FlarePath

Advanced Vehicle Fire Safety and Monitoring with Rapid Emergency Dispatch Solutions

R24-058

SLIIT
FACULTY OF COMPUTING

Some vehicle fire incidents in Sri Lanka

NEWSWIRE

HOME NEWS WORLD BIZWIRE SPORTS CRICWIRE YAMU EDUWIRE SRILANKANIZATION

Vehicle catches fire on Dehiwala bridge last night (video)

April 2, 2024 at 9:58 AM

NEWSWIRE

A car had caught fire while on the Dehiwala flyover on Monday night (Apr 01), it was reported.

According to reports, it is suspected that the car had caught fire due to a technical fault in the vehicle.

The incident occurred when the vehicle was plying from Colombo to Dehiwala around 7.45 pm.

ADA derana

SRI LANKA'S #1 NEWS SOURCE

adaderana.lk

Hot News Biz Sports Entertainment Tech Videos Archive Contact us At

State Minister Chamara Sampath's vehicle catches fire in Halpe

April 16, 2024 08:55 am

Like 413K people like this. Be the first of your friends.

State Minister Chamara Sampath Dasanayake's vehicle has suddenly caught fire at around 12.45 a.m. today (16), Ada Derana learns.

The incident took place in the Halpe area of Bandarawela when the Dasanayake had been travelling from Mahiyanganaya to Ella.

However, the state minister and his driver have managed to escape the fire unharmed.

The fire trucks of Bandarawela Municipal Council, Ella Police and the residents of the area had tackled the flames

Daily News

Local Politics Entertainment Sports Business Feature Events Law & Order Editorial

Monday, September 9, 2024 BREAKING NEWS US confirms first human bird flu case with no known animal exposure

Home » Car catches fire near Royal

LOCAL

Car catches fire near Royal

August 5, 2023 1:12 am • 0 comment

Popular Posts

1 Obituaries

Obituaries

September 4, 2024

Hiru News

Home Local World Entertainment Business Sports Contact Us Subscribe Watch Hiru TV

සිංහල தமிழ் Hiru Gossip

LATEST NEWS tal dues Youth missing after being swept away by waves in Panadura Sri Lanka to chase 219 after England were bundled out for their lowest

Van, cab, and three-wheeler destroyed by fire in Padukka

Saturday, 07 September 2024 - 11:29

The Padukka Police reported that a van, a cab, and a three-wheeler parked at a house in Mahingala, Padukka, were suddenly destroyed by fire.

ENHANCED BY Google

Trending News

Gone in 15 seconds: Louisiana skyscraper implodes

08 September 2024

Cess levy on imported cement reduced

08 September 2024

Israeli attacks kill over a 60 people

08 September 2024

NEWSWIRE

HOME NEWS WORLD BIZWIRE SPORTS CRICWIRE YAMU EDUWIRE SRILANKANIZATION

Watch : Vehicle catches fire near Colombo Lakehouse Roundabout

July 31, 2023 at 4:43 PM

NEWSWIRE

A vehicle caught fire near the Lakehouse Roundabout in Colombo today, according to video footage shared by eyewitness.

The Police stated that no injuries or casualties were reported in the incident.

Video footage shows officials from the Sri Lanka Navy and Colombo fire brigade working together to douse the

The Police added that the fire has been doused and vehicular movement has returned to the road.

