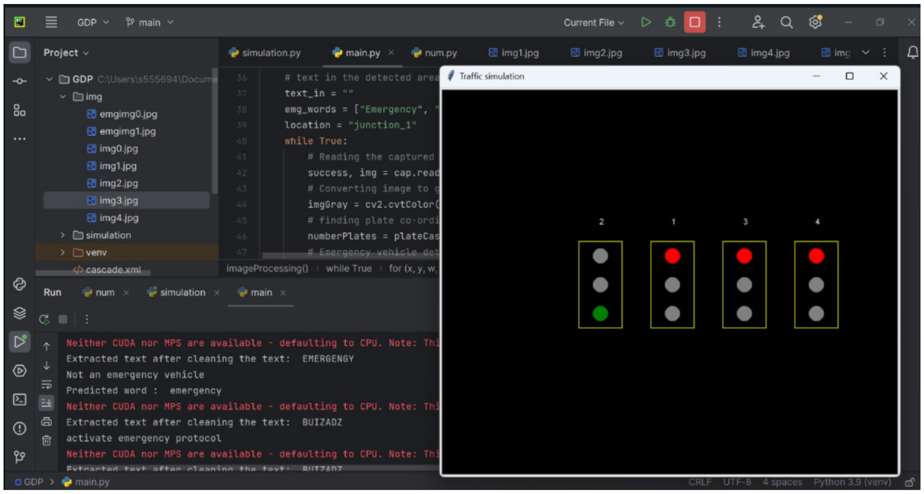
```
ber': 'BUIZADZ', 'Time': datetime.datetime(2023, 12, 6, 18, 35, 8, 862000), 'Location': 'junction_1', 'Signal_NO': 1, 'Vehicle': 'Emergency'}


ber': 'BUIZADZ', 'Time': datetime.datetime(2023, 12, 6, 18, 35, 14, 998000), 'Location': 'junction_1', 'Signal_NO': 1, 'Vehicle': 'Emergency'}

ber': 'BUIZADZ', 'Time': datetime.datetime(2023, 12, 6, 18, 35, 14, 998000), 'Location': 'junction_1', 'Signal_NO': 1, 'Vehicle': 'Emergency'}
```

Figure 7:

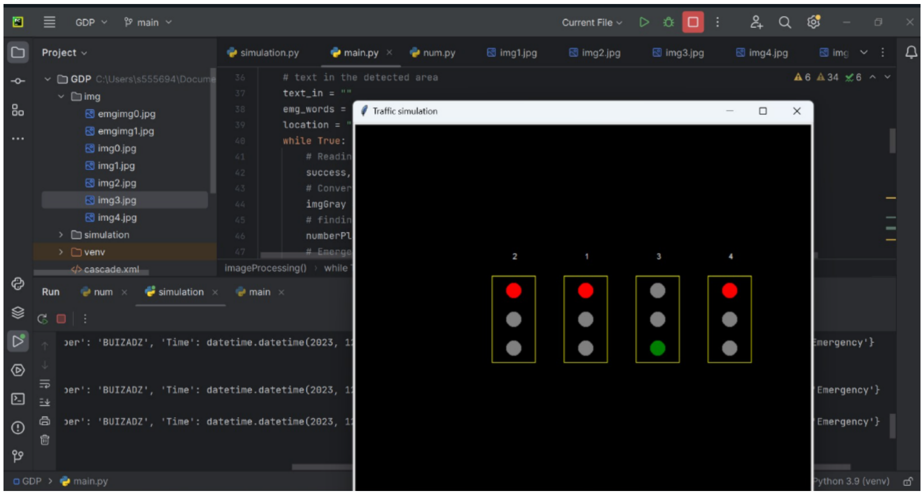
The figure 7 represents the successful extracting the text the polygon. The text along with the time stamp, location, vehicle type, and signal number are written in the database in the format displayed in the console panel.



```
# text in the detected area
text_in = ""
emg_words = ["Emergency", "
location = "junction_1"
while True:
    # Reading the captured
    success, img = cap.read
    # Converting image to g
    imgGray = cv2.cvtColor(
    # finding plate co-ordi
    numberPlates = plateCas
    # Emergency vehicle det
```

```
Neither CUDA nor MPS are available - defaulting to CPU. Note: Thi
Extracted text after cleaning the text:  EMERGENGY
Not an emergency vehicle
Predicted word :  emergency
Neither CUDA nor MPS are available - defaulting to CPU. Note: Thi
Extracted text after cleaning the text:  BUIZADZ
activate emergency protocol
Neither CUDA nor MPS are available - defaulting to CPU. Note: Thi
Extracted text after cleaning the text:  BUIZADZ
```

