FLAREPATH – ADVANCED VEHICLE FIRE SAFETY AND MONITORING WITH RAPID EMERGENCY DISPATCH SOLUTIONS

R24-058

Status Document - 2

Anthick G.N – IT21096266

B.Sc. (Hons) Degree in Information Technology specializing in

Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

September 2024

Group Details

Supervisor – Mr.Nelum Chathuranga Amarasena

Co-supervisor – Mr. Deemantha Nayanajith Siriwardana

External Supervisor – Mr. Onray Sahinda

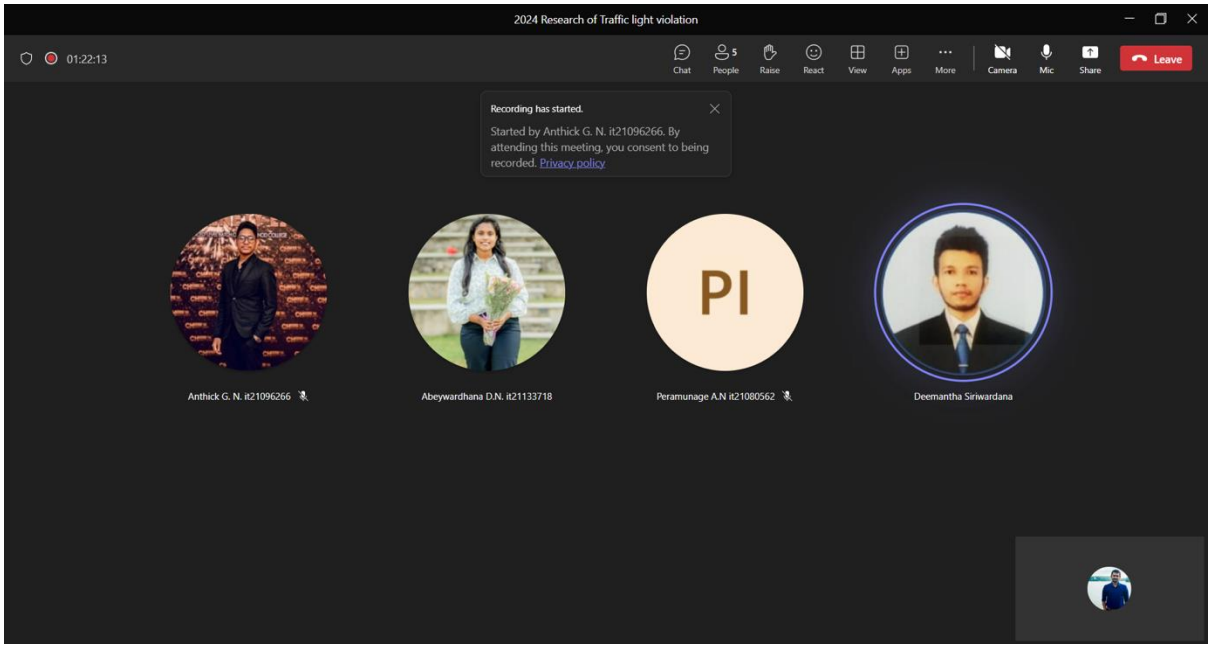
Student Name	**Student ID**	**Contact No**	**Email Address**
Anthick G.N	IT21096266	0779820516	It21096266@my.sliit.lk
Dharmagunawardana W.M.P.I	IT21132346	0772785361	it21132346@my.sliit.lk
Peramunage A.N	IT21080562	0713999266	it21080562@my.sliit.lk
Abeywardhana D.N	IT21133718	0714057155	it21133718@my.sliit.lk