Our Team

MR.NELUM CHATHURANGA AMARASENA
SUPERVISOR

MR.DEEMANTHA NAYANJITH SIRIWARDANA
CO-SUPERVISOR

MR. W. D. ONRAY SAHINDA
EXTERNAL-SUPERVISOR

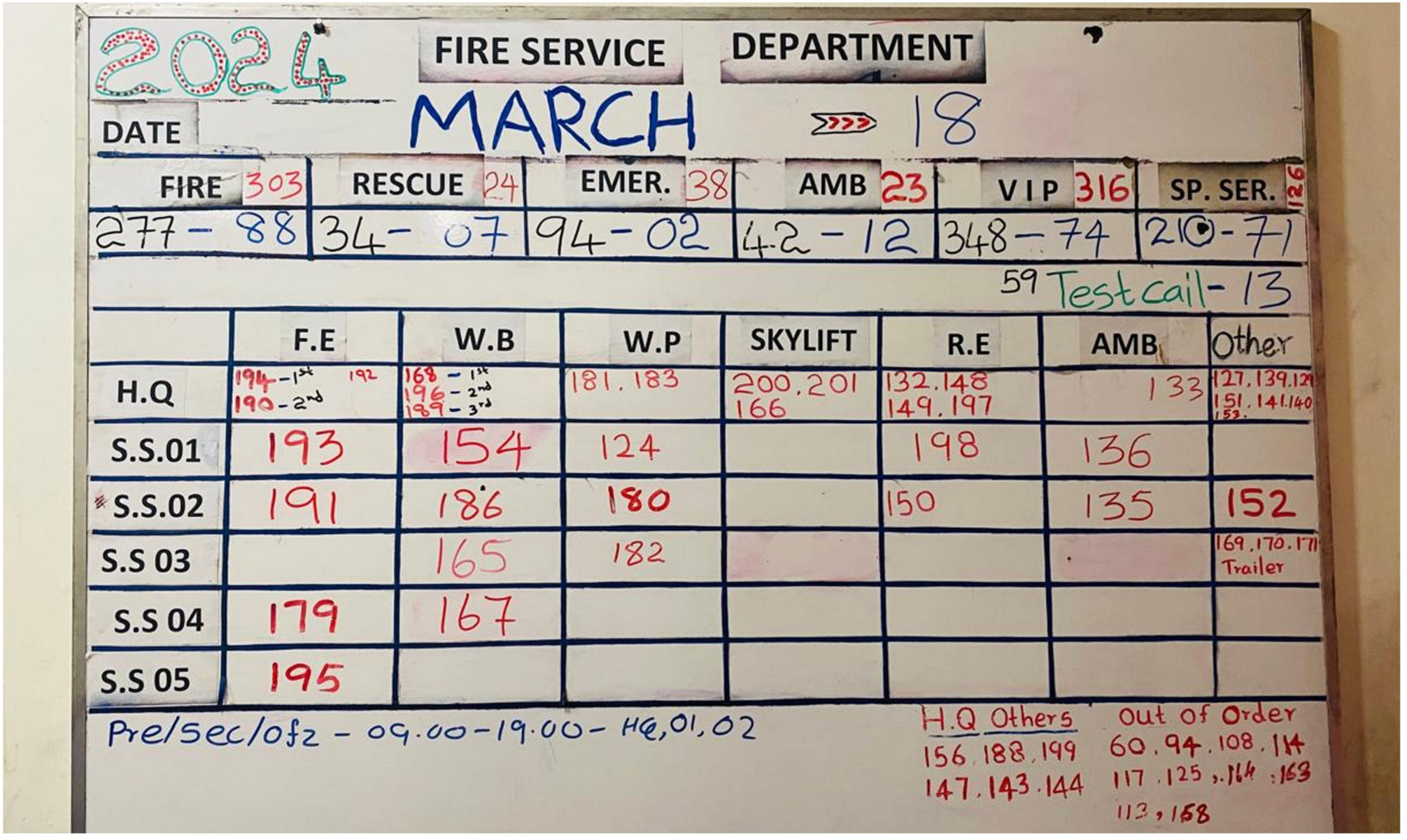
PERAMUNAGE A.N
IT21080562

ABEYWARDHANA D.N
IT21133718

DHARMAGUNAWARDANA W.M.P.I
IT21132346

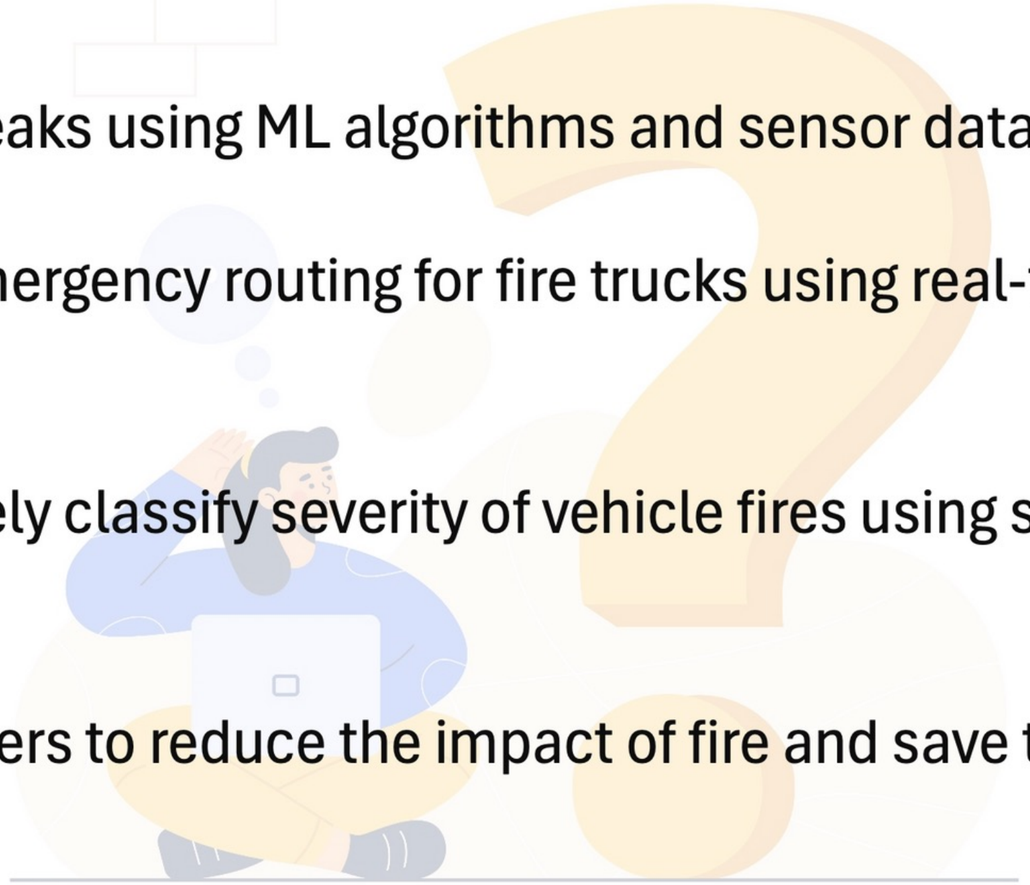
ANTHICK G.N
IT21096266

SLIIT
FACULTY OF COMPUTING

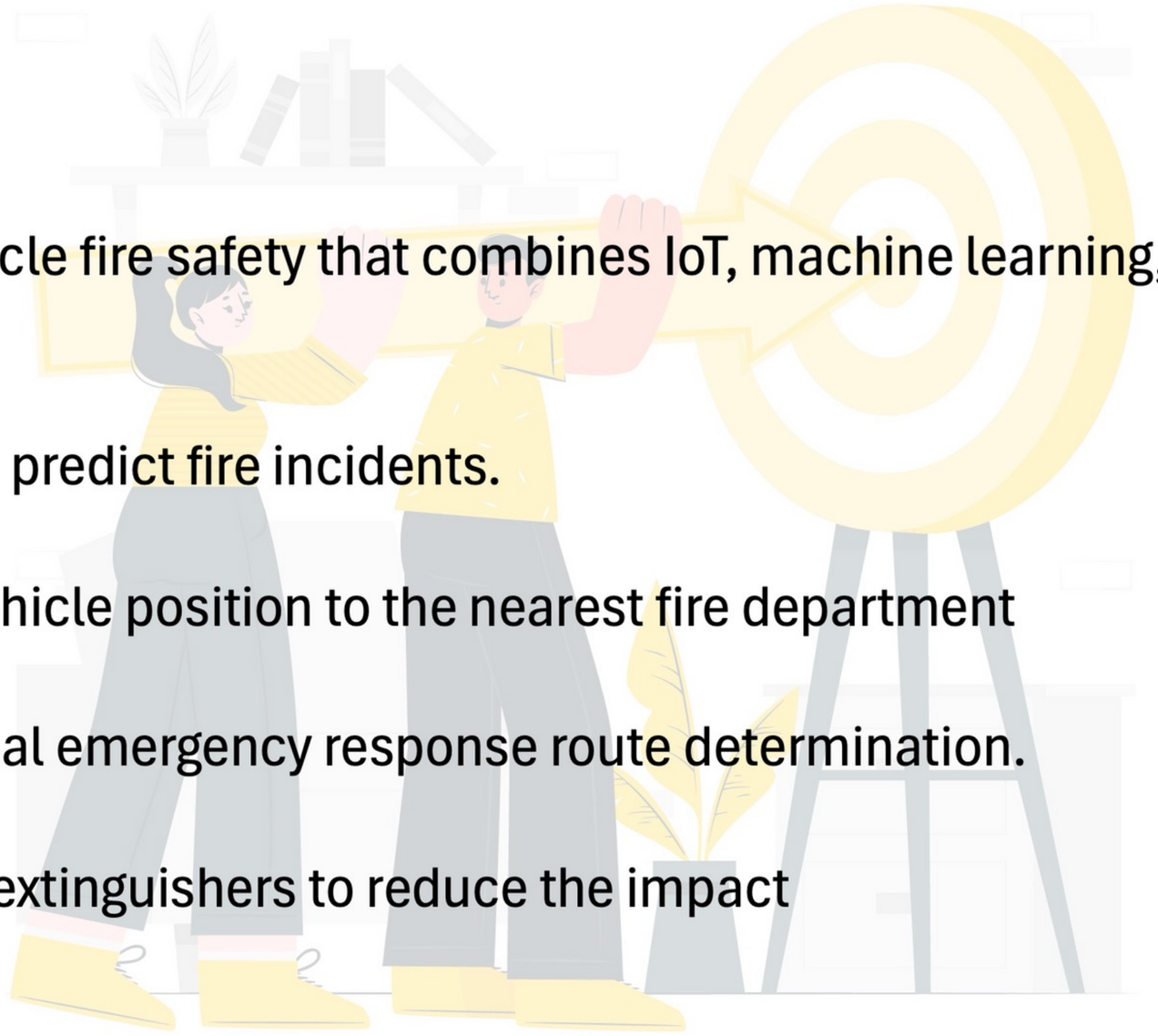

2024
FIRE SERVICE DEPARTMENT
DATE
MARCH
18
FIRE 303
RESCUE 24
EMER. 38
AMB 23
VIP 316
SP. SER. 126
277 - 88
34 - 07
94 - 02
42 - 12
348 - 74
210 - 71
59 Test call - 13
F.E
W.B
W.P
SKYLIFT
R.E
AMB
Other
H.Q
194 - 1st 192
190 - 2nd
168 - 1st
196 - 2nd
189 - 3rd
181, 183
200, 201
166
132, 148
149, 197
133
127, 139, 129
151, 141, 140
153
S.S.01
193
154
124
198
136
S.S.02
191
186
180
150
135
152
S.S 03
165
182
169, 170, 171
Trailer
S.S 04
179
167
S.S 05
195
Pre/Sec/Ofz - 09.00 - 19.00 - HQ, 01, 02
H.Q Others
156, 188, 199
147, 143, 144
Out of Order
60, 94, 108, 114
117, 125, 164, 163
113, 158

Research Problem

- How to accurately predict vehicle fire outbreaks using ML algorithms and sensor data?
- How can cloud-based analytics optimize emergency routing for fire trucks using real-time and historical traffic data?
- What methods can be employed to accurately classify severity of vehicle fires using sensor data?
- How to implement automatic fire extinguishers to reduce the impact of fire and save the vehicle?

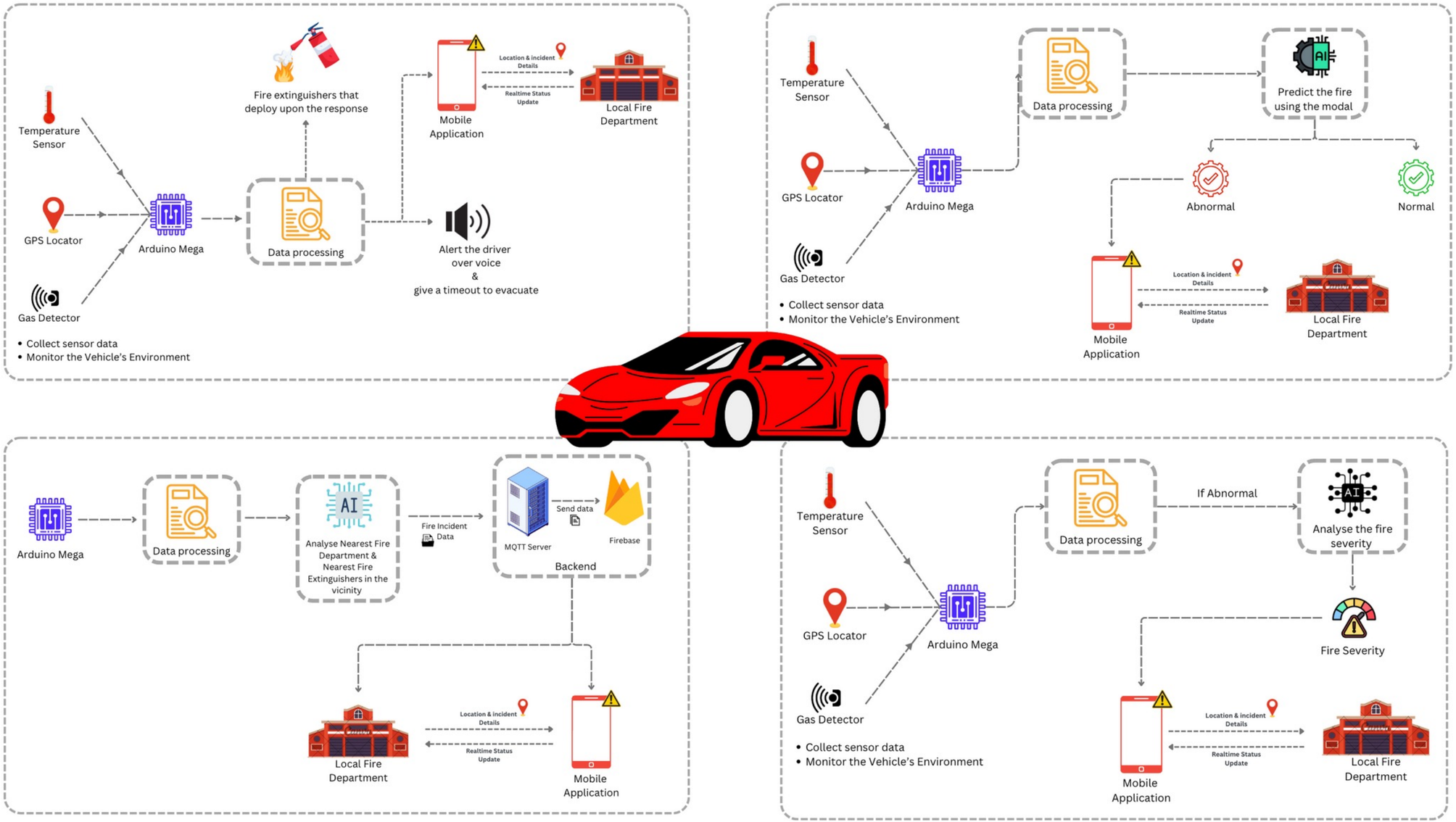


Objectives



- Create an integrated solution for vehicle fire safety that combines IoT, machine learning, and cloud computing.
- Utilize ML to analyze sensor data and predict fire incidents.
- Instantly relay fire information and vehicle position to the nearest fire department
- Monitor vehicle parameters for optimal emergency response route determination.
- Implement IOT based automatic fire extinguishers to reduce the impact

System Overview



Individual Components

Peramunage A.N | IT21080562
Specializing in Information Technology

Vehicle Safety System for Fire Detection and Prevention

Research Problem

Limited proactive fire detection

Current vehicle fire detection systems are reactive, lacking the ability to detect potential fire hazards before they escalate.

Inadequate automotive fire detection technologies

Existing systems are not designed for the automotive environment, leading to suboptimal performance in detecting and preventing vehicle fires.

Inadequate automotive fire detection technologies

The lack of advanced detection mechanisms in vehicles results in delayed response times, reducing the effectiveness of fire prevention measures.

Research Gap

- Research needed for improved early detection and accurate prediction of normal or abnormal events.
- Seamless connection of detection systems with vehicle electronics crucial for swift prediction and response.
- Developing algorithms to accurately classify events as normal or abnormal, facilitating prompt action and mitigating risks effectively.
- Developing streamlined methods for sensor data collection in vehicles to ensure data quality and minimize latency, thereby improving the accuracy and timeliness of fire prediction models.

Objectives

Design and implement a novel fire detection solution for vehicles by installing sensors to detect temperature

- Develop ML algorithms for analyzing sensor data and predicting potential fire situations.
- Establish a real-time data processing system for immediate response to fire threats.
- Create an alert mechanism based on continuous sensor monitoring and integration for accurate detection.
- ML for predictive analysis and continuous learning to enhance detection accuracy.
- Provide customized alerts to improve driver safety and response time.

Completion Of The Component

Progress at PP1 – 50%

- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Gathered normal data
- Created Datasets
- Data pre-processing
- Model Selection
- Model trained
- Developed the highest accuracy model
- Design System flow chart
- Design mobile app wireframe
- Design mobile app UIs
- Learning about IOT as a beginner
- Studying IOT Tutorials
- Studying IOT sensors

Completion Of The Component

Progress at PP2 – 90%

- Design mobile app frontend
- Design mobile app backend
- Implementing iot device
- Regather data using iot device
- Gather abnormal data
- Integrated with other components
- integrated fully IOT device
- integrated mobile app
- Trained the model
- Test the model
- Integrated fully system
- Run a test
- UI/UX Optimization for mobile app