Figure 8:



```
# text in the detected area
text_in = ""
emg_words =
location = "
while True:
    # Readin
    success,
    # Conver
    imgGray
    # findin
    numberPl
    # Emerge
```

```
ɔer': 'BUIZADZ', 'Time': datetime.datetime(2023, 1          Emergency'}


ɔer': 'BUIZADZ', 'Time': datetime.datetime(2023, 1          'Emergency'}

ɔer': 'BUIZADZ', 'Time': datetime.datetime(2023, 1          'Emergency'}
```

Figure 9:

The figure 8 and figure 9 represents the signal simulator which represents the working the of the traffic signals. In the figure 8 the emergency vehicle is not detected and the traffic signals are performing without any interruption. In the figure 9 the emergency vehicle is detected in the traffic line 2 so instead of changing the traffic line 4 to green traffic line 2 is changed to green traffic.

Non-functional testing :

Here we are using the Chrome Developer tools to test the website PFA , screenshots for details .

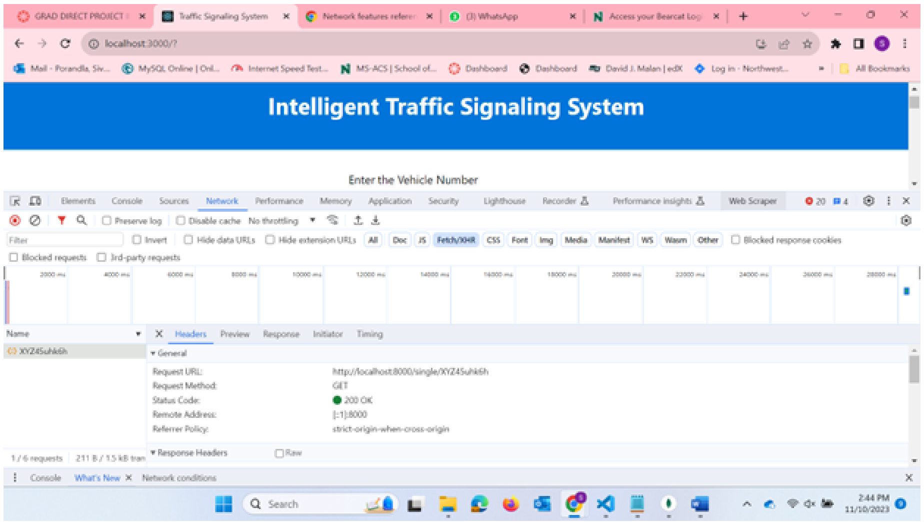
Functional testing :

At the End we are doing functional testing of a website I mean in the UI Components and its easy understandable for the Consumer .

Non-functional testing :

Network :

Here the user is trying to retrieve the data for the single resource which is XYZ45uhk6h from the server running on localhost:8000 and here the request method is GET which means the user is trying to retrieve the data and not modifying it . The status code is 200 ok which means the server is server is able to process the request and return the resource



Intelligent Traffic Signaling System

Enter the Vehicle Number

Name: XYZ45uhk6h

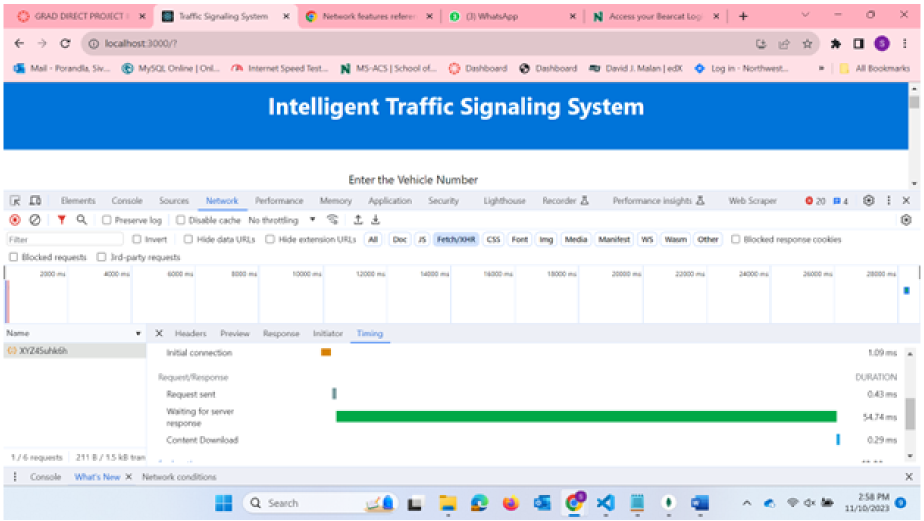
▾ General

Request URL: http://localhost:8000/single/XYZ45uhk6h
Request Method: GET
Status Code: ● 200 OK
Remote Address: [::1]:8000
Referrer Policy: strict-origin-when-cross-origin