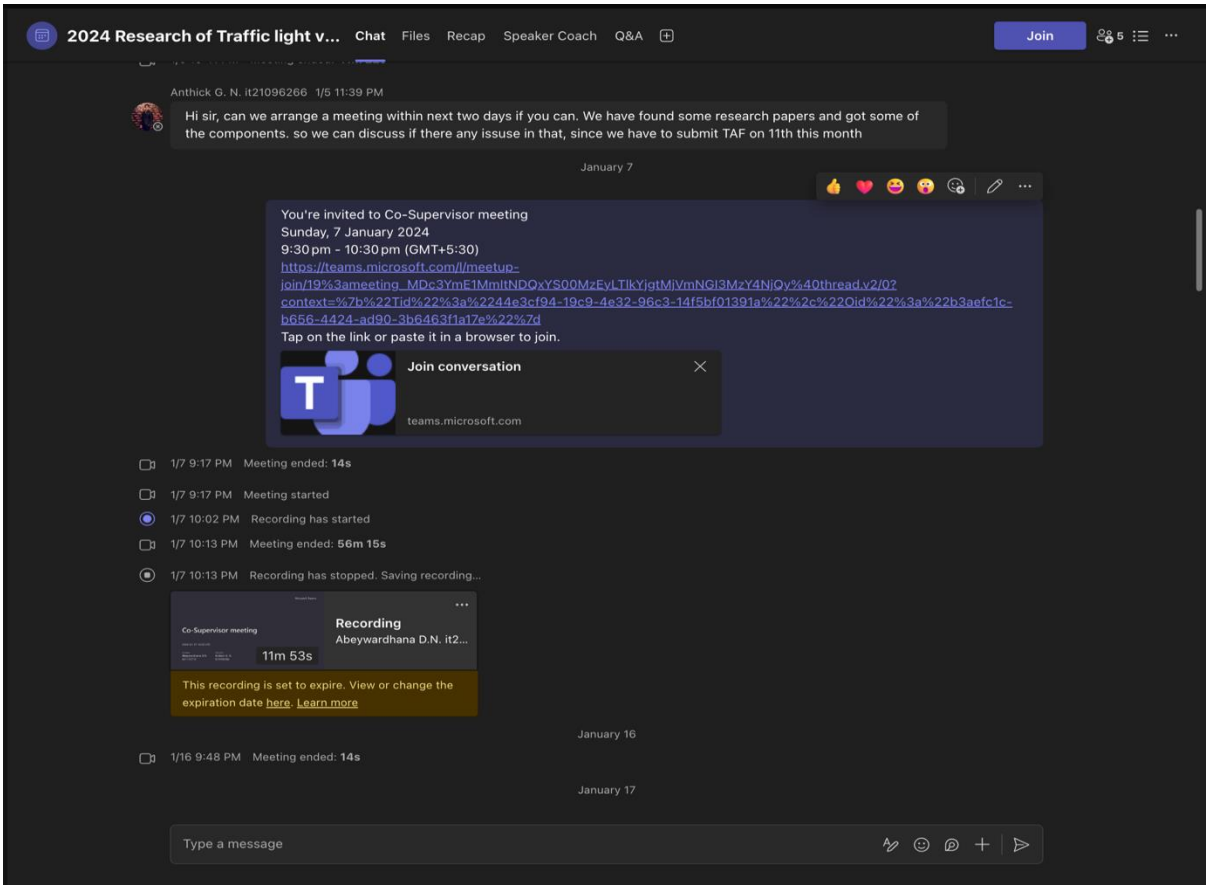
Table of Contents

Meetings & Calls

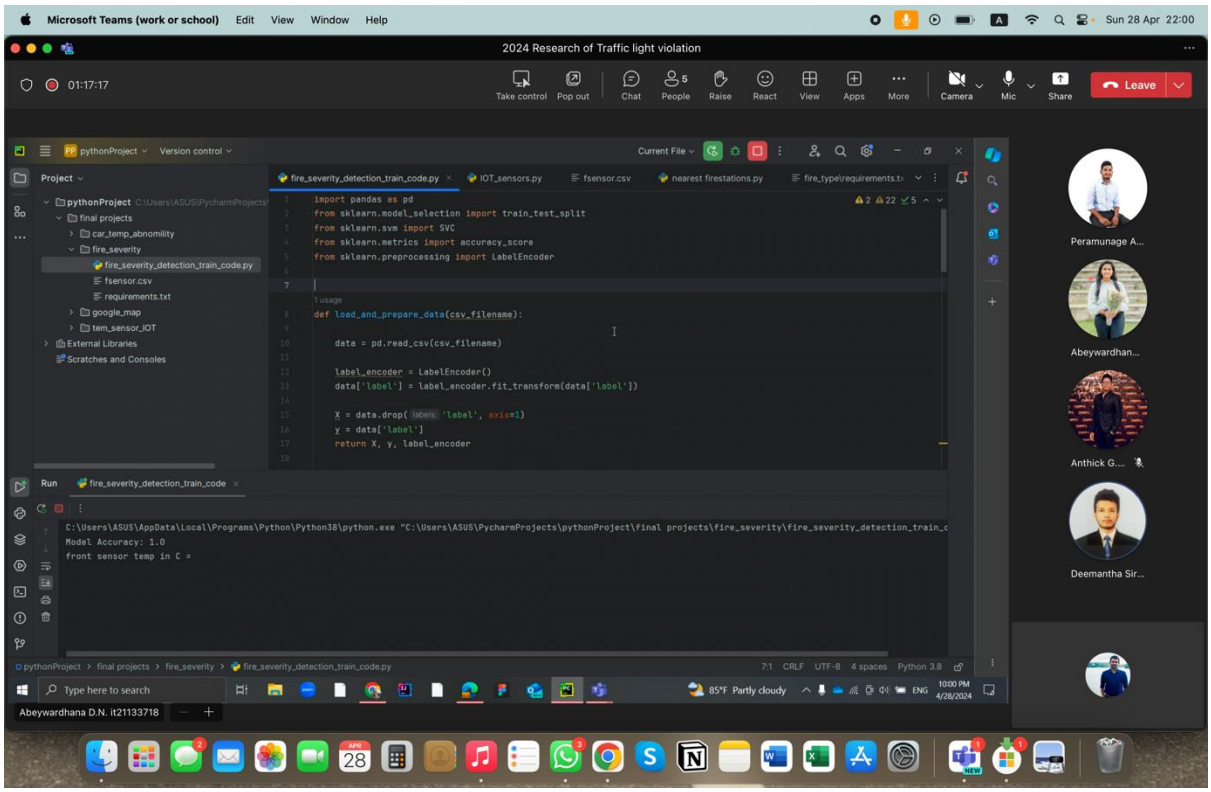
Meetings with supervisor and co-supervisor

Meeting with both supervisor and co-supervisor about the project progress and improvements that we need to do to our project.

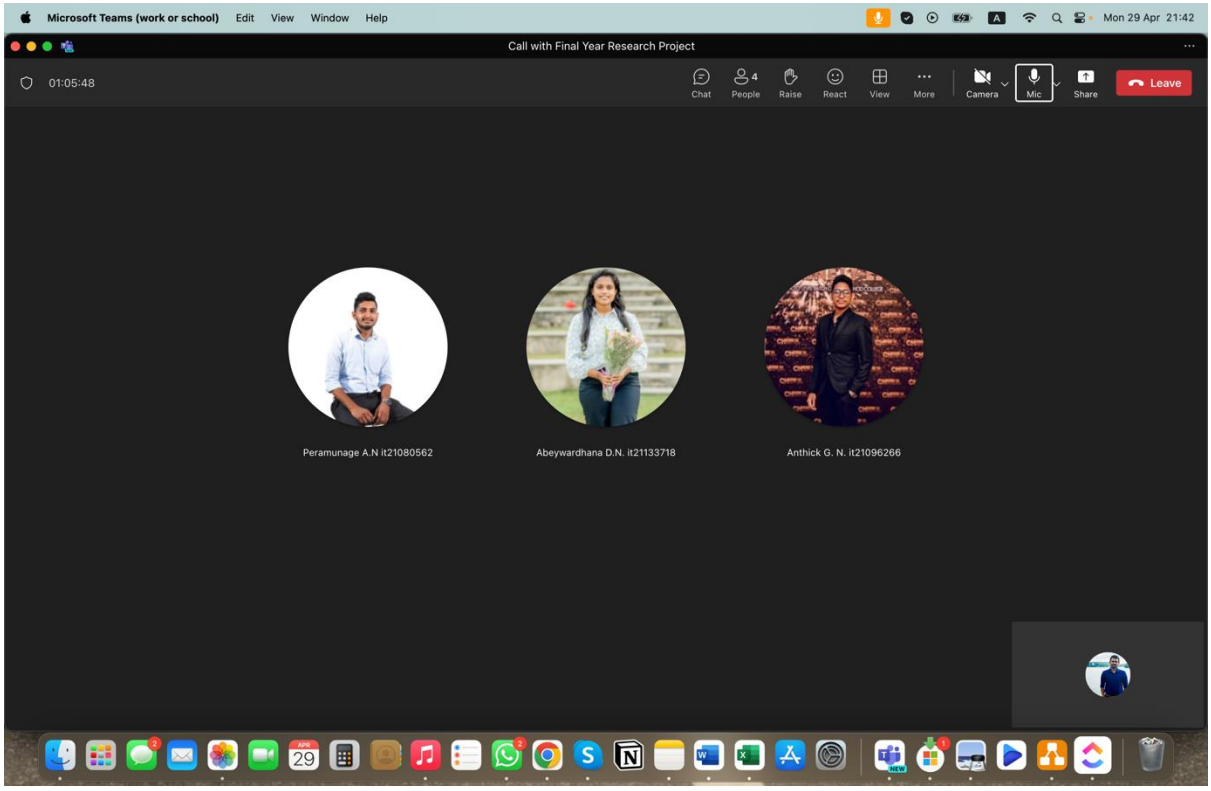




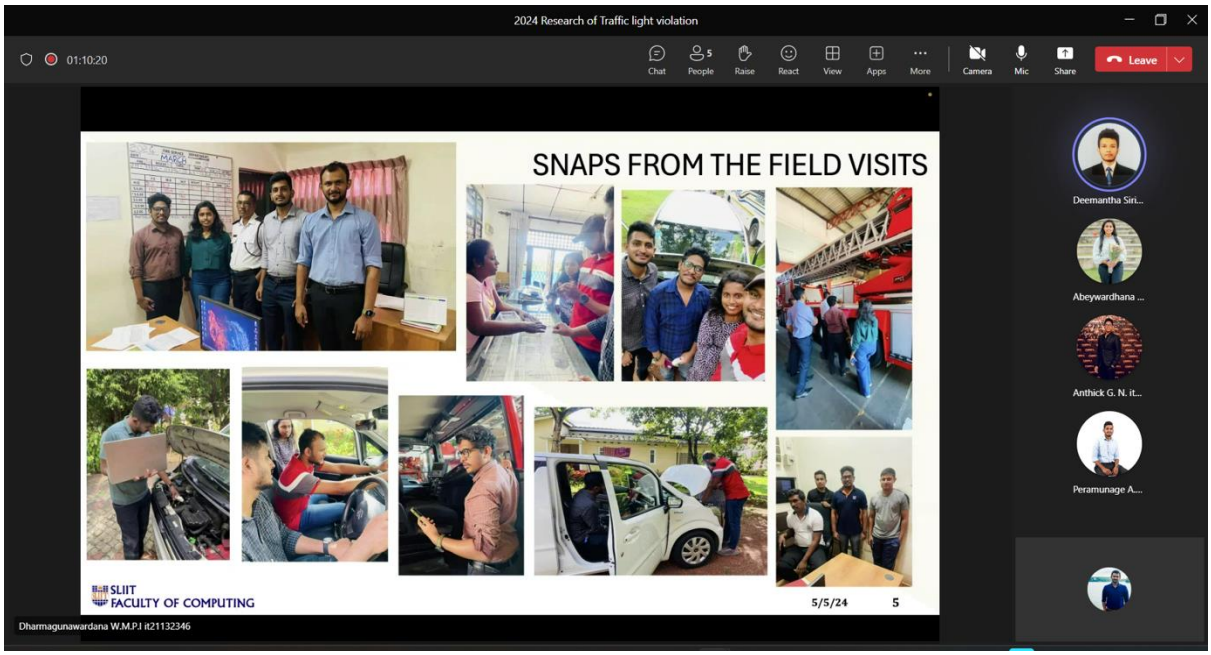
Code review with co-supervisor



Group meeting with group members



Presentation review with supervisor



Meetings with Domain Experts

Meeting with fire department officers and staff.