Progress at Final – 10%

- System Testing

Gathered Data

Axio 2015

Back			Front			Mid			Back			Front			Mid		
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3
26.87	26.87	35.95	29.37	29.5	29.25	29.97	29.45	34.91	25.15	29.13	28.99	46.56	37.4	39.76	25.62	29.67	29.15
31.05	27.5	36.84	29.37	29.25	35.62	30.31	30.84	36.23	27.5	29.13	25.35	43.43	42.54	44.16	26.28	26.32	28.92
32.36	36.34	26.39	32.47	32.48	29.37	38.37	33.6	37.65	25.35	26.45	28.45	39.88	44.14	48.39	28.2	28.06	25.21
38.53	29.08	36.84	30.34	29.84	29.08	29.97	29.45	34.91	27.84	27.68	25.14	43.51	47.77	43.32	28.91	28.95	30
28.59	27.4	34.14	32.57	36.39	35.02	33.14	26.2	38.99	29.22	28.79	25.32	38.5	44.25	43.55	29.96	28.4	29.44
28.83	35.63	30.6	34.17	34.78	31.92	38.98	26.15	31.54	25.33	26.58	26.72	40.95	49.72	38.21	29.26	31.45	30.68
26.84	27.37	27.18	34.39	33.3	33.68	38.98	39.17	32.16	25	26.58	25.91	49.34	39.68	39.91	25.81	30.87	31.31
29.93	36.52	33.11	35.19	34.99	29.69	33.93	39.92	32.68	28.73	26.76	25.36	40.54	43.85	41.86	27.64	26.54	28.53
29.71	38.2	35.67	33.29	34.99	36.8	37.03	26.11	32.71	27.74	27.63	27.34	40.96	44.01	44.81	25.94	25.12	31.38
27.85	29.14	39.84	31.11	30.05	30.17	37.67	32.32	31.85	27.51	27.35	28.6	42.81	39.21	49.87	28.37	29.87	30.49
31.39	36.31	37.57	36.8	36.59	34.14	32.63	30.97	29.26	29.64	25.13	26.48	45.8	38.39	44.09	26.43	28.55	26.67
31.37	35.14	33.07	34.48	33.96	32.65	32.93	36.87	36.54	27.01	26.63	28.79	44.75	40.68	38.8	29.31	26.15	25.86
32.6	32.43	33.85	34.62	29.25	30.94	35.94	35.07	39.03	29.95	25.8	26.21	44.81	39.73	38.29	30.64	31	27.29
35.98	33.1	26.86	33.97	30.39	33.55	37.35	33.3	36.26	29.03	29.57	25.26	44.2	38.57	47.74	25.26	25.52	30.89
34.17	39.58	28.46	33.88	36.04	35.77	28.71	27.28	37.88	28.53	26.13	29.49	41.54	38.08	42.36	25.14	31.9	27.8
31.8	37.49	34.91	36.51	31.14	29.06	35.36	34.87	33.23	26.42	27.82	26.9	43.3	48.17	43.43	28.32	28.86	31.37
38.04	32.59	37.42	36.58	31.58	31.72	39.94	29.75	31.71	25.55	29.62	27.81	41.13	45.28	47.59	26.13	27.22	30.15
35.1	33.19	38.1	31.09	32.73	35.68	28.73	29.77	35.17	29.5	26.27	25.17	42.89	38.82	40.5	26.17	29.96	27.81
26.65	29.32	33.06	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.74	28.24	45.14	43.47	39.3	26.79	26.25	29.59
39.57	30.56	28.21	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	43.16	42.77	42.63	27.07	29.62	27.89
34.83	35.37	34	29.35	31.4	31.14	29.58	29.61	39.13	27.83	27.1	27.92	42.39	37.31	47.42	27.25	28.93	26.82
27.78	36.27	27.31	33.88	32.85	36.39	29.35	35.93	29.71	29.62	28.63	25.62	41.14	47.21	45.03	27.15	31.8	25.84
34.88	31.69	35.69	30.32	30.51	34.86	30.11	32.58	37.86	29.38	27.49	27.88	47.83	39.09	41.16	29.36	28.64	31.87
39.15	29.77	36.96	32.41	33.21	32.14	37.79	36.71	32.36	25.96	25.1	28.23	39.44	38.78	43.13	30.41	26.18	26.28
32	35.18	33.04	32.38	31.72	34.41	35.44	28.55	30.82	27.37	27.34	28.27	48.09	48.31	46.52	26.03	30.93	31.61
26.26	32.45	36.26	35.88	35.4	34.65	38.08	34.27	37.6	29.16	28.16	26.47	41	47.19	40.62	28.99	29.63	29.26
30.13	33.09	31.52	33.2	32.55	30.11	32.27	28.24	29.12	29.39	28.86	26.71	49.47	46.44	46.5	26.55	25.06	25.86
36.81	36.68	34.35	30.48	33.48	35.29	31.23	39.26	34.24	27.85	26.99	29.56	44	45.81	47.63	30.23	25.22	28.09
39.16	35.82	36.35	31.71	32.21	34.07	31.34	29.58	30.64	25.45	27.88	29.68	42.97	38.29	40.48	29.79	31.37	31.32
37.34	33.99	31.19	29.4	33.98	33.2	31.75	33.07	34.64	29.08	27.89	26.8	42.81	45.31	49.65	25.28	28.56	27.31
32.74	27.76	33.99	30.89	36.98	34.77	33.83	38.79	38.08	27.75	28.04	26.31	42.85	44.05	40.24	26.51	25.3	31.58
32.99	32.18	38.3	32.07	34.42	35.64	28.04	33.39	38.83	28.49	26.89	28.76	45.82	39.88	39.42	25.98	27.91	28.87
30.59	30.92	29.16	35.42	30.43	30.02	34.79	33.05	39.68	25.15	28.98	26.57	43.79	38.4	47.6	25.93	27.09	25.55
34.73	26.8	28.71	36.14	33.31	33.02	38.05	35.72	39.34	28.71	27.61	26.42	49.83	40.68	38.76	31.5	25.47	29.77
28.42	33.32	30.26	31.15	36.37	31.39	35.05	29.95	30.36	25.96	25.32	26.67	49.1	43.37	39.57	29.25	27.87	27.43

Sheet1

Gathered Data

Toyota Corolla 121

Back			Front			Mid			Back			Front			Mid		
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3
27.84	26.5	33.94	31	32.84	34.59	35.71	33.19	38.63	25	26.58	25.91	38.05	37.09	38.02	25.26	25.52	30.89
26.5	27.84	26.5	31.5	36.23	35.62	30.31	30.84	36.23	25.33	26.58	26.72	38.11	37.17	38.10	25.14	31.9	27.8
31.05	27.5	36.84	32.47	32.48	33.89	38.37	33.6	37.65	25.35	26.45	28.45	38.13	37.20	38.14	28.32	28.86	31.37
32.36	36.34	26.39	30.34	29.84	29.08	29.97	29.45	34.91	27.84	27.68	25.14	38.25	37.23	38.16	26.13	27.22	30.15
34.52	39.15	27.43	32.57	36.39	35.02	33.14	26.2	38.99	29.22	28.79	25.32	38.27	37.28	38.22	26.17	29.96	27.81
30.08	35.56	28.96	34.17	34.78	31.92	38.98	26.15	31.54	25.33	26.58	26.72	38.29	37.37	38.24	26.79	26.25	29.59
32.5	29.57	32.08	34.39	33.3	33.68	38.98	39.17	32.16	25	26.58	25.91	38.32	37.40	38.29	27.07	29.62	27.89
29.15	38.39	38.46	35.19	34.99	29.69	33.93	39.92	32.68	28.73	26.76	25.36	38.43	37.49	38.31	27.25	28.93	26.82
38.53	29.08	36.84	33.29	34.99	36.8	37.03	26.11	32.71	27.74	27.63	27.34	38.60	37.50	38.38	27.15	31.8	25.84
28.59	27.4	34.14	31.11	30.05	30.17	37.67	32.32	31.85	27.51	27.35	28.6	38.67	37.53	38.39	29.36	28.64	31.87
28.83	35.63	30.6	36.8	36.59	34.14	32.63	30.97	29.26	29.64	25.13	26.48	38.78	37.57	38.54	26.43	28.55	26.67
26.84	27.37	27.18	34.48	33.96	32.65	32.93	36.87	36.54	27.01	26.63	28.79	38.80	37.61	38.66	29.31	26.15	25.86
29.93	36.52	33.11	34.62	29.25	30.94	35.94	35.07	39.03	29.95	25.8	26.21	38.83	37.62	38.70	30.64	31	27.29
29.71	38.2	35.67	33.97	30.39	33.55	37.35	33.3	36.26	29.03	29.57	25.26	38.83	37.65	38.72	25.26	25.52	30.89
27.85	29.14	39.84	33.88	36.04	35.77	28.71	27.28	37.88	28.53	26.13	29.49	38.90	37.68	38.83	25.14	31.9	27.8
31.39	36.31	37.57	36.51	31.14	29.06	35.36	34.87	33.23	26.42	27.82	26.9	38.96	37.72	38.89	27.15	31.8	25.84
31.37	35.14	33.07	36.58	31.58	31.72	39.94	29.75	31.71	25.55	29.62	27.81	38.95	37.75	38.93	29.36	28.64	31.87
32.6	32.43	33.85	31.09	32.73	35.68	28.73	29.77	35.17	29.5	26.27	25.17	38.98	37.76	38.95	30.41	26.18	26.28
35.98	33.1	26.86	33.92	31.22	33.46	37.01	26.37	30.22	25.51	27.74	28.24	39.01	37.79	38.96	26.79	26.25	29.59
34.17	39.58	28.46	36.78	30.57	32.71	37.98	32.4	39.15	25.83	27.25	26.85	39.04	37.83	39.00	27.07	29.62	27.89
31.8	37.49	34.91	29.35	31.4	31.14	29.58	29.61	39.13	27.83	27.1	27.92	39.06	38.19	39.23	27.25	28.93	26.82
38.04	32.59	37.42	33.88	32.85	36.39	29.35	35.93	29.71	29.62	28.63	25.62	39.26	38.23	39.26	27.15	31.8	25.84
35.1	33.19	38.1	30.32	30.51	34.86	30.11	32.58	37.86	29.38	27.49	27.88	39.32	38.25	39.34	29.36	28.64	31.87
26.65	29.32	33.06	32.41	33.21	32.14	37.79	36.71	32.36	25.96	25.1	28.23	39.34	38.32	39.36	30.41	26.18	26.28
39.57	30.56	28.21	32.38	31.72	34.41	35.44	28.55	30.82	27.37	27.34	28.27	39.37	38.34	39.41	26.03	30.93	31.61
34.83	35.37	34	35.88	35.4	34.65	38.08	34.27	37.6	29.16	28.16	26.47	39.43	38.44	39.43	28.99	29.63	29.26
27.78	36.27	27.31	33.2	32.55	30.11	32.27	28.24	29.12	29.39	28.86	26.71	39.50	38.56	39.49	26.55	25.06	25.86
34.88	31.69	35.69	30.48	33.48	35.29	31.23	39.26	34.24	27.85	26.99	29.56	39.52	38.58	39.53	30.23	25.22	28.09
39.15	29.77	36.96	31.71	32.21	34.07	31.34	29.58	30.64	25.45	27.88	29.68	39.60	38.61	39.59	29.79	31.37	31.32

Sheet1

Gathered Data

Vitz 2016

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Engine Off												Engine Running										
Back				Front				Mid				Back				Front				Mid		
sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3
32	35.18	33.04		33.97	30.39	33.55		29.35	35.93	29.71		25.96	25.1	28.23		41.54	38.08	42.36		25.26	25.52	30.8
26.26	32.45	36.26		33.88	36.04	35.77		30.11	32.58	37.86		27.37	27.34	28.27		43.3	48.17	43.43		25.14	31.9	27
30.13	33.09	31.52		36.51	31.14	29.06		37.79	36.71	32.36		29.16	28.16	26.47		41.13	45.28	47.59		28.32	28.86	31.1
36.81	36.68	34.35		36.58	31.58	31.72		35.44	28.55	30.82		29.39	28.86	26.71		42.89	38.82	40.5		26.13	27.22	30.
39.16	35.82	36.35		31.09	32.73	35.68		38.08	34.27	37.6		27.85	26.99	29.56		45.14	43.47	39.3		26.17	29.96	27.8
37.34	33.99	31.19		33.92	31.22	33.46		32.27	28.24	29.12		25.45	27.88	29.68		43.16	42.77	42.63		26.79	26.25	29.5
32.74	27.76	33.99		36.78	30.57	32.71		31.23	39.26	34.24		29.08	27.89	26.8		42.39	37.31	47.42		27.07	29.62	27.8
32.99	32.18	38.3		29.35	31.4	31.14		31.34	29.58	30.64		27.75	28.04	26.31		41.14	47.21	45.03		27.25	28.93	26.8
30.59	30.92	29.16		33.88	32.85	36.39		31.75	33.07	34.64		28.49	26.89	28.76		47.83	39.09	41.16		27.15	31.8	25.8
34.73	26.8	28.71		30.32	30.51	34.86		33.83	38.79	38.08		27.01	26.63	28.79		39.44	38.78	43.13		29.36	28.64	31.8
28.42	33.32	30.26		32.41	33.21	32.14		28.04	33.39	38.83		29.95	25.8	26.21		48.09	48.31	46.52		30.41	26.18	26.1
36.6	28.14	31.72		32.38	31.72	34.41		34.79	33.05	39.68		29.03	29.57	25.26		41	47.19	40.62		26.03	30.93	31.0
28.39	37.67	38.43		35.88	35.4	34.65		38.05	35.72	39.34		28.53	26.13	29.49		49.47	46.44	46.5		28.99	29.63	29.1
37.05	33.7	34.73		33.2	32.55	30.11		35.05	29.95	30.36		26.42	27.82	26.9		44.2	38.57	47.74		25.26	25.52	30.8
28.06	37.09	30.2		30.48	33.48	35.29		30.43	35.83	33.88		25.55	29.62	27.81		41.54	38.08	42.36		25.14	31.9	27
38.65	34.23	35.41		31.71	32.21	34.07		32.03	38.38	31.93		29.5	26.27	25.17		43.3	48.17	43.43		28.32	28.86	31.1
32.47	26.76	31.29		29.4	33.98	33.2		36.62	28.36	37.95		25.51	27.74	28.24		41.13	45.28	47.59		26.13	27.22	30.
27.72	35.7	29.09		30.89	36.98	34.77		31.92	35.04	33.48		29.5	26.27	25.17		42.89	38.82	40.5		26.17	29.96	27.8
30.5	32.56	28.94		32.07	34.42	35.64		30.05	37.7	35		25.51	27.74	28.24		45.14	43.47	39.3		26.79	26.25	29.5
30.36	30.33	37.3		35.42	30.43	30.02		37.86	31.17	34.21		25.83	27.25	26.85		43.16	42.77	42.63		27.07	29.62	27.8
38.97	38.57	39.34		36.14	33.31	33.02		31.58	34.39	32.56		27.83	27.1	27.92		42.39	37.31	47.42		30.41	26.18	26.1
40	33.88	39.01		31.15	36.37	31.39		32.65	36.56	28.64		29.62	28.63	25.62		41.14	47.21	45.03		26.03	30.93	31.0
39.43	31.54	31.17		34.65	33.08	35.43		39.67	31.24	35.16		29.38	27.49	27.88		47.83	39.09	41.16		28.99	29.63	29.1
26.67	34.2	28.81		32.94	35.46	34.35		37.25	32.13	29.09		25.96	25.1	28.23		39.44	38.78	43.13		26.55	25.06	25.8
39.93	34.45	26.24		29.53	32.12	32.15		29.11	38.19	33.73		27.37	27.34	28.27		48.09	48.31	46.52		26.03	30.93	31.0
33.49	34.77	32.55		33.94	31.32	31.72		36.36	33.75	30.14		29.16	28.16	26.47		41	47.19	40.62		28.99	29.63	29.1