▾ Response Headers

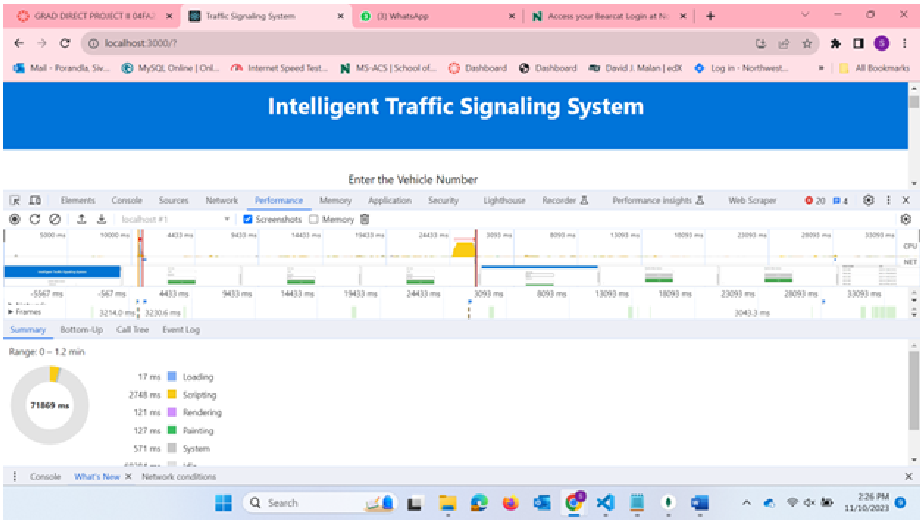
1 / 6 requests | 211 B / 1.5 kB tran

Figure 10: Waiting time for the server response is 54.74 ms from the mongo Db is good .


Intelligent Traffic Signaling System
Enter the Vehicle Number
localhost:3000/?
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights Web Scraper
Headers Preview Response Initiator Timing
XYZ45uhk6h
1 / 6 requests 211 B / 1.5 kB tran
Console What's New Network conditions

Initial connection	1.09 ms
Request/Response	DURATION
Request sent	0.43 ms
Waiting for server response	54.74 ms
Content Download	0.29 ms

Figure 11: **Performance :**The Total time spent on all the tasks is between 0-1.2 is actually a good thing that website is performing well .


Intelligent Traffic Signaling System
Enter the Vehicle Number
localhost:3000/?
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights Web Scraper
localhost #1
☑ Screenshots ☐ Memory
Summary Bottom-Up Call Tree Event Log
Console What's New Network conditions

Range: 0 – 1.2 min

71869 ms

17 ms	Loading
2748 ms	Scripting
121 ms	Rendering
127 ms	Painting
571 ms	System

Figure 12:

Functional testing :

Here we are doing the smoke testing to test the basic functionality of the website.

1.The Website Looks Good and "Responsive "on Different Screen Sizes. 2.After Entering VIN number user is able to fetch the data like Vehicle Number, time and location. 3.User can see the results in csv format.

Here are some changes for more user friendly application 1.Enter Vehicle Number label could be more specific Such as Enter License plate number 2.Search button Could be more descriptive

System testing :

This type of testing focuses on testing the entire software system as a whole.

I mean here we are combining modules like opencv and mongodb and website and testing all three at once.

Testing the OpenCV :

1.Object Detection Algorithm is chosen Successfully. 2.Loaded object detection model to the open cv successful. 3.Read the image by using open cv successfully. which is the result would be alpha numberic and only alphabets I mean like Department. 4.Output is displayed successfully in the terminal with number plate .

Testing mongoDB against website :

All the attributes in mongodb and website is same and fetched successfully by using Get method .

9 Conlusion

In conclusion, the intelligent traffic signaling system is an new innovative solution for prioritizing emergency vehicles and ensure a rapid response for emergency vehicles. By seamlessly integrating with existing traffic infrastructure, the program minimizes delays for emergency services and potentially life-saving interventions during emergencies. The program's ability to accurately detect emergency vehicles in real-time, coupled with its intelligent traffic signal interruption mechanism, not only optimizes emergency response times but also its future capabilities contributes to overall traffic management and congestion reduction.The Intelligent traffic signaling system in a new approach towards gathering the traffic and vehicle information through a location.

10 References

https://opencv.org/university/
https://amin-ahmadi.com/cascade-trainer-gui/
https://www.youtube.com/watch?v=VanxJiUwOrQ
https://github.com/isha-git/GreenGo
https://github.com/kennethleungty/Car-Plate-Detection-OpenCV-TesseractOCR