Mr.Nanayakkara the chief officer of the fire department and we discussed about the domain knowledge and requirements.

Snapshots from Field Visit

2022 FIRE CALLS AND OTHER SERVICES

	january	february	march	april	may	june	july	august	september	october	november	december	TOTAL
FIRE CALL	38	31	48	10	24	12	25	25	24	21	16	25	299
RESCUE CALL	2	1	5	2	1	2	0	2	4	2	3	2	26
EMERGENCY CALL	6	1	0	1	8	0	0	3	2	9	2	5	37
AMBULANCE CALL	0	1	2	0	2	0	1	2	0	1	6	9	24
VIP DUTIES	37	37	45	1	1	2	15	33	32	38	32	33	306
SPECIAL SERVICE	24	19	12	11	2	7	1	6	6	5	10	20	123
TEST CALL	3	3	3	0	1	5	1	2	5	5	7	9	44
INSPECTION OF DANGER PLASE	0	0	0	0	0	0	0	0	0	39	16	40	95
TOTAL	110	93	115	25	39	28	43	73	73	120	92	143	954

prepared by - K.T.S.Fernando

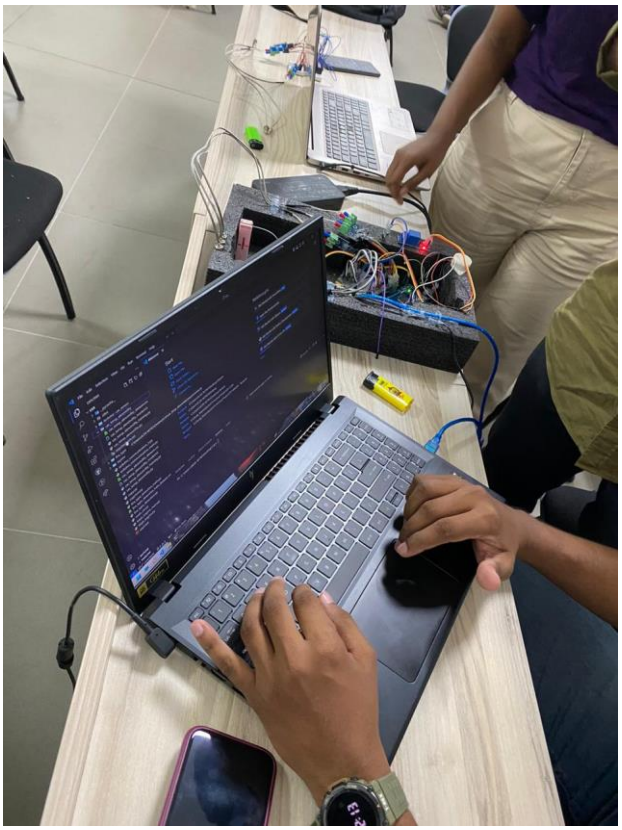
K.P.P.R Nanayakkara
Control Room officer

A.P.J.Preethilal
Station officer
(Communucation)

W.S.R.N Senanayake
Divisional fire officer
(Operation)