Sheet1

Gathered Data

Wagon R 2016

Engine Off										Engine Running								
Back			Front			Mid				Back			Front			Mid		
sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3	sensor 1	sensor 2	sensor 3
28.83	35.63	30.6	31	32.48	31.5	28.71	27.28	38.63		25.15	29.13	28.99	38.06	37.10	38.02	25.62	29.67	29.15
26.84	27.37	27.18	31.5	36.23	35.62	30.31	30.84	36.23		27.5	29.13	25.35	38.09	37.17	38.09	26.28	26.32	28.92
29.93	36.52	33.11	32.47	32.48	33.89	28.73	29.77	37.65		25.35	26.45	28.45	38.13	37.18	38.15	28.2	28.06	25.21
29.71	38.2	35.67	30.34	29.84	29.08	29.97	29.45	34.91		27.84	27.68	25.14	38.25	37.22	38.17	28.91	28.95	30
27.85	29.14	39.84	30.32	30.51	34.86	33.14	26.2	38.99		25.96	25.1	28.23	38.27	37.28	38.22	29.96	28.4	29.44
31.39	36.31	37.57	32.41	33.21	32.14	28.73	29.77	31.54		25.33	26.58	26.72	38.28	37.35	38.25	29.26	31.45	30.68
31.37	35.14	33.07	32.38	31.72	34.41	38.98	39.17	32.16		25	26.58	25.91	38.32	37.40	38.28	25.81	30.87	31.31
32.6	32.43	33.85	35.88	35.4	34.65	32.93	36.87	32.68		28.73	26.76	25.36	38.44	37.49	38.30	27.64	26.54	28.53
35.98	33.1	26.86	33.2	32.55	30.11	28.73	29.77	32.71		27.74	27.63	27.34	38.61	37.50	38.37	25.94	25.12	31.38
34.17	39.58	28.46	30.48	33.48	35.29	37.35	33.3	31.85		25.96	25.1	28.23	38.66	37.54	38.40	28.37	29.87	30.49
31.8	37.49	34.91	36.8	36.59	34.14	28.71	27.28	29.26		29.64	25.13	26.48	38.79	37.59	38.53	26.43	28.55	26.67
38.04	32.59	37.42	34.48	33.96	32.65	35.36	34.87	36.54		27.01	26.63	28.79	38.79	37.60	38.67	29.31	26.15	25.86
35.1	33.19	38.1	34.62	29.25	30.94	39.94	29.75	39.03		29.95	25.8	26.21	38.83	37.61	38.69	30.64	31	27.29
26.65	29.32	33.06	33.97	30.39	33.55	28.73	29.77	36.26		29.03	29.57	25.26	38.84	37.66	38.72	25.26	25.52	30.89
39.57	30.56	28.21	33.88	36.04	35.77	37.01	26.37	37.88		28.53	26.13	29.49	38.90	37.68	38.84	25.14	31.9	27.8
34.83	35.37	34	36.51	31.14	29.06	37.98	32.4	33.23		26.42	27.82	26.9	38.95	37.72	38.90	28.32	28.86	31.37
27.78	36.27	27.31	36.58	31.58	31.72	29.58	29.61	31.71		25.55	29.62	27.81	38.97	37.75	38.93	26.13	27.22	30.15
32.6	32.43	33.85	31.09	32.73	35.68	29.35	35.93	35.17		29.5	26.27	25.17	38.98	37.77	38.94	26.17	29.96	27.81
35.98	33.1	26.86	33.92	31.22	33.46	37.01	26.37	30.22		25.51	27.74	28.24	39.03	37.78	38.95	26.79	26.25	29.59
34.17	39.58	28.46	36.78	30.57	32.71	37.98	32.4	39.15		25.83	27.25	26.85	39.03	37.84	39.01	27.07	29.62	27.89
31.8	37.49	34.91	29.35	31.4	31.14	29.58	29.61	39.13		27.83	27.1	27.92	39.06	38.18	39.23	27.25	28.93	26.82
38.04	32.59	37.42	33.88	32.85	36.39	29.35	35.93	29.71		29.62	28.63	25.62	39.26	38.22	39.26	27.15	31.8	25.84
35.1	33.19	38.1	30.32	30.51	34.86	30.11	32.58	37.86		29.38	27.49	27.88	39.30	38.25	39.35	29.36	28.64	31.87
26.65	29.32	33.06	32.41	33.21	32.14	37.79	36.71	32.36		25.96	25.1	28.23	39.34	38.33	39.37	30.41	26.18	26.28
39.57	30.56	28.21	32.38	31.72	34.41	35.44	28.55	30.82		27.37	27.34	28.27	39.36	38.36	39.41	26.03	30.93	31.61
34.83	35.37	34	35.88	35.4	34.65	38.08	34.27	37.6		29.16	28.16	26.47	39.43	38.43	39.42	28.99	29.63	29.26

Gathered Data

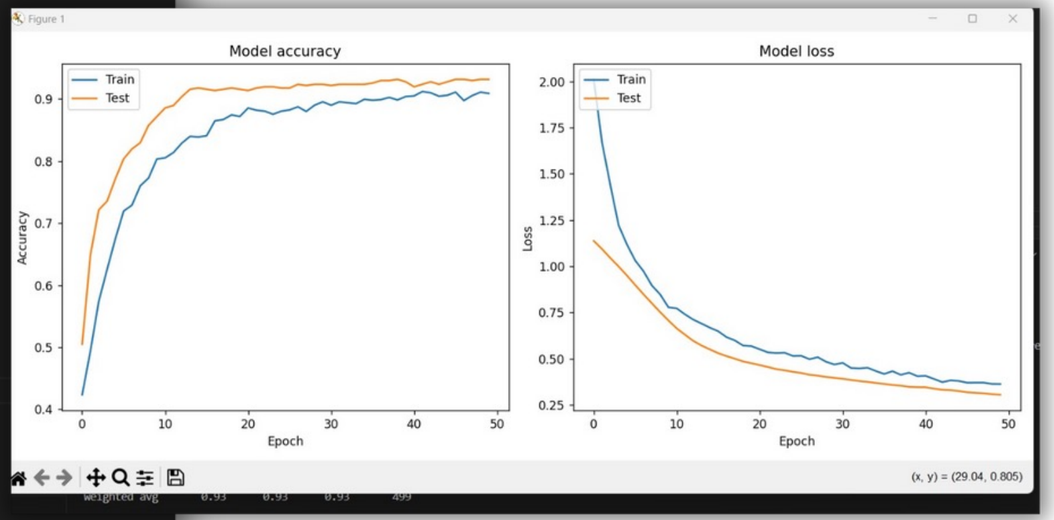
Wagon R 2018

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Engine Off												Engine Running										
Back				Front				Mid				Back				Front				Mid		
sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3		sensor 1	sensor 2	sensor 3
28.83	35.63	30.6		31	32.48	31.5		28.71	27.28	38.63		25.15	29.13	28.99		38.06	37.10	38.02		25.62	29.67	29.15
26.84	27.37	27.18		31.5	36.23	35.62		30.31	30.84	36.23		27.5	29.13	25.35		38.09	37.17	38.09		26.28	26.32	28.92
29.93	36.52	33.11		32.47	32.48	33.89		28.73	29.77	37.65		25.35	26.45	28.45		38.13	37.18	38.15		28.2	28.06	25.21
29.71	38.2	35.67		30.34	29.84	29.08		29.97	29.45	34.91		27.84	27.68	25.14		38.25	37.22	38.17		28.91	28.95	30
27.85	29.14	39.84		30.32	30.51	34.86		33.14	26.2	38.99		25.96	25.1	28.23		38.27	37.28	38.22		29.96	28.4	29.44
31.39	36.31	37.57		32.41	33.21	32.14		28.73	29.77	31.54		25.33	26.58	26.72		38.28	37.35	38.25		29.26	31.45	30.68
31.37	35.14	33.07		32.38	31.72	34.41		38.98	39.17	32.16		25	26.58	25.91		38.32	37.40	38.28		25.81	30.87	31.31
32.6	32.43	33.85		35.88	35.4	34.65		32.93	36.87	32.68		28.73	26.76	25.36		38.44	37.49	38.30		27.64	26.54	28.53
35.98	33.1	26.86		33.2	32.55	30.11		28.73	29.77	32.71		27.74	27.63	27.34		38.61	37.50	38.37		25.94	25.12	31.38
34.17	39.58	28.46		30.48	33.48	35.29		37.35	33.3	31.85		25.96	25.1	28.23		38.66	37.54	38.40		28.37	29.87	30.49
31.8	37.49	34.91		36.8	36.59	34.14		28.71	27.28	29.26		29.64	25.13	26.48		38.79	37.59	38.53		26.43	28.55	26.67
38.04	32.59	37.42		34.48	33.96	32.65		35.36	34.87	36.54		27.01	26.63	28.79		38.79	37.60	38.67		29.31	26.15	25.86
35.1	33.19	38.1		34.62	29.25	30.94		39.94	29.75	39.03		29.95	25.8	26.21		38.83	37.61	38.69		30.64	31	27.29
26.65	29.32	33.06		33.97	30.39	33.55		28.73	29.77	36.26		29.03	29.57	25.26		38.84	37.66	38.72		25.26	25.52	30.89
39.57	30.56	28.21		33.88	36.04	35.77		37.01	26.37	37.88		28.53	26.13	29.49		38.90	37.68	38.84		25.14	31.9	27.8
34.83	35.37	34		36.51	31.14	29.06		37.98	32.4	33.23		26.42	27.82	26.9		38.95	37.72	38.90		28.32	28.86	31.37
27.78	36.27	27.31		36.58	31.58	31.72		29.58	29.61	31.71		25.55	29.62	27.81		38.97	37.75	38.93		26.13	27.22	30.15
32.6	32.43	33.85		31.09	32.73	35.68		29.35	35.93	35.17		29.5	26.27	25.17		38.98	37.77	38.94		26.17	29.96	27.81
35.98	33.1	26.86		33.92	31.22	33.46		37.01	26.37	30.22		25.51	27.74	28.24		39.03	37.78	38.95		26.79	26.25	29.59
34.17	39.58	28.46		36.78	30.57	32.71		37.98	32.4	39.15		25.83	27.25	26.85		39.03	37.84	39.01		27.07	29.62	27.89
31.8	37.49	34.91		29.35	31.4	31.14		29.58	29.61	39.13		27.83	27.1	27.92		39.06	38.18	39.23		27.25	28.93	26.82
38.04	32.59	37.42		33.88	32.85	36.39		29.35	35.93	29.71		29.62	28.63	25.62		39.26	38.22	39.26		27.15	31.8	25.84
35.1	33.19	38.1		30.32	30.51	34.86		30.11	32.58	37.86		29.38	27.49	27.88		39.30	38.25	39.35		29.36	28.64	31.87
26.65	29.32	33.06		32.41	33.21	32.14		37.79	36.71	32.36		25.96	25.1	28.23		39.34	38.33	39.37		30.41	26.18	26.28
39.57	30.56	28.21		32.38	31.72	34.41		35.44	28.55	30.82		27.37	27.34	28.27		39.36	38.36	39.41		26.03	30.93	31.61
34.83	35.37	34		35.88	35.4	34.65		38.08	34.27	37.6		29.16	28.16	26.47		39.43	38.43	39.42		28.99	29.63	29.26