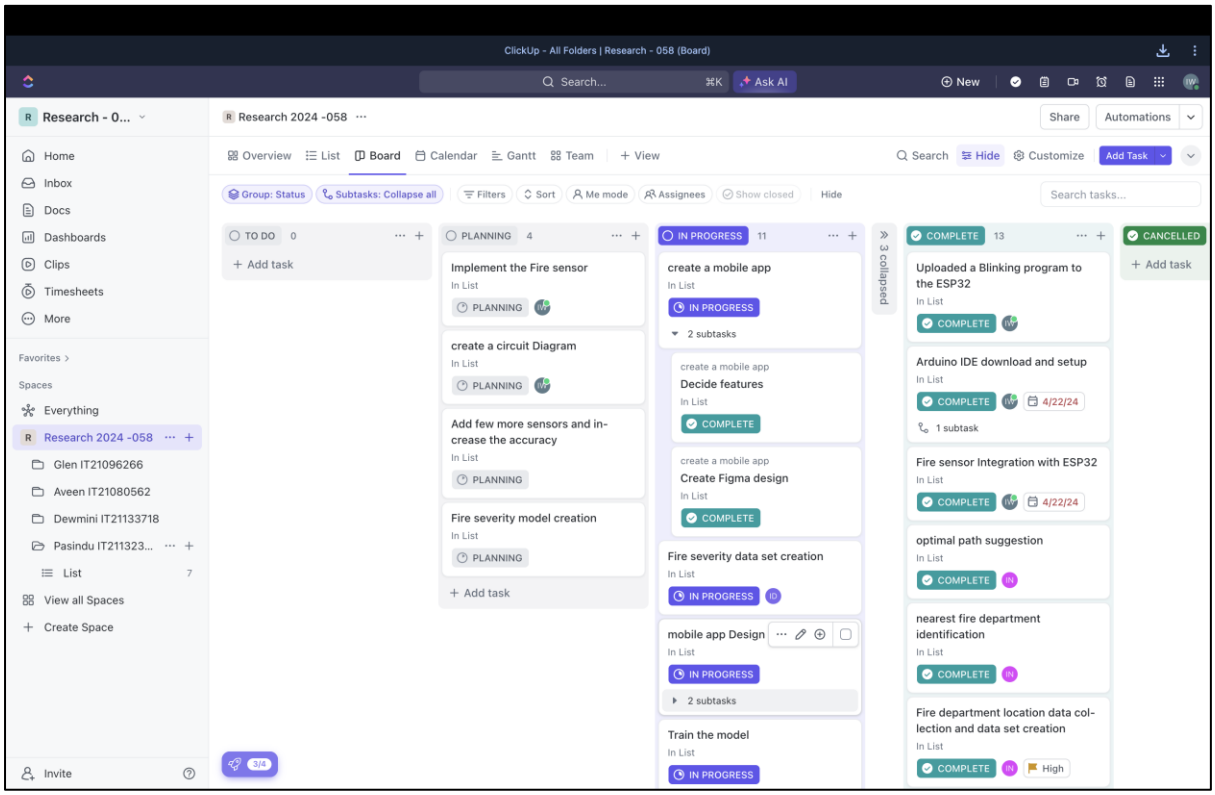
P.D.K.A.Wilson
Chief fire officer


FIRE
MAN
PZA 0019

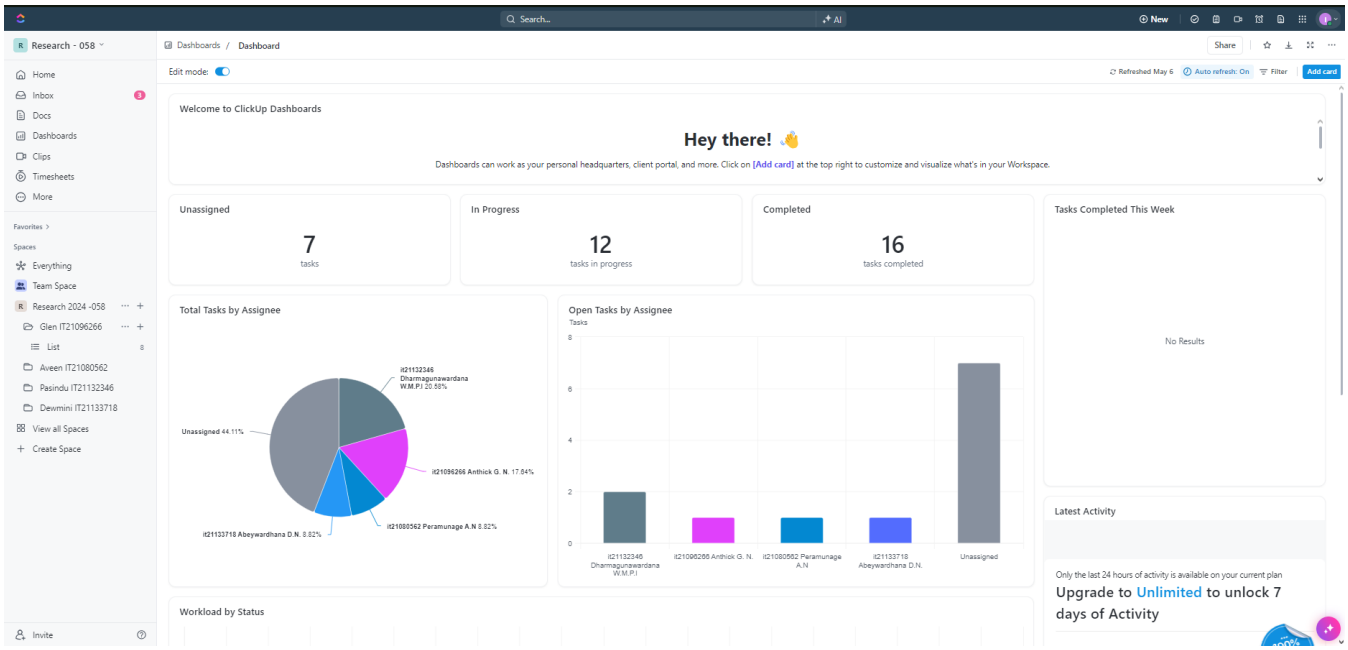



FIRE & RESCUE
COLOMBO

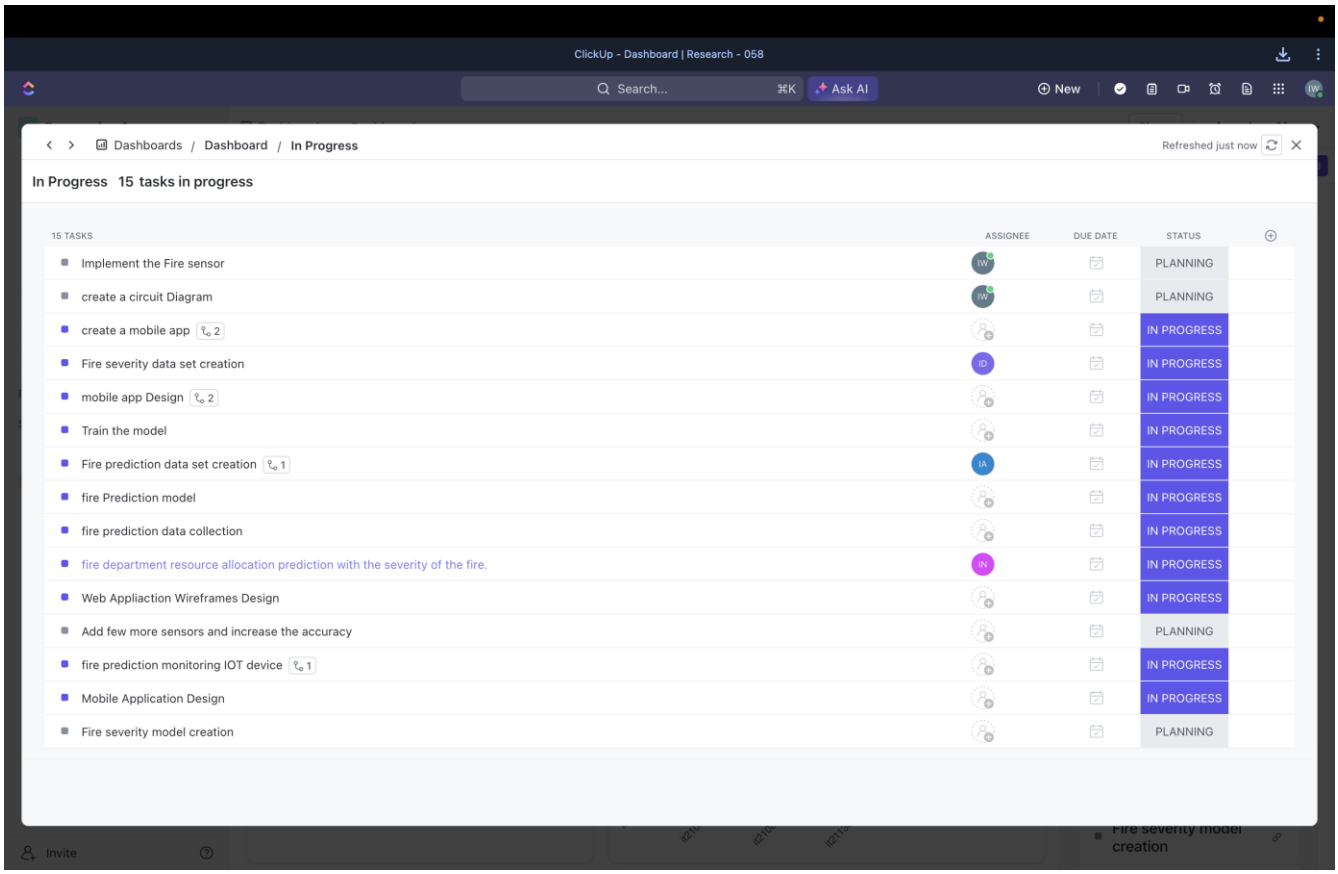
Click Up Tasks Allocation



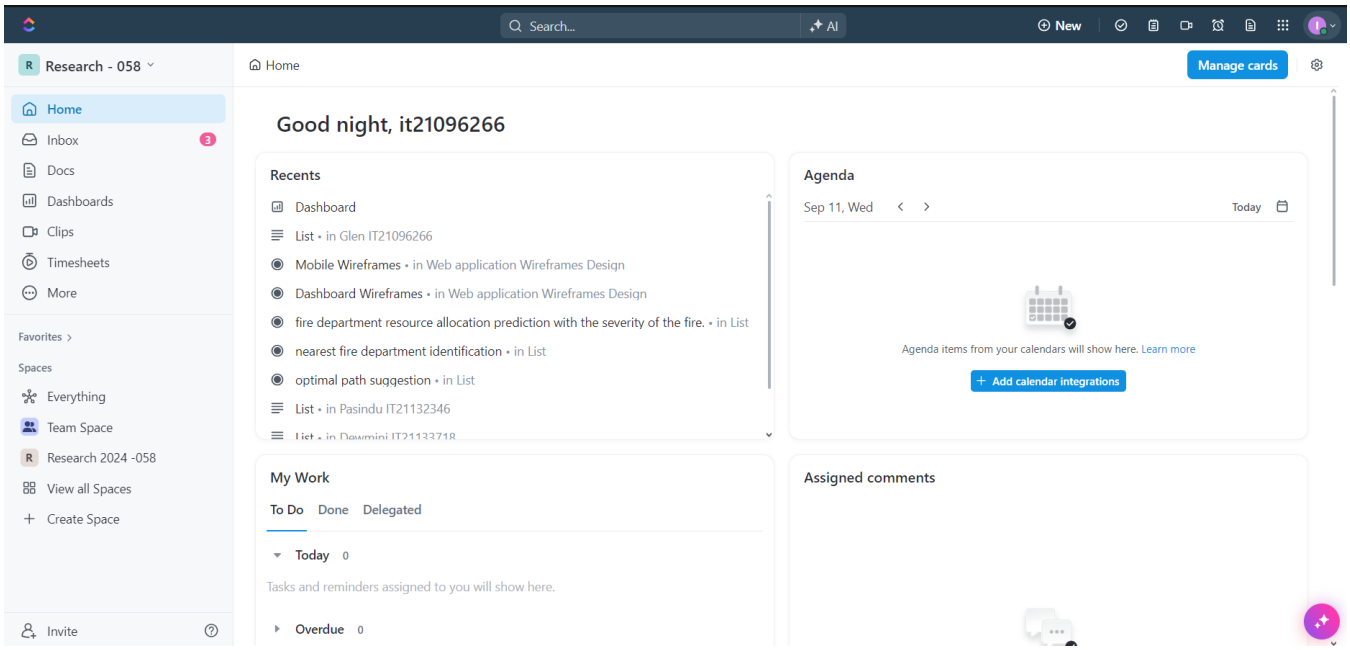
Click Up Dashboard



In Progress Tasks



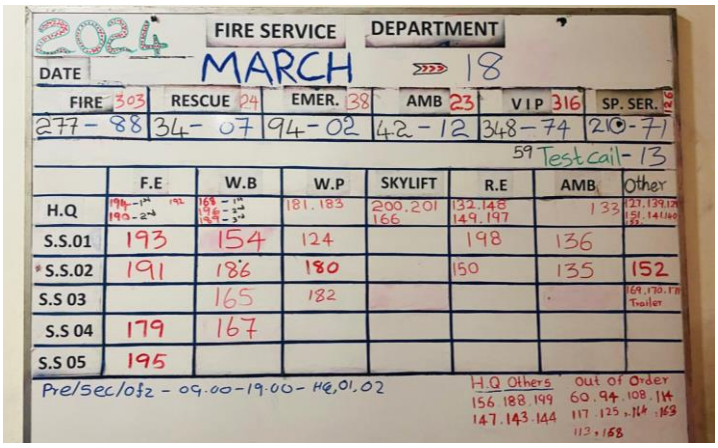
Completed Tasks up to PP2



Project Implementation

Data Collection

Collecting data from fire department



Fire department vehicle dataset

vehicle_registered_number	department_registered_number	station	vehicle_type	fuel_capacity_liters	fuel_consumption_Km/L	approved_monthly_fuel_quantity_liters	water_capacity_height	foam_capacity_height	max_height_lift	none_user_ve
41-3154	60	H	Water Carrier	99	3	unlimited	6750 L	N/A	N/A	N
41-6595	94	H	Water Carrier	116	3	unlimited	6750 L	N/A	N/A	N
41-9077	108	3	T T Ladder	200	1.5	unlimited	N/A	N/A	45 M	Y
43-6259	113	1	Canter	90	6	unlimited	N/A	N/A	N/A	N
47-2272	114	3	Canter	90	6	unlimited	N/A	N/A	N/A	N
226-2485	117	1	Sky Lift	300	2	unlimited	N/A	N/A	54 M	Y
JO-9877	124	1	Canter	90	6	unlimited	N/A	N/A	N/A	Y
LA-1272	125	H	Water Carrier	300	3	unlimited	8000 L	N/A	N/A	N
KA-9955	127	H	Car	50	10	200	N/A	N/A	N/A	Y
253-0090	128	H	Double Cab	70	7	150	N/A	N/A	N/A	N
PA-7804	132	H	Rescue Cab	75	5	unlimited	N/A	N/A	N/A	Y
LW-0270	133	H	Ambulance	70	6	300	N/A	N/A	N/A	Y
LW-0271	134	H	Ambulance	70	6	300	N/A	N/A	N/A	Y
LW-0272	135	2	Ambulance	70	6	300	N/A	N/A	N/A	Y
LW-0273	136	1	Ambulance	70	6	300	N/A	N/A	N/A	Y
KE-6158	139	H	Jeep	180	5	unlimited	N/A	N/A	N/A	Y
NA-5585	140	H	Mini Bus	70	6	unlimited	N/A	N/A	N/A	Y
NA-5586	141	H	Mini Bus	70	6	unlimited	N/A	N/A	N/A	Y
LF-1098	143	3	Command Cont	100	5	unlimited	N/A	N/A	N/A	Y
LF-1114	144	H	Mobile Worksh	100	5	200	N/A	N/A	N/A	Y
LF-1136	147	H	Boom Truck	200	7	500	N/A	N/A	N/A	Y