Sheet1

CNN Model

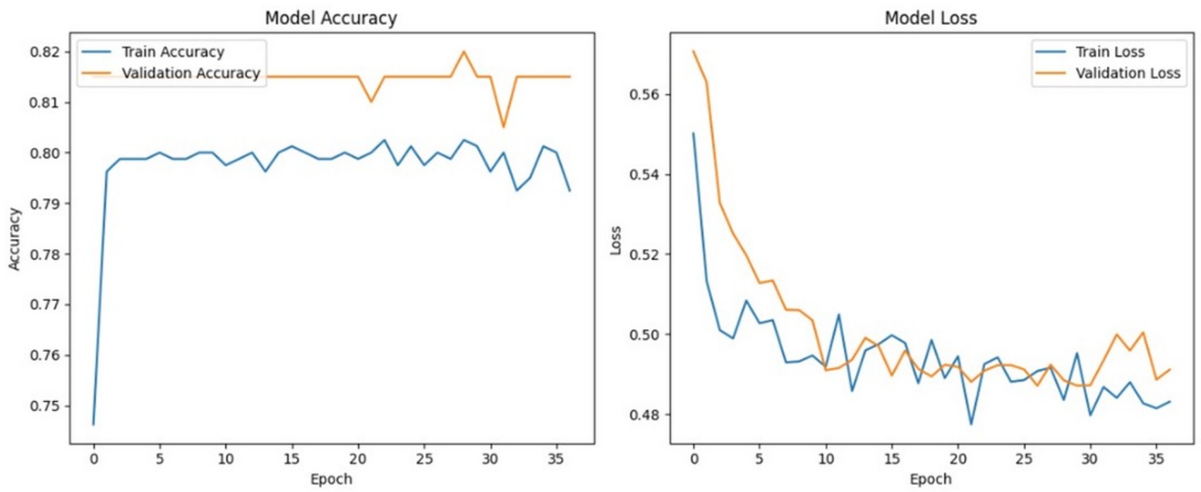
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Conv1D, Flatten, MaxPooling1D, Dropout, BatchNormalization
from tensorflow.keras.regularizers import l2
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report

# Load the adjusted dataset
df = pd.read_csv('car_temp_abnomility_dataset.csv')

# Label encoding: Convert 'normal' to 0 and 'abnormal' to 1
df['label'] = df['label'].apply(lambda x: 0 if x == 'normal' else 1)

# Feature Scaling
scaler = MinMaxScaler()
X = scaler.fit_transform(df[['sensor_front', 'sensor_mid', 'sensor_back']])

# Convert labels to categorical (one-hot encoding)
y = to_categorical(df['label'])

# Reshape X to 3D array (samples, timesteps, features)
X = X.reshape(X.shape[0], X.shape[1], 1)

# Moderate noise to slow down initial learning
noise_factor = 0.15  # Moderate noise factor
X_train_noisy = X + noise_factor * np.random.normal(loc=0.0, scale=1.0, size=X.shape)
X_train_noisy = np.clip(X_train_noisy, 0., 1.)

# Split the data
X_train, X_test, y_train, y_test = train_test_split(X_train_noisy, y, test_size=0.2, random_state=42)
```



Abnormal	0.94	0.95	0.94	247
accuracy			0.94	499
macro avg	0.94	0.94	0.94	499
weighted avg	0.94	0.94	0.94	499

RNN Model(LSTM)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
from tensorflow.keras.optimizers import Adam

# Load the dataset
df = pd.read_csv('sensor_reading_abno.csv')

# Prepare the input features and labels
features = ['1000_rpm_temprature', '1000_rpm', '2000_rpm_temprature', '3000_rpm_temprature',
            '4000_rpm_temprature', '5000_rpm_temprature', 'cabin_without_ac_sunny_day',
            'rear_without_ac_sunny_day', '2000_rpm', '3000_rpm', '4000_rpm', '5000_rpm']
X = df[features].values
y = df['label'].apply(lambda x: 0 if x == 'normal' else 1).values
y = to_categorical(y)

# Normalize features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
X_scaled = X_scaled.reshape(X_scaled.shape[0], X_scaled.shape[1], 1)

# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split( *arrays: X_scaled, y, test_size=0.2, random_state=42)
```




```
25/25 - 0s - loss: 0.4831 - accuracy: 0.7925 - val_loss: 0.4911 - val_accuracy: 0.8150 - 213ms/epoch - 9ms/step
7/7 [==============================] - 1s 3ms/step
Accuracy: 81.50%
```

SVM Model

```
SVM Accuracy: 0.58

Classification Report:
              precision    recall  f1-score   support

           0       0.84      0.60      0.70       163
           1       0.22      0.51      0.31        37

    accuracy                           0.58       200
   macro avg       0.53      0.55      0.50       200
weighted avg       0.73      0.58      0.63       200
```

```python
# Build SVM model
svm_model = SVC(class_weight='balanced')  # Adjusting class weights for handling imbalanced data
svm_model.fit(X_train, y_train)

# Predict labels for test set
y_pred = svm_model.predict(X_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("SVM Accuracy:", accuracy)

# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
plt.figure(figsize=(8, 6))
sns.heatmap(cm, annot=True, fmt="d", cmap='Blues', xticklabels=['Normal', 'Abnormal'], yticklabels=['Normal', 'Abnormal'])
plt.title('Confusion Matrix for SVM Model')
plt.ylabel('True Label')
plt.xlabel('Predicted Label')
plt.show()

# Classification Report with zero_division=1
print("\nClassification Report:\n", classification_report(y_test, y_pred, zero_division=1))
```

Model Accuracy

Used Algorithm	Overall Accuracy
CNN Algorithm	94.0%
Support Vector Machine(SVM)	81.50%
RNN Model(LSTM)	81.50%

CNN Model Tests

```python
import numpy as np

from tensorflow.keras.models import load_model


model = load_model('car_temp_abnomility_model.h5')

def predict_sensor_data(sensor_data):

    fire_state = ['normal' , 'abnormal']

    sensor_data = np.array(sensor_data).reshape(1, 3, 1)  # Reshape for the model
    prediction = model.predict(sensor_data)
    label = np.argmax(prediction, axis=1)[0]

    return fire_state[label]


new_data = [24, 26, 31]  # Example new sensor data
fire_state = predict_sensor_data(new_data)
print(fire_state)

```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

  warnings.warn(
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metr
l.
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a freshl
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 71ms/step
normal
```

```python
import numpy as np

from tensorflow.keras.models import load_model


model = load_model('car_temp_abnomility_model.h5')

def predict_sensor_data(sensor_data):

    fire_state = ['normal' , 'abnormal']

    sensor_data = np.array(sensor_data).reshape(1, 3, 1)  # Reshape for the model
    prediction = model.predict(sensor_data)
    label = np.argmax(prediction, axis=1)[0]

    return fire_state[label]


new_data = [146, 105, 95]  # Example new sensor data
fire_state = predict_sensor_data(new_data)
print(fire_state)

```

```
PROBLEMS 7  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

  warnings.warn(
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train
l.
WARNING:absl:Error in loading the saved optimizer state. As a result, your model is starting with a freshly initialized optimizer.
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 73ms/step
abnormal
```

IOT Integration

```python
def read_sensor_values():

    line = ser.readline().decode('utf-8').strip()

    # Split the line by comma
    data = line.split(',')

    # Print the data

    if len(data) == 8:
        # print(data)

        temsensor1 = int(data[0])
        temsensor2 = int(data[1])
        temsensor3 = int(data[2])

        gassensor1 = int(data[3])
        gassensor1 = int(data[4])
        gassensor1 = int(data[5])

        gps_long = data[6]
        gps_lat = data[7]

        return data
```

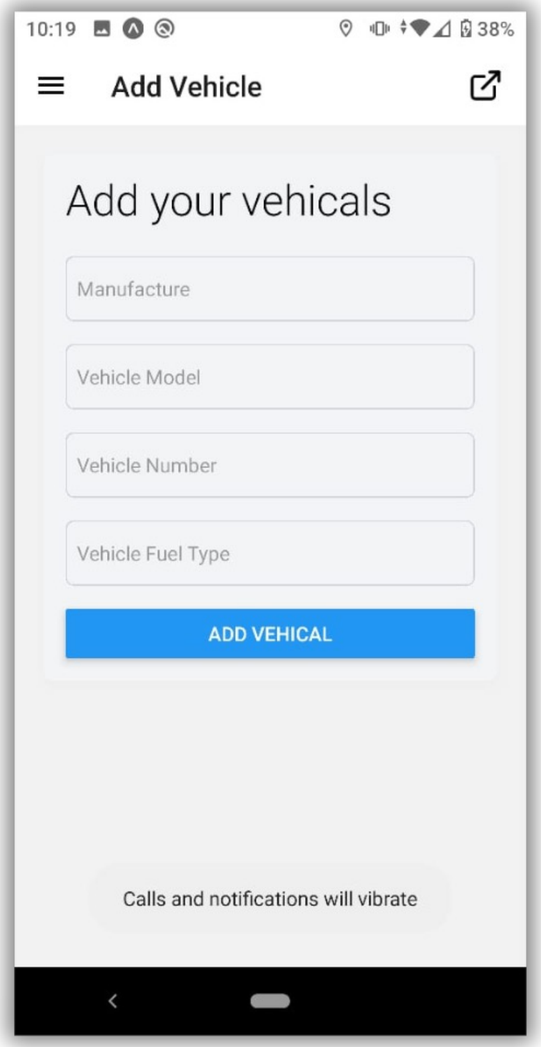
```python
    sensor_values = read_sensor_values()



if sensor_values is not None:
    # print("Sensor 1: {sensor_values[0]}, Sensor 2: {sensor_values[1]}, Sensor 3: {sensor_values[2]},
    #       Sensor 4: {sensor_values[3]}, Sensor 5: {sensor_values[4]}, Sensor 6: {sensor_values[5]}" #gas
    #       f", Sensor 6: {sensor_values[6]}, Sensor 6: {sensor_values[7]}")   # GPS --> All comeing as str

    veryfi = 1


    w_sensors = [ int (sensor_values[0]),int (sensor_values[1]),int (sensor_values[2]),
                  int (sensor_values[3]),int (sensor_values[4]),int (sensor_values[5]),
                  sensor_values[6],sensor_values[7],veryfi]
```

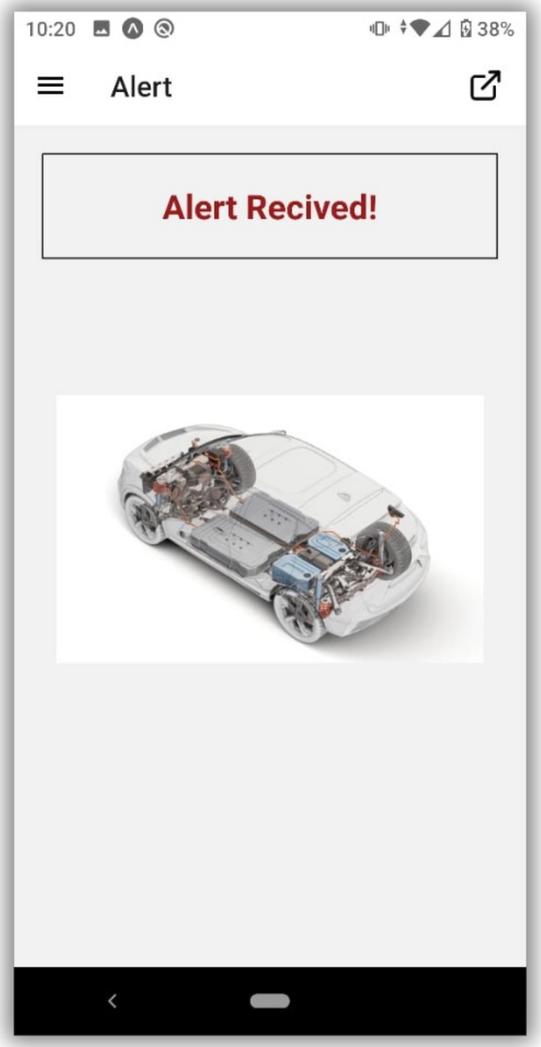
```python
from arduino_communication_2 import get_sensor_data , send_actuators_data ,send_actuators_reset,arduino_systme_init_wait
import time

arduino_systme_init_wait(info=False)


for x in range(2000):

    sensor_deta = get_sensor_data()

    print(sensor_deta)

    time.sleep(1)


# while True:
```

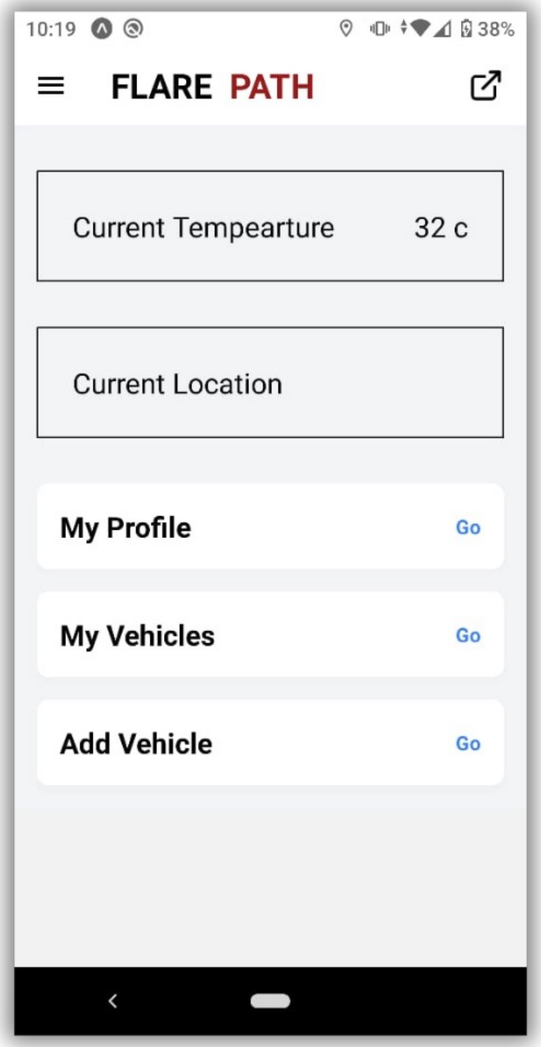
```
data = get_sensor_data()

temsensor1 = int(data[0])
temsensor2 = int(data[1])
temsensor3 = int(data[2])

gassensor1 = int(data[3])
gassensor2 = int(data[4])
gassensor3 = int(data[5])

gps_lat = data[6]
gps_lon = data[7]


fire_type_sensor_data = [temsensor1, temsensor2, temsensor3]
fire_type = predict_fire_type(fire_type_sensor_data)

new_data = [temsensor1, temsensor2, temsensor3]
fire_state = predict_sensor_data(new_data)

print(type(fire_type) , fire_state)
```

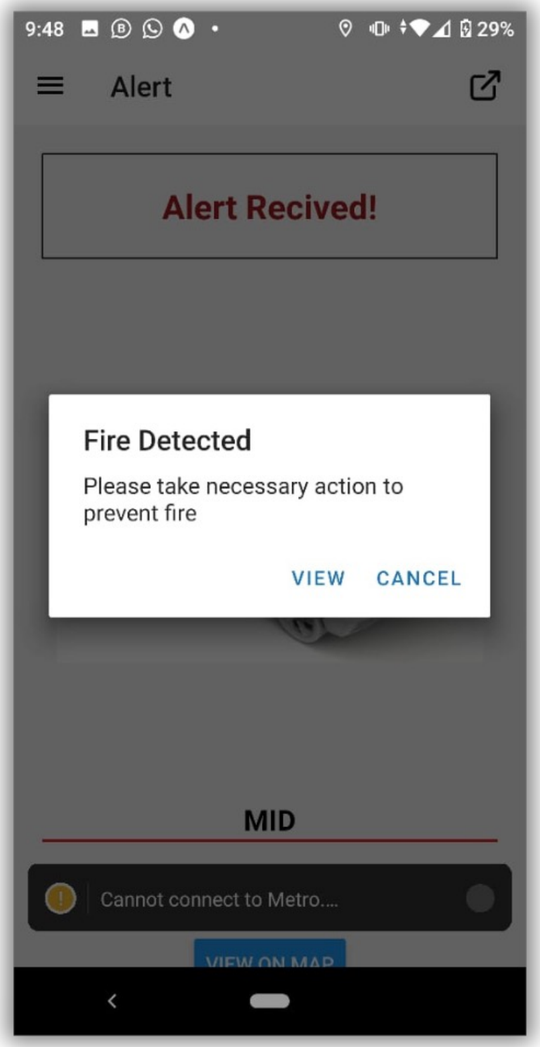
```
if fire_state == 'abnormal':

    payload = str(gps_lat) + ',' + str(gps_lon)  + ',' + str(wehical_no) + ',' + str(fire_type)
    car_clnt.publish('f_station', payload)

    print('Signal Send successful....')


if fire_state != 'normal':

    send_actuators_data([0, 1])
```

Mobile App


10:19
38%
Add Vehicle
Add your vehicals
Manufacture
Vehicle Model
Vehicle Number
Vehicle Fuel Type
ADD VEHICAL
Calls and notifications will vibrate


10:20
38%
Alert
Alert Recived!


10:19
38%
FLARE PATH
Current Tempearture
32 c
Current Location
My Profile
Go
My Vehicles
Go
Add Vehicle
Go

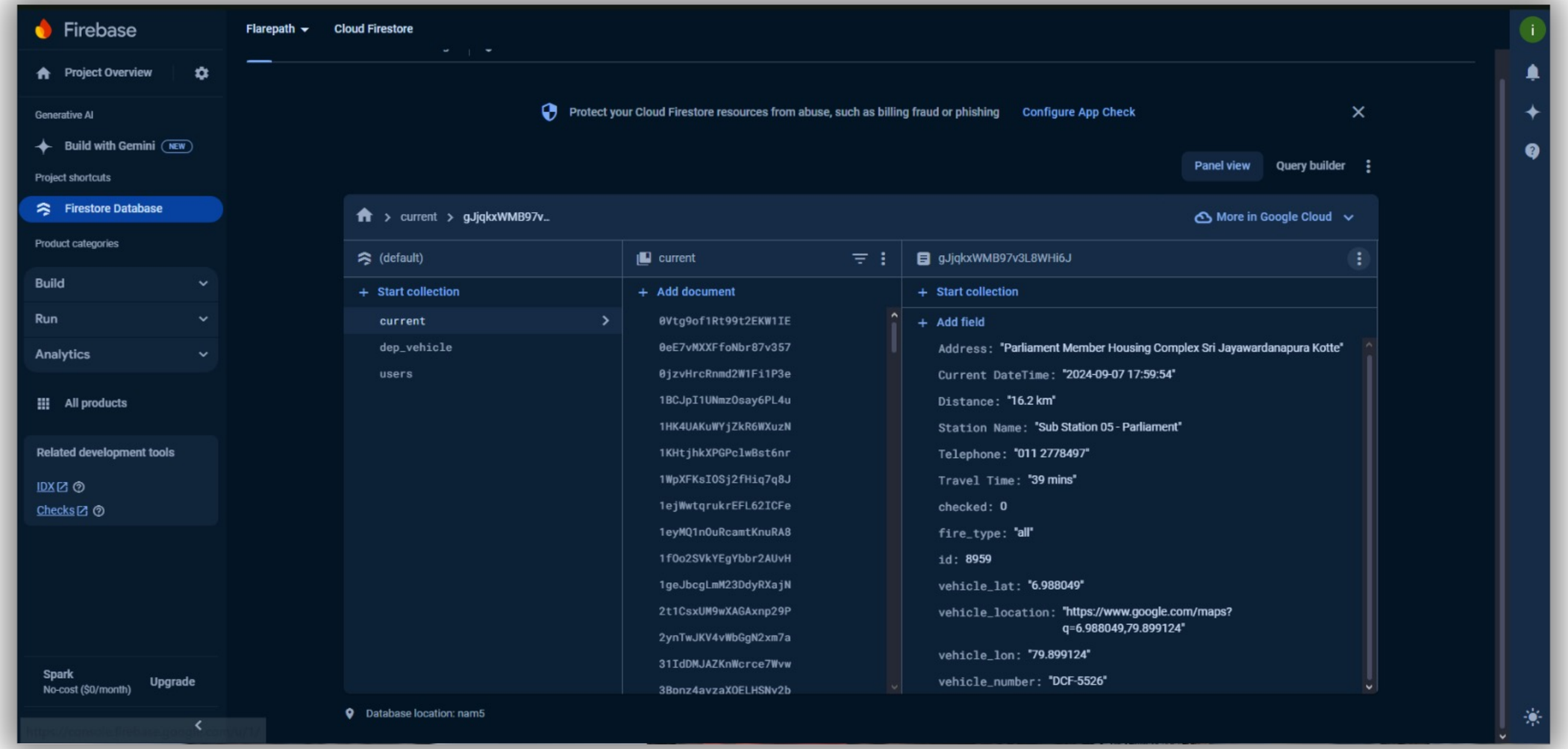

9:48
29%
Alert
Alert Recived!
Fire Detected
Please take necessary action to prevent fire
VIEW
CANCEL
MID
Cannot connect to Metro....

Mobile App - Codes

```
import { View, Text} from 'react-native'
import React, { useEffect, useState } from 'react'
import axios from 'axios'

import { useRouter } from 'expo-router'
import { Drawer } from 'expo-router/drawer'
import * as Location from 'expo-location';
import { HomeIcon } from 'lucide-react-native'



const UserHome = () => {

  const [location, setLocation] = useState(null);
  const [errorMsg, setErrorMsg] = useState(null);

  const [ad,setAd]=useState('')

  const router = useRouter()


  useEffect(() => {
    (async () => {

      let { status } = await Location.requestForegroundPermissionsAsync();
      if (status !== 'granted') {
        setErrorMsg('Permission to access location was denied');
        return;
      }

LEMS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

History restored
```

```
import { View, Text,TextInput,Button, Alert, ActivityIndicator } from 'react-native'
import React,{useState} from 'react'
import { Drawer } from 'expo-router/drawer';
import {  Plus } from 'lucide-react-native';
import { db } from '../../utils/firebaseConfig';
import { addDoc, arrayUnion, collection, doc,setDoc,updateDoc } from 'firebase/firestore';
import { useAuth } from '../../context/authContext';

const VehicalRegistration = () => {

  const [values, setValues] = useState({
    manfacture: '',
    modelName: '',
    fuelType: '',
    vehicalNumber:''
  })

  const [loading, setLoading] = useState(false)

  const {user}=useAuth()

  async function handleSubmit() {

    console.log(values)

    if (values.fuelTypename==='' || values.manfacture==='' ||
      values.modelName==='' || values.vehicalNumber==='') {
      return Alert.alert('Please fill all fields','all the fields are required')
    }

MS 2  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS
```

Mobile App - Firebase



Methodology

Requirement Analysis

Functional Requirements

- Install sensors for temperature
- Implement algorithms for sensor data analysis and fire prediction.
- Process sensor data in real-time for immediate response.
- Develop a mechanism to alert drivers of potential fire hazards and mobile app.
- Monitor sensor performance continuously for accurate detection.

Non-Functional Requirements

- **Ensure quick data processing and alert generation.**
- **Provide accurate detection and alerting under all conditions.**
- **Design the system to accommodate a large number of vehicles and sensors.**
- **Protect sensor data from unauthorized access.**
- **Create a user-friendly interface for easy understanding and response.**
- **Ensure the system is easy to maintain and update.**

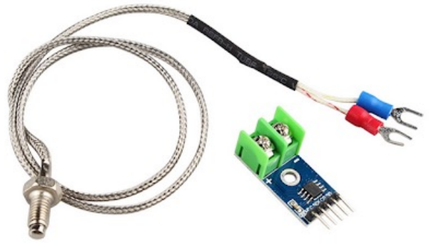
Tools and Technologies

Hardware Tools

- ESP32
- Arduino mega board
- Arduino UNO board
- max6675 K type thermocouple
- Temperature Sensor

Software Tools

- Micro Python
- Pycharm
- Python
- React Native



References

[1] Research on Prediction Method of Armored Vehicle Fire Control System Based on BAS-RVM. (2020, August 5). IEEE Conference Publication | IEEE Xplore. https://ieeexplore.ieee.org/document/9353131

[2] Li, D., Zhu, G., Zhu, H., Yu, Z., Gao, Y., & Jiang, X. (2017, September 1). Flame spread and smoke temperature of full-scale fire test of car fire. Case Studies in Thermal Engineering. https://doi.org/10.1016/j.csite.2017.08.001

[3] Jiang, X., Zhu, G., Zhu, H., & Li, D. Y. (2018, January 1). Full-scale Experimental Study of Fire Spread Behavior of Cars. Procedia Engineering.

[4]Shintani, Y., Kakae, N., Harada, K., & Takahashi, W. (2004, March 1). *Experimental Investigation of Burning Behavior of Automobiles*. ResearchGate.

[5]Deckers, X., Haga, S. J., Tilley, N., & Merci, B. (2013, April 1). *Smoke control in case of fire in a large car park: CFD simulations of full-scale configurations*. Fire Safety Journal. https://doi.org/10.1016/j.firesaf.2012.02.005

Abeywardhana D.N| IT21133718
Specializing in Information Technology

Fire Severity Assessment for Emergency Services

Research Problem

How can fire departments optimize their protocols to enable firefighters to anticipate and mitigate fires more efficiently before upon arrival?

Fire department emergency systems are not designed with prepare their resources related to real time fire severity with more efficiency before reaching the current location.

How can fire departments prepare their equipment and resources in advance to mitigate potential risks?

Existing fire Emergency systems are not designed with identify the vehicle fire severity to manage their resources and safety of the responders and the public before arriving.

Research Gap

- Detecting the real time localization of the spreading fire position(Front/Mid/Back) on a vehicle at the moment of the fire.
- Effectively integrating and analysing data from variety of sensors(flame,temperature,smoke) in real time to predict how fire spreading with locations and fire severity levels.
- Show the real time fire severity levels through mobile application and the fire department dashboard.

Objectives

Predict the classification of fire severity of the vehicle fire

- **Install sensors in vehicle to detect various fire indicators**
- **Train highly accurate ML models with a collection of real-time data from sensors under various conditions.**
- **After Analyzing data then predict fire severity level and Send to the cloud and find a nearest fire department.**
- **Display the real time fire severity level in the mobile app.**

SVM Model