LW-0336	148	H	R I V	100	6	500	N/A	N/A	N/A	Y
LW-0343	149	2	R I V	100	6	500	N/A	N/A	N/A	Y
LW-0347	150	3	R I V	100	6	500	N/A	N/A	N/A	Y
PB-5971	151	H	Double Cab	80	8	150	N/A	N/A	N/A	Y
LF-7192	152	2	High Angle Resc	180	6	150	N/A	N/A	N/A	Y

Extinguishes dataset

no	name	Latitude	Longitude	water	foam	powder	co2	wet_chemical
1	Kia Motors - Workshop & Collision Repair Center	6.917058265	79.97257495	y	n	y	y	n
2	Tesco Office Automation (Pvt) Ltd	6.916702431	79.97298493	n	n	y	y	n
3	Punchi Car Niwasa	6.916545363	79.97238275	n	y	y	y	n
4	Pizza Hut - Kothalawala	6.916317544	79.97236758	y	y	y	y	y
5	Ky Mart	6.916011475	79.97224681	y	n	n	y	n
6	P&S (Perera and Sons) - Malabe	6.914903526	79.9720504	y	y	y	y	n
7	Mansa Fitness	6.914873697	79.97213944	y	n	n	n	n
8	Cargills Food City - Welivita	6.914704616	79.97206031	y	n	n	y	n
9	Sen-Saal Waliwita	6.914124147	79.97223734	y	y	n	n	n
10	Malabe Auto Car mart (Pvt)	6.914065567	79.97206702	n	n	n	y	y
11	SPAR Supermarket - Malabe	6.911995477	79.97228405	n	y	y	y	n
12	AutoSpa Malabe	6.911515749	79.97205599	n	y	y	y	y
13	Dinlo Lanka Pvt Ltd	6.911118909	79.97176331	y	n	y	n	n
14	Hotel Queensbury	6.918854667	79.97440902	y	n	n	y	n
15	Sugath Car Decor	6.919936875	79.97443643	n	y	y	y	n
16	Cargills Food City - Kothalawala	6.920075695	79.97413194	y	n	n	y	n
17	NIRO LANKA AUTO TRADERS	6.920947796	79.97492925	n	n	y	y	n
18	Domino's Pizza - Kaduwela	6.921115723	79.97459431	n	y	y	y	y
19	Bubble Mania - Malabe	6.921208673	79.97468686	y	y	n	y	n
20	Jetters	6.921194606	79.97525984	y	n	n	n	n
21	Sarasavi Building	6.921640501	79.97528825	y	n	n	n	n
22	Sitrek Lanka - Kaduwela	6.921596206	79.97621364	n	n	n	y	n
23	Okidmo Preschool & Daycare	6.921628671	79.97652039	y	n	y	n	n
24	Sanoora Auto Traders	6.921923103	79.97688464	n	n	n	y	n
25	Land of Kings Cafe & Restaurant	6.923685466	79.97781486	n	n	y	y	n

Implementations

Frontend of Dashboard

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

frontend > src > components > screencomponents > TestScreen > TestScreen.js > TestScreen

```
1   import React, { useState, useEffect, useRef } from "react";
2   import {FaPhone, FaMapMarkerAlt, FaCar, FaFire, FaRegCopy,} from "react-icons/fa";
3   import Swal from "sweetalert2"; // Import SweetAlert2
4   import { io } from "socket.io-client";
5   import { collection, getDocs, onSnapshot, doc, updateDoc } from "firebase/firestore"; // Import Firestore functions
6   import { db } from "../../../firebase/firebase.js";
7   import { GoogleMap, LoadScript, Marker } from '@react-google-maps/api';
8
9   import "./TestScreen.css";
10
11  // const socket = io("http://localhost:5000"); // Adjust the URL as needed
12
13  const TestScreen = () => {
14    const [selectedStation, setSelectedStation] = useState(null);
15    const [actionedStations, setActionedStations] = useState([]);
16    const [currentStations, setCurrentStations] = useState([]);
17    const [vehiclesLeft, setVehiclesLeft] = useState([]);
18    const [tab, setTab] = useState("current");
19    const [loading, setLoading] = useState(false);
20    const [cardsLoading, setCardsLoading] = useState(true);
21    const userInteractedRef = useRef(null);
22
23    const newItemRef = useRef(null);
24    const intervalRef = useRef(null);
25
26    useEffect(() => {
27      // Add an event listener to capture user interaction
28      const handleUserInteraction = () => {
29        console.log("has is done");
30        userInteractedRef.current = true;
31        // Remove the event listener after the first interaction
32        document.removeEventListener("click", handleUserInteraction);
33        document.removeEventListener("keydown", handleUserInteraction);
34        document.removeEventListener("scroll", handleUserInteraction);
35      };
36
37      // Attach the event listeners
38      document.addEventListener("click", handleUserInteraction);
39      document.addEventListener("keydown", handleUserInteraction);
40      document.addEventListener("scroll", handleUserInteraction);
41
42 >    ///====================...
105
106     return () => {
107       document.removeEventListener("click", handleUserInteraction);
108       document.removeEventListener("keydown", handleUserInteraction);
109       document.removeEventListener("scroll", handleUserInteraction);
110     };
111   }, []);
112
```

Extinguishers within 300m radius code

nearast_etg.py

```python
import pandas as pd
import numpy as np

def nearest_station(y_lat, my_lon, radius=300):

    file_path = 'extinguishers.csv'  # Your file path
    data = pd.read_csv(file_path)

    def haversine(lat1, lon1, lat2, lon2):
        R = 6371000  # Radius of the Earth in meters
        phi1 = np.radians(lat1)
        phi2 = np.radians(lat2)
        delta_phi = np.radians(lat2 - lat1)
        delta_lambda = np.radians(lon2 - lon1)
        a = np.sin(delta_phi / 2.0) ** 2 + np.cos(phi1) * np.cos(phi2) * np.sin(delta_lambda / 2.0) ** 2
        c = 2 * np.arctan2(np.sqrt(a), np.sqrt(1 - a))
        return int(R * c)


    # Calculate distance between your location and all extinguisher stations
    data['distance'] = data.apply(lambda row: haversine(y_lat, my_lon, row['Latitude'], row['Longitude']), axis=1)

    # Filter stations that are within the radius (300 meters)
    stations_within_radius = data[data['distance'] <= radius]

    if stations_within_radius.empty:
        return "No fire extinguisher stations found within 300 meters."

    # Convert the filtered DataFrame to a list of dictionaries
    nearest_stations_info = stations_within_radius.to_dict('records')

    return nearest_stations_info
```

nearast_etg_main.py

```python
from nearast_etg import nearest_station


# Example usage:
# 6.914730776020716, 79.97316762867632
my_lat = 6.914730776020716
my_lon = 79.97316762867632

#6.915624, 79.972343

# Call the function
details_array = nearest_station(my_lat, my_lon)

# Print the results
print("Nearest extinguisher Information:")
if isinstance(details_array, str):
    # If no extinguishers are found within the radius, print the message
    print(details_array)
else:
    # Iterate over the list of extinguisher information
    for extinguisher_info in details_array:
        print("Extinguisher Details:")
        for key, value in extinguisher_info.items():
            print(f"{key}: {value}")
        print()  # Add a new line between extinguishers
```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\ACER NITRO\Desktop\Station> cd Backend
PS C:\Users\ACER NITRO\Desktop\Station\Backend> python nearast_etg_main.py
Nearest extinguisher Information:
Extinguisher Details:
no: 1
name: Kia Motors - Workshop & Collision Repair Center
Latitude: 6.917058265
Longitude: 79.97257495
water: y
foam: n
powder: y
co2: y
wet_chemical: n
distance: 266

Extinguisher Details:
no: 2
name: Tesco Office Automation (Pvt) Ltd
Latitude: 6.916702431
Longitude: 79.97298493
water: n
foam: n
powder: y
co2: y
wet_chemical: n
distance: 220

Extinguisher Details:
no: 3
name: Punchi Car Niwasa
Latitude: 6.916545363
Longitude: 79.97238275
water: n
foam: y
powder: y
co2: y
wet_chemical: n
distance: 219

Extinguisher Details:
no: 4
name: Pizza Hut - Kothalawala
Latitude: 6.916317544
Longitude: 79.97236758
water: y
foam: y
powder: y
co2: y
wet_chemical: y
distance: 197
```

Nearest fire sub-station Code

GPS_calculations.py

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

Backend > GPS_calculations.py > ...

```python
import networkx as nx
import pandas as pd
import googlemaps
from datetime import datetime
# import webbrowser  # To open the map in a browser

# csv_path = 'GPS_calculations_Stations_Coordinates.csv'
# api_key = 'AIzaSyA_ZSQQ7qESG6TPvKviBf0XUI1IcGd84I4'


def get_current_datetime():
    return datetime.now().strftime("%Y-%m-%d %H:%M:%S")


def fetch_road_data_from_google(start_coords, end_coords, gmaps):
    """
    Use Google Maps API to fetch road segment data, including distance, speed, and traffic conditions.
    """
    # traffic_model='best_guess' to get real-time traffic data
    road_info = gmaps.directions(start_coords, end_coords, mode="driving", departure_time="now", traffic_model='best_guess')
    if road_info:
        leg = road_info[0]['legs'][0]
        distance = leg['distance']['value']  # in meters
        duration = leg['duration_in_traffic']['value']  # in seconds
        # Get Google Maps directions URL
        directions_url = f"https://www.google.com/maps/dir/?api=1&origin={start_coords[0]},{start_coords[1]}&destination={end_coords[0]},{end_coords[1]}&travelmode=driving"
        return distance, duration, directions_url
    return None, None, None


def create_graph_from_google_data(stations_df, lat, lon, gmaps):
    """
    Create a directed graph using the station coordinates and Google Maps road data.
    """
    G_combined = nx.DiGraph()  # Graph for combined weight

    # Add the incident node to the graph
    G_combined.add_node('incident')  # This ensures the 'incident' node exists in the graph

    directions_urls = {}  # Dictionary to store directions URLs

    # Loop through each fire station in the CSV and add nodes/edges
    for _, station in stations_df.iterrows():
        station_coords = (station['latitude'], station['longitude'])
        distance, duration, directions_url = fetch_road_data_from_google(station_coords, (lat, lon), gmaps)  # Station to incident
        if distance is not None and duration is not None:
            print(f"Station: {station['station_name']}, Distance: {distance}m, Duration: {duration}s")
```

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

Backend > GPS_calculations.py > ...

```python
def create_graph_from_google_data(stations_df, lat, lon, gmaps):
            # Combined weight prioritizing distance and factoring in duration
            combined_weight = (0.7 * distance) + (0.3 * duration)  # 70% is the weight for distance and 30% is the weight for duration.
            print(f"Combined Weight for {station['station_name']}: {combined_weight}")

            G_combined.add_edge('incident', station['station_name'], weight=combined_weight, distance=distance, time=duration)

            # Store the Google Maps directions URL for the station
            directions_urls[station['station_name']] = directions_url

    return G_combined, directions_urls


def find_optimal_route(lat, lon):
    api_key = 'AIzaSyA_ZSQQ7qESG6TPvKviBf0XUI1IcGd84I4'
    gmaps = googlemaps.Client(key=api_key)

    # Read the CSV file with fire station coordinates
    csv_path = 'GPS_calculations_Stations_Coordinates.csv'
    stations_df = pd.read_csv(csv_path)

    # Create a graph representing the road network with combined distance and duration weight
    G_combined, directions_urls = create_graph_from_google_data(stations_df, lat, lon, gmaps)

    try:
        # Find all paths and their combined weights
        shortest_path_by_combined = nx.single_source_dijkstra_path(G_combined, 'incident', weight='weight')

        # Print all stations with their combined weights (for debugging)
        print("\nShortest path by combined weight (in order):")
        station_weights = []
        for station_name in shortest_path_by_combined:
            if station_name != 'incident':  # Skip the 'incident' node itself
                combined_weight = G_combined['incident'][station_name]['weight']
                distance = G_combined['incident'][station_name]['distance']
                duration = G_combined['incident'][station_name]['time']
                station_weights.append((station_name, combined_weight, distance, duration))
                print(f"Station: {station_name}, Combined Weight: {combined_weight}")
```

```python

    # Create a graph representing the road network with combined distance and duration weight
    G_combined, directions_urls = create_graph_from_google_data(stations_df, lat, lon, gmaps)

    try:
        # Find all paths and their combined weights
        shortest_path_by_combined = nx.single_source_dijkstra_path(G_combined, 'incident', weight='weight')

        # Print all stations with their combined weights (for debugging)
        print("\nShortest path by combined weight (in order):")
        station_weights = []
        for station_name in shortest_path_by_combined:
            if station_name != 'incident':  # Skip the 'incident' node itself
                combined_weight = G_combined['incident'][station_name]['weight']
                distance = G_combined['incident'][station_name]['distance']
                duration = G_combined['incident'][station_name]['time']
                station_weights.append((station_name, combined_weight, distance, duration))
                print(f"Station: {station_name}, Combined Weight: {combined_weight}")

        # Find the station with the minimum combined weight
        nearest_station_name, min_weight, nearest_distance, nearest_duration = min(station_weights, key=lambda x: x[1])

        # Get the station data from the CSV file
        nearest_station_data = stations_df[stations_df['station_name'] == nearest_station_name].iloc[0]

        # Fetch the Google Maps directions URL for the nearest station
        directions_url = directions_urls[nearest_station_name]

        # Build the nearest station information
        nearest_station_info = {
            "Station_Name": nearest_station_data['station_name'],
            # "Combined Weight": min_weight,  # Use the minimum weight
            "Distance": nearest_distance,  # Include the distance in meters
            "Travel_Time": nearest_duration,  # Include the time in seconds
            "Address": nearest_station_data['address'],
            "Telephone": nearest_station_data['telephone'],
            # "Google Maps Route": directions_url,  # Add the route URL
            "Current_DateTime": get_current_datetime()
        }
        # Open the Google Maps route in a web browser for visualization
        # webbrowser.open(directions_url)

        return nearest_station_info

    except nx.NetworkXNoPath:
        return " ~ No valid path found to any fire station ~ "

```

GPS_calculations_main.py

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

Backend > GPS_calculations_main.py > ...

```python
from GPS_calculations import find_optimal_route

# Test usage
# 6.914730776020716, 79.97316762867632
lat = 6.914730776020716  # Test latitude
lon = 79.97316762867632  # Test longitude

# Call the function to get the nearest station info
nearest_station_info = find_optimal_route(lat, lon)

# Print the nearest station info
print("\nNearest Station Info:", nearest_station_info)

```

Output

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS

PS C:\Users\ACER NITRO\Desktop\Station\Backend> python GPS_calculations_main.py
Station: Head Quarters - Maradana, Distance: 14596m, Duration: 1610s
Combined Weight for Head Quarters - Maradana: 10700.199999999999
Station: Sub Station 01 - Hettiyawaththa, Distance: 15577m, Duration: 1869s
Combined Weight for Sub Station 01 - Hettiyawaththa: 11464.6
Station: Sub Station 02 - Gaspaha, Distance: 17300m, Duration: 1911s
Combined Weight for Sub Station 02 - Gaspaha: 12683.3
Station: Sub Station 03 - Wellawaththa, Distance: 17391m, Duration: 2005s
Combined Weight for Sub Station 03 - Wellawaththa: 12775.199999999999
Station: Sub Station 04 - Pettah, Distance: 16610m, Duration: 1876s
Combined Weight for Sub Station 04 - Pettah: 12189.8
Station: Sub Station 05 - Parliament, Distance: 10505m, Duration: 1119s
Combined Weight for Sub Station 05 - Parliament: 7689.199999999999

Shortest path by combined weight (in order):
Station: Head Quarters - Maradana, Combined Weight: 10700.199999999999
Station: Sub Station 01 - Hettiyawaththa, Combined Weight: 11464.6
Station: Sub Station 02 - Gaspaha, Combined Weight: 12683.3
Station: Sub Station 03 - Wellawaththa, Combined Weight: 12775.199999999999
Station: Sub Station 04 - Pettah, Combined Weight: 12189.8
Station: Sub Station 05 - Parliament, Combined Weight: 7689.199999999999

Nearest Station Info: {'Station_Name': 'Sub Station 05 - Parliament', 'Distance': 10505, 'Travel_Time': 1119, 'Address': 'Parliament Member Housing Complex Sri Jayawardanapura Kotte', 'Telephone': '011 2778497', 'Current_DateTime': '2024-09-11 23:09:45
'}
PS C:\Users\ACER NITRO\Desktop\Station\Backend>
```

Server-side code

server.py

```
current.json   station_main.py   server.py ×   GPS_calculations_main.py   GPS_calculations.py   nearast_etg_main.py   nearast_etg.py   TestScreen.js
Backend > server.py > real_station
from datetime import datetime, date
import os
import json
import time
import random
import firebase_admin
from flask import Flask, jsonify
from flask_cors import CORS
from flask_socketio import SocketIO, emit
from threading import Thread
from firebase_admin import credentials, firestore
from station_main import get_signals_from_network  # from station_main file


app = Flask(__name__)
cors = CORS(app)
app.config['CORS_HEADERS'] = 'Content-Type'
app.config['SECRET_KEY'] = 'your-secret-key'
socketio = SocketIO(app, cors_allowed_origins="*")
PORT = 5000


# File paths
current_file = 'current.json'
history_file = 'history.json'
vehicles_file = 'vehiclesleft.json'

# Initialize Firebase app
cred = credentials.Certificate("C:\\Users\\ACER NITRO\\Desktop\\Station\\Backend\\firebase\\firebase-adminsdk.json")
firebase_admin.initialize_app(cred)

db = firestore.client()


def initialize_file(file_path, initial_data):
    """Ensure a JSON file exists, creating it with initial data if not present."""
    if not os.path.isfile(file_path):
        with open(file_path, 'w') as f:
            json.dump(initial_data, f)

# Initialize JSON files with empty lists
initialize_file(current_file, [])
initialize_file(history_file, [])
initialize_file(vehicles_file, [])

def generate_random_station():
    """Generate a random station data."""
    station_names = ["Kottawa", "Maharagama", "Nugegoda", "Piliyandala", "Boralesgamuwa"]
    fire_types = ["normal", "chemical", "electrical", "forest", "vehicle"]

    station = {
        "Address": "gagan",
        "Current DateTime": datetime.now().strftime("%Y-%m-%d %H:%M:%S"),
        "Distance": f"{random.uniform(5, 20):.1f} km",
        "Station Name": random.choice(station_names),
```

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

Backend > server.py > real_station

```python
def send_new_station_periodically():
    """Emit a new station object every 5 seconds."""
    while True:

        car_data = get_signals_from_network()   # getting data from station_main file

        if car_data != {}:

            print("fronttt - ",car_data)
            # new_station = generate_random_station()
            new_station = real_station(car_data)

            # Update Firebase 'current' collection
            db.collection('current').add(new_station)
            print("Firebase: Updated current data")

            socketio.emit('update_new_current', new_station) #, broadcast=True


            # Add to current.json
            with open(current_file, 'r') as f:
                current_data = json.load(f)
            current_data.append(new_station)
            with open(current_file, 'w') as f:
                json.dump(current_data, f)

            # Send data to all connected clients
            # emit('update_current', current_data)


        # time.sleep(30)

@app.route('/data', methods=['GET'])
def get_data():
    # Load and return data from JSON files
    with open(current_file, 'r') as f:
        current_data = json.load(f)
    with open(history_file, 'r') as f:
        history_data = json.load(f)
    with open(vehicles_file, 'r') as f:
        vehicles_data = json.load(f)

    return jsonify({
        'current': current_data,
        'history': history_data,
        'vehicles': vehicles_data
    })

@socketio.on('send_station')
def handle_station(data):
    # Add to current.json
    with open(current_file, 'r') as f:
        current_data = json.load(f)
    current_data.append(data)
```

current.json | station_main.py | server.py | GPS_calculations_main.py | GPS_calculations.py | nearast_etg_main.py | nearast_etg.py | TestScreen.js

Backend > server.py > real_station

```python
def handle_move_to_history(data):
    with open(history_file, 'r') as f:
        history_data = json.load(f)

    current_data = [item for item in current_data if item['id'] != data['id']]
    history_data.append(data)

    with open(current_file, 'w') as f:
        json.dump(current_data, f)
    with open(history_file, 'w') as f:
        json.dump(history_data, f)

    # Send updated data to all connected clients
    emit('update_current', current_data, broadcast=True)
    emit('update_history', history_data, broadcast=True)

@socketio.on('move_to_current')
def handle_move_to_current(data):
    # Remove from history.json and add to current.json
    with open(current_file, 'r') as f:
        current_data = json.load(f)
    with open(history_file, 'r') as f:
        history_data = json.load(f)

    history_data = [item for item in history_data if item['id'] != data['id']]
    current_data.append(data)

    with open(current_file, 'w') as f:
        json.dump(current_data, f)
    with open(history_file, 'w') as f:
        json.dump(history_data, f)

    # Send updated data to all connected clients
    emit('update_current', current_data, broadcast=True)
    emit('update_history', history_data, broadcast=True)

@socketio.on('update_vehicles')
def handle_update_vehicles(data):
    # Update vehiclesleft.json
    with open(vehicles_file, 'w') as f:
        json.dump(data, f)

    # Send updated vehicles data to all connected clients
    emit('update_vehicles', data, broadcast=True)

if __name__ == '__main__':
    # Start the background thread that sends new stations periodically
    thread = Thread(target=send_new_station_periodically)
    thread.daemon = True
    thread.start()

    socketio.run(app, host='0.0.0.0', port=PORT, debug=True)
```

current.json

current.json ✕ station_main.py server.py GPS_calculations_main.py GPS_calculations.py nearast_etg_main.py nearast_etg.py TestScreen.js

Backend > current.json > ...


```
"T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:47:41", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "350", "fire_type": "mid", "vehicle_lat": "6.
926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 1902, "address": "T.B. Jaya Mawatha, Colombo 10", "date":
"2024-09-08 21:47:53", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "349", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117",
"vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 6124, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:48:04", "distance":
"2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "352", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.
com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 9291, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:48:15", "distance": "2185", "station_name": "Head Quarters -
Maradana", "telephone": "011-4222222", "travel_time": "353", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117",
"vehicle_number": "DCF-5526", "checked": 0}, {"id": 5795, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:48:26", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone":
"011-4222222", "travel_time": "353", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526",
"checked": 0}, {"id": 5164, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:48:38", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "352",
"fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 1227, "address":
"T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:48:48", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "349", "fire_type": "normal", "vehicle_lat": "6.
926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 7002, "address": "T.B. Jaya Mawatha, Colombo 10", "date":
"2024-09-08 21:48:59", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "346", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117",
"vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 8419, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:49:10", "distance":
"2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "346", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.
com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 4347, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:49:22", "distance": "2185", "station_name": "Head Quarters -
Maradana", "telephone": "011-4222222", "travel_time": "346", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117",
"vehicle_number": "DCF-5526", "checked": 0}, {"id": 2503, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:49:32", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone":
"011-4222222", "travel_time": "346", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526",
"checked": 0}, {"id": 8191, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:49:43", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "344",
"fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 8877, "address":
"T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:49:54", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "344", "fire_type": "normal", "vehicle_lat": "6.
926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 5654, "address": "T.B. Jaya Mawatha, Colombo 10", "date":
"2024-09-08 21:50:05", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "343", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117",
"vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 9417, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 21:50:17", "distance":
"2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "343", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.
com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 3423, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:00:12", "distance": "2185", "station_name": "Head Quarters -
Maradana", "telephone": "011-4222222", "travel_time": "320", "fire_type": "front", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117",
"vehicle_number": "DCF-5526", "checked": 0}, {"id": 6445, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:01:04", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone":
"011-4222222", "travel_time": "323", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526",
"checked": 0}, {"id": 9555, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:01:54", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "322",
"fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 4404, "address":
"T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:02:45", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "317", "fire_type": "normal", "vehicle_lat": "6.
926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 1179, "address": "T.B. Jaya Mawatha, Colombo 10", "date":
"2024-09-08 22:03:36", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "314", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117",
"vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 4833, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:04:27", "distance":
"2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "312", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.
com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 4413, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:12:19", "distance": "2185", "station_name": "Head Quarters -
Maradana", "telephone": "011-4222222", "travel_time": "324", "fire_type": "back", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117",
"vehicle_number": "DCF-5526", "checked": 0}, {"id": 6013, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:13:10", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone":
"011-4222222", "travel_time": "335", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526",
"checked": 0}, {"id": 1963, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:15:29", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "325",
"fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 5727, "address":
"T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:16:21", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "320", "fire_type": "normal", "vehicle_lat": "6.
926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 2002, "address": "T.B. Jaya Mawatha, Colombo 10", "date":
"2024-09-08 22:17:12", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "329", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117",
"vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 2251, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:40:57", "distance":
"2185", "station_name": "Head Quarters - Maradana", "telephone": "011-4222222", "travel_time": "329", "fire_type": "back", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.
com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526", "checked": 0}, {"id": 9708, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:41:47", "distance": "2185", "station_name": "Head Quarters -
Maradana", "telephone": "011-4222222", "travel_time": "325", "fire_type": "normal", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117",
"vehicle_number": "DCF-5526", "checked": 0}, {"id": 2182, "address": "T.B. Jaya Mawatha, Colombo 10", "date": "2024-09-08 22:47:04", "distance": "2185", "station_name": "Head Quarters - Maradana", "telephone":
"011-4222222", "travel_time": "326", "fire_type": "mid", "vehicle_lat": "6.926878", "vehicle_lon": "79.850117", "vehicle_location": "https://www.google.com/maps?q=6.926878,79.850117", "vehicle_number": "DCF-5526",
"checked": 0}]
```


station_main.py

current.json station_main.py ✕ server.py GPS_calculations_main.py GPS_calculations.py nearast_etg_main.py nearast_etg.py TestScreen.js

Backend > station_main.py > ...

```python
from Mqtt_engine import mqtt_clint
from GPS_calculations import find_optimal_route
import threading


car_clnt = mqtt_clint(···

data_baffer = {}


def watching_signals():

    global data_baffer

    while True:
        msg, topic = car_clnt.subscribe_last_buffer()

        if msg != None and topic == 'f_station':

            data = msg.split(',')

            try:
                your_lat = float(data[0])
                your_lon = float(data[1])
                # Print the values of latitude and longitude for debugging
                print(f"Latitude: {your_lat}, Longitude: {your_lon}")
                vehicle_number = data[2]
                fire_type = data[3]

                nearest_station_info = find_optimal_route(your_lat, your_lon)
                data_baffer = nearest_station_info
                data_baffer['vehicle_number'] = vehicle_number
                data_baffer['fire_type'] = fire_type
                data_baffer['vehicle_lat'] = your_lat
                data_baffer['vehicle_lon'] = your_lon

                def create_google_maps_link(latitude, longitude):
                    base_url = "https://www.google.com/maps?q="
                    return f"{base_url}{latitude},{longitude}"

                link = create_google_maps_link(your_lat, your_lon)
                data_baffer['vehicle_location'] = link

                # Debugging Output
                print("Updated Data Buffer: ", data_baffer)

            except ValueError as e:
                print(f"Error converting latitude/longitude to float: {e}")
            except Exception as e:
                print(f"Error processing incoming data: {e}")


thread1 = threading.Thread(target=watching_signals)
thread1.start()
```

Work Breakdown