```python
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score
from sklearn.preprocessing import LabelEncoder


def load_and_prepare_data(csv_filename):

    data = pd.read_csv(csv_filename)

    label_encoder = LabelEncoder()
    data['label'] = label_encoder.fit_transform(data['label'])

    X = data.drop('label', axis=1)
    y = data['label']
    return X, y, label_encoder


def train_svm_model(X_train, y_train):
    # Initialize and train the SVM model
    svm_model = SVC(kernel='linear', random_state=42)
    svm_model.fit(X_train, y_train)
    return svm_model


def evaluate_model(svm_model, X_test, y_test):
    # Make predictions and evaluate the model
    y_pred = svm_model.predict(X_test)
    accuracy = accuracy_score(y_test, y_pred)
    print(f"Model Accuracy: {accuracy}")


def predict_with_new_data(svm_model, label_encoder, new_data):
    # Structure the new data with column names
    column_names = ['sensor_front', 'sensor_mid', 'sensor_back']  # Ensure these match your training data
    new_data_df = pd.DataFrame([new_data], columns=column_names)
    prediction = svm_model.predict(new_data_df)
    # Decode the numerical prediction back to a label
    predicted_label = label_encoder.inverse_transform(prediction)
    return predicted_label


csv_filename = 'fire_type_detection_train_dataset.csv'
X, y, label_encoder = load_and_prepare_data(csv_filename)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
svm_model = train_svm_model(X_train, y_train)
evaluate_model(svm_model, X_test, y_test)


def predict_fire_type(new_sensor_data):
    global svm_model

    predicted_label = predict_with_new_data(svm_model, label_encoder, new_sensor_data)

    return predicted_label[0]
```

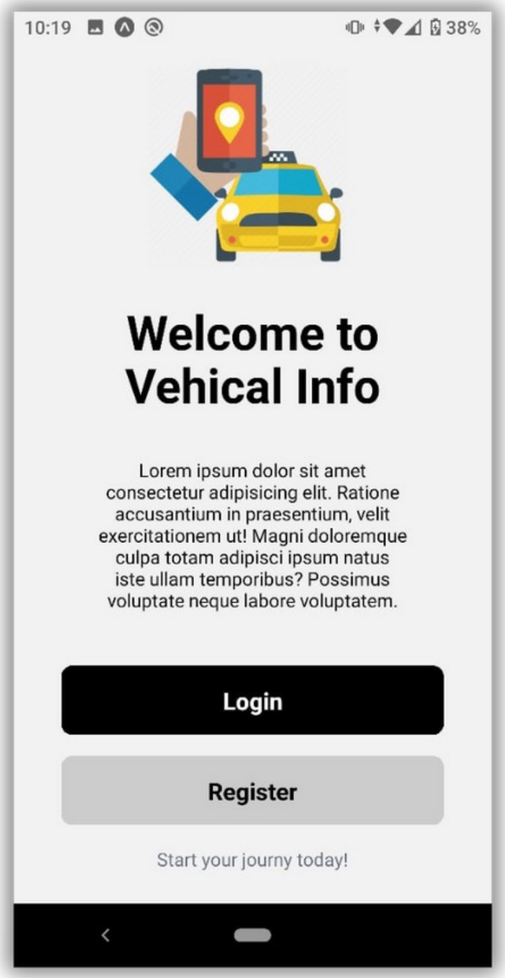
Model Accuracy: 95.20%

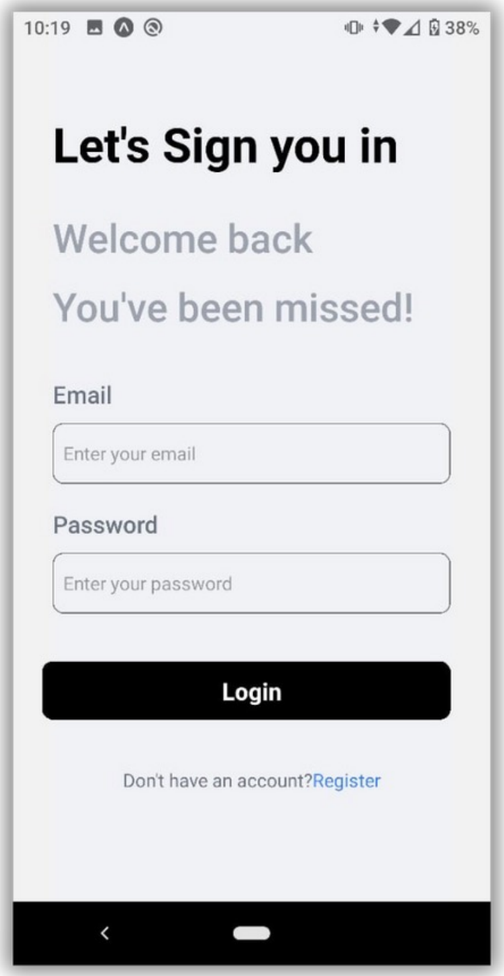
Arduino Intergration

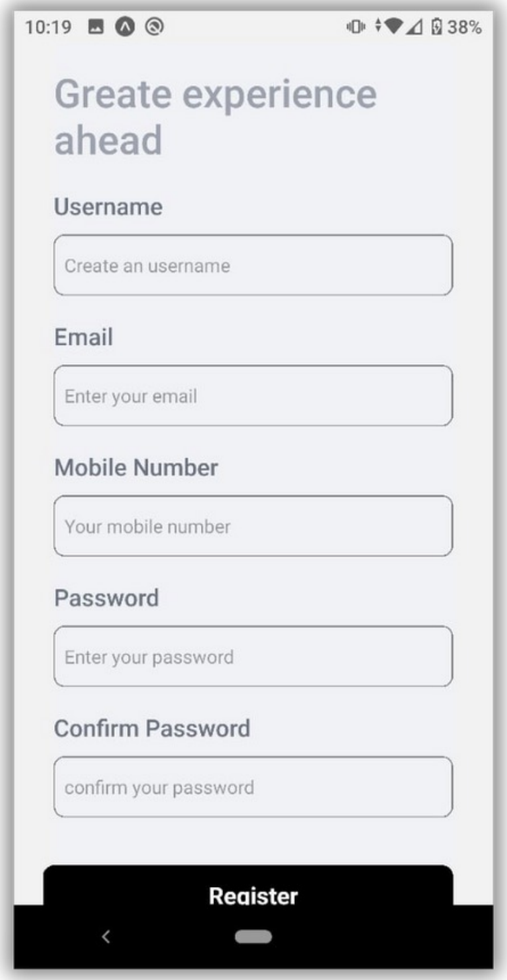
```python
fire_type_sensor_data = [temsensor1, temsensor2, temsensor3]
fire_type = predict_fire_type(fire_type_sensor_data)

new_data = [temsensor1, temsensor2, temsensor3]
fire_state = predict_sensor_data(new_data)

print(type(fire_type) , fire_state)



if fire_state == 'abnormal':

    payload = str(gps_lat) + ',' + str(gps_lon)  + ',' + str(wehical_no) + ',' + str(fire_type)
    car_clnt.publish('f_station', payload)

    print('Signal Send successful....')



if fire_state != 'normal':

    send_actuators_data([0, 1])


demo_mood(gps_lat , gps_lon , wehical_no ,fire_type)

time.sleep(1)
```

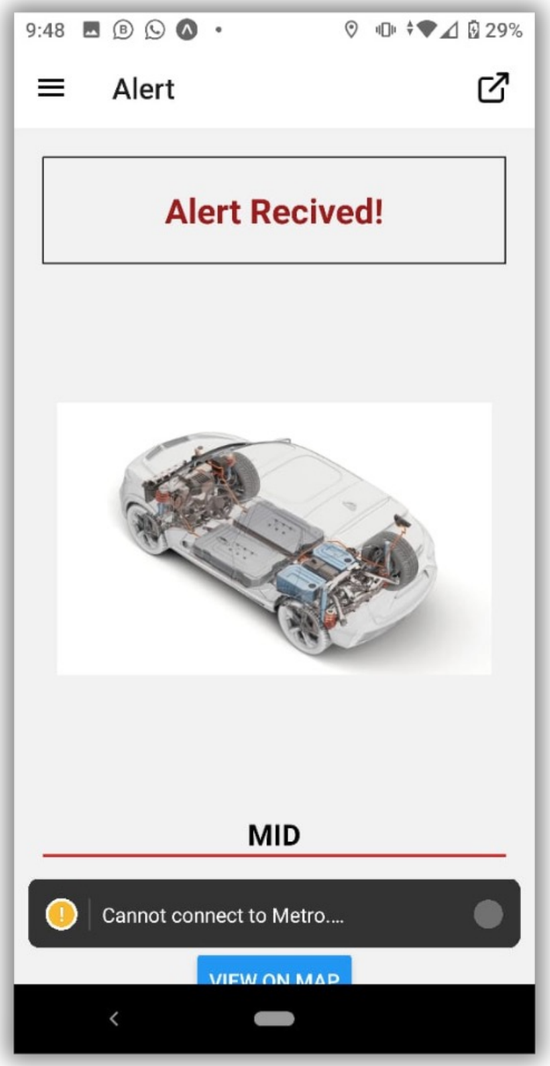
```python
def run_sensor_task(serial_port='COM6', baud_rate=9600):  ########  <------ Set comport


    ser = serial.Serial(serial_port, baud_rate, timeout=1)

    def send_command(command):
        ser.write(command.encode())


    def read_sensor_values():

        line = ser.readline().decode('utf-8').strip()

        # Split the line by comma
        data = line.split(',')

        # Print the data

        if len(data) == 8:
            # print(data)

            temsensor1 = int(data[0])
            temsensor2 = int(data[1])
            temsensor3 = int(data[2])

            gassensor1 = int(data[3])
            gassensor1 = int(data[4])
            gassensor1 = int(data[5])

            gps_long = data[6]
            gps_lat = data[7]

            return data

        else:

            return [0,0]
```

Mobile App


10:19
38%
Welcome to Vehical Info
Lorem ipsum dolor sit amet consectetur adipisicing elit. Ratione accusantium in praesentium, velit exercitationem ut! Magni doloremque culpa totam adipisci ipsum natus iste ullam temporibus? Possimus voluptate neque labore voluptatem.
Login
Register
Start your journy today!


10:19
38%
Let's Sign you in
Welcome back
You've been missed!
Email
Enter your email
Password
Enter your password
Login
Don't have an account?Register


10:19
38%
Greate experience ahead
Username
Create an username
Email
Enter your email
Mobile Number
Your mobile number
Password
Enter your password
Confirm Password
confirm your password
Register


9:48
29%
Alert
Alert Recived!
MID
Cannot connect to Metro....

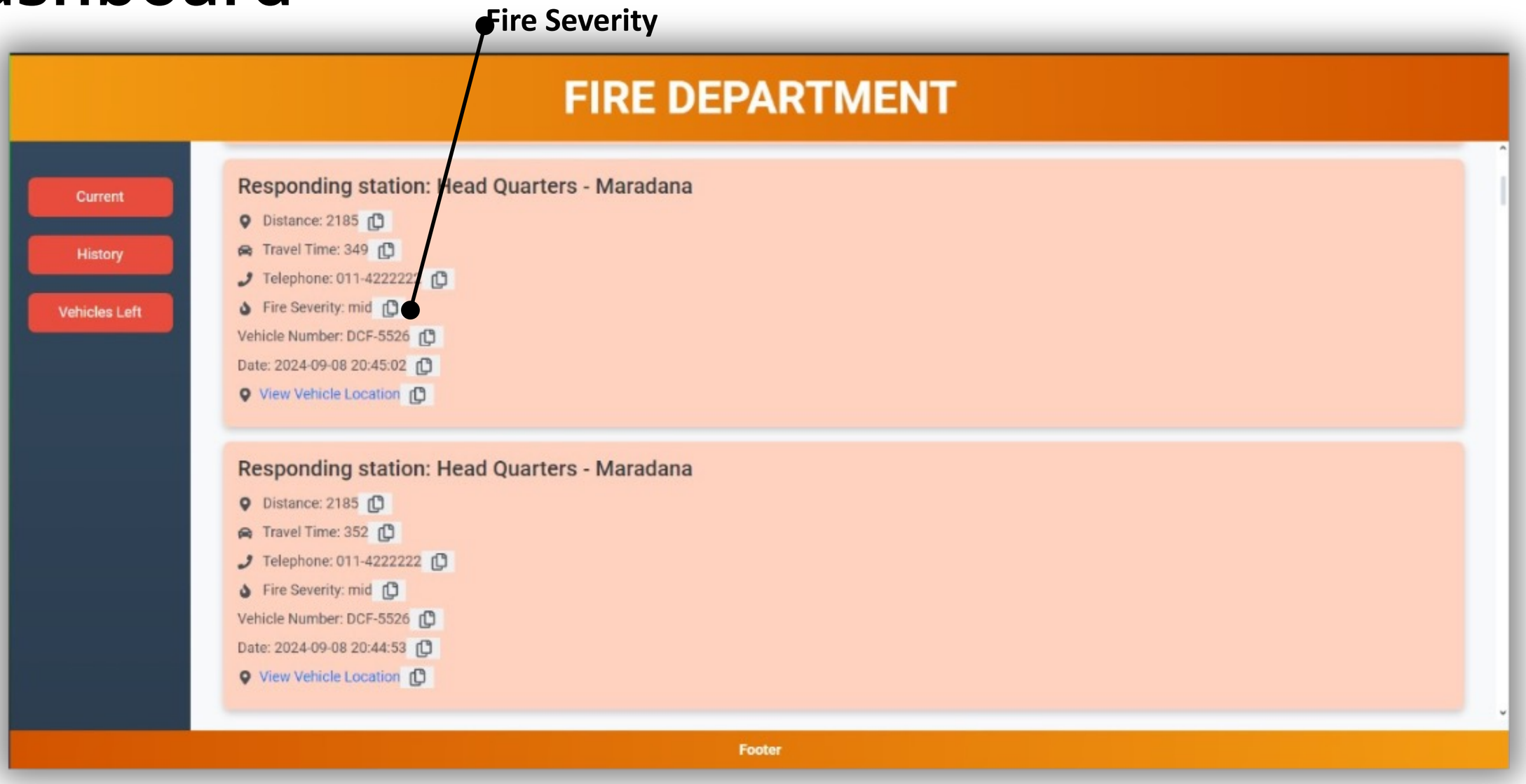
Mobile App - Codes

```
import { auth } from '../utils/firebaseConfig'
import { signInWithEmailAndPassword } from 'firebase/auth'

import { useAuth } from '../context/authContext'

const Login = () => {

  const { setUser, setAuthenticated } = useAuth()

  const [email, setEmail] = React.useState('')
  const [password, setPassword] = React.useState('')
  const [emailError, setEmailError] = React.useState(false)
  const [passwordError, setPasswordError] = React.useState(false)
  const [error, setError] = React.useState()

  const [loading, setLoading] = React.useState(false)

  const router = useRouter()

  async function login_user() {

    setEmailError(false)
    setPasswordError(false)

    if (email.length < 1) {
      setEmailError(true)
      return
    }
    if (password.length < 1) {
      setPasswordError(true)
      return
    }

    setLoading(true)
```

```
import { View, Text,TextInput,Button, Alert, ActivityIndicator } from 'react-native'
import React,{useState} from 'react'
import { Drawer } from 'expo-router/drawer';
import {  Plus } from 'lucide-react-native';
import { db } from '../../utils/firebaseConfig';
import { addDoc, arrayUnion, collection, doc,setDoc,updateDoc } from 'firebase/firestore';
import { useAuth } from '../../context/authContext';

const VehicalRegistration = () => {

  const [values, setValues] = useState({
    manfacture: '',
    modelName: '',
    fuelType: '',
    vehicalNumber:''
  })

  const [loading, setLoading] = useState(false)

  const {user}=useAuth()

  async function handleSubmit() {

    console.log(values)

    if (values.fuelTypename==='' || values.manfacture==='' ||
      values.modelName==='' || values.vehicalNumber==='') {
      return Alert.alert('Please fill all fields','all the fields are required')
    }
```

MS 2 OUTPUT DEBUG CONSOLE TERMINAL PORTS COMMENTS

Dashboard


Fire Severity
FIRE DEPARTMENT
Current
History
Vehicles Left
Responding station: Head Quarters - Maradana
Distance: 2185
Travel Time: 349
Telephone: 011-4222222
Fire Severity: mid
Vehicle Number: DCF-5526
Date: 2024-09-08 20:45:02
View Vehicle Location
Responding station: Head Quarters - Maradana
Distance: 2185
Travel Time: 352
Telephone: 011-4222222
Fire Severity: mid
Vehicle Number: DCF-5526
Date: 2024-09-08 20:44:53
View Vehicle Location
Footer

Completion Of The Component

Progress at PP1 – 50%

- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Identify Software requirements
- Collect data manually (Measure temperatures/RPM)
- Create dataset
- Data preprocessing
- Model Selection
- Design System flow chart
- Design Mobile App main page , home page , login page , signup page wireframes
- Design Mobile App Verification page, Password reset , Fire severity , Optimal path page wireframes
- Design Mobile App main page , home page , login page , signup page UI/UX designs
- Design Mobile App Verification page, Password reset , Fire severity , Optimal path page UI/UX designs
- Collect data manually from Fire department and create dataset
- Design System flow chart
- Design Dashboard, Active Incidents, Active Incidents view and Logs wireframe
- Design Dashboard, active Incidents, active Incidents view and logs frontend design
- Complete Mobile App UI/UX flows and Prototypes

Completion Of The Component

Progress at PP2 – 90%

- Improve the Data collection
- Improve ML Model Selection
- Improve Data Processing
- Improve Model Training
- Implementing IOT device
- Intergrating IOT device
- Testing the IOT device
- Getting Real time Data
- Improve Arduino codes
- Developing mobile appication
- Testing the integrated IOT device and ML model
- Validating the system

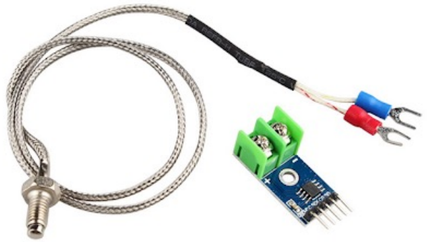
In progress – 10%

- UI/UX optimization for mobile app

Tools and Technologies

Hardware Tools

- ESP32
- Arduino mega board
- Arduino UNO board
- max6675 K type thermocouple
- Temperature Sensor



Software Tools

- Micro Python
- c/c++
- Arduino IDE
- Pycharm
- Python
- React Native

Requirement Analysis

Functional Requirements

- System should be able to collect and store real-time data under various conditions from IoT-enabled sensors.
- System should be able to send real time fire severity to the mobile app without any delay.
- System should be able to send real time fire severity levels to the fire department dashboard without any delay.
- The system must be capable of integrating with existing vehicle systems.

Non-Functional Requirements

- Interfaces should be more user-friendly.
- Application should be reliable
- Should respond Realtime.
- Should have high security.

References

[1]Jiang, X. H., Zhu, G. Q., Zhu, H., & Li, D. Y. (2018). Full-scale Experimental Study of Fire Spread Behavior of Cars. Procedia Engineering, 211, 297–305. https://doi.org/10.1016/j.proeng.2017.12.016

[2] Mohd Tohir, M. Z., & Spearpoint, M. (2013). Distribution analysis of the fire severity characteristics of single passenger road vehicles using heat release rate data. Fire Science Reviews. http://www.firesciencereviews.com/content/2/1/5

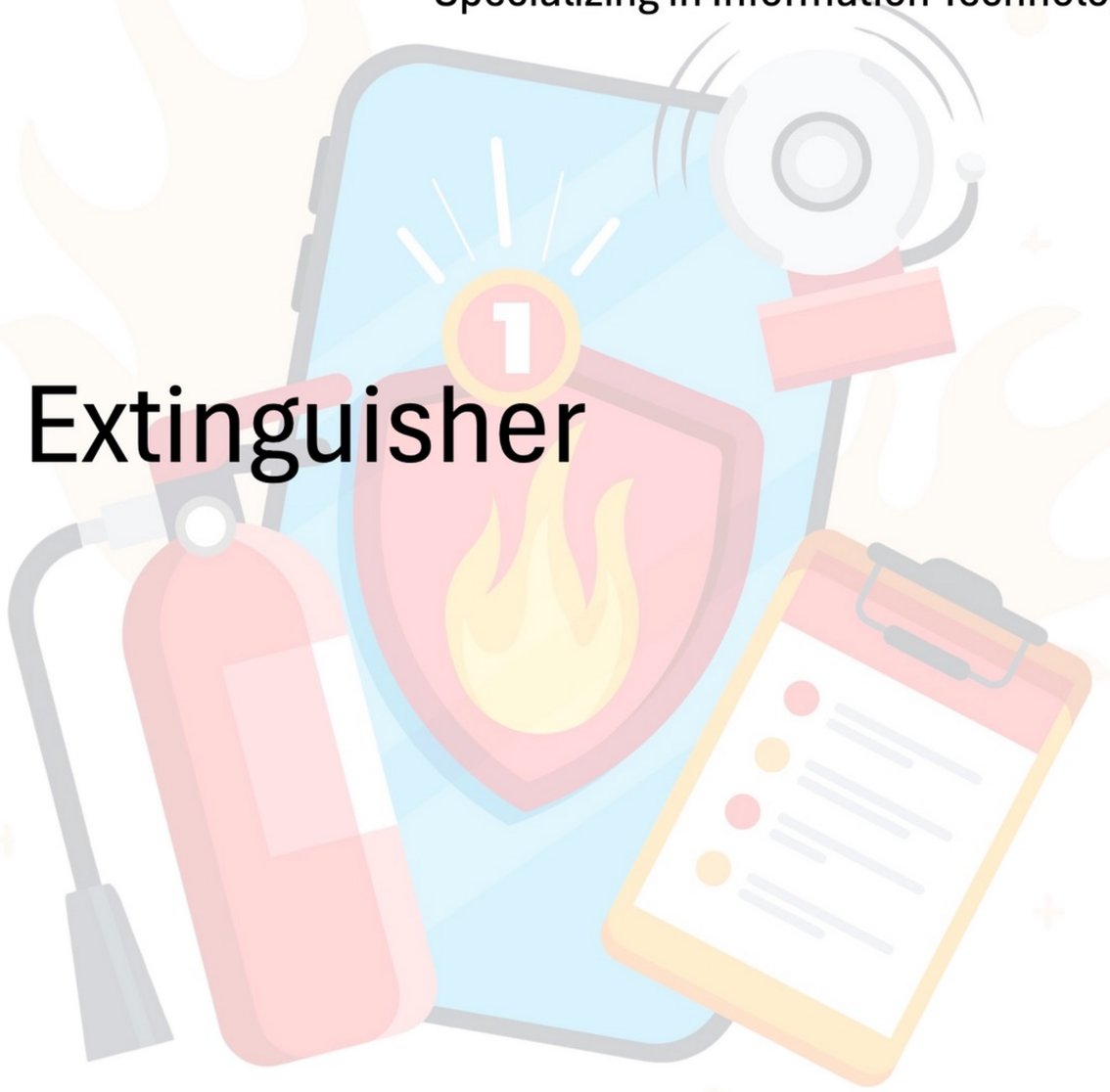
[3] Sowah, R., Ampadu, K. O., Ofoli, A. R., Koumadi, K., Mills, G. A., & Nortey, J. (2019, March). A Fire-Detection and Control System in Automobiles: Implementing a Design That Uses Fuzzy Logic to Anticipate and Respond. IEEE Industry Applications Magazine, 25(2), 57–67. https://doi.org/10.1109/mias.2018.287518

[4]Mathavan, J. J., Faslan, A., Basith, N. U. A., & Wanigasinghe, W. (2020, June). Hardware Implementation of Fire Detection, Control and Automatic Door Unlocking System for Automobiles. *2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184)*. https://doi.org/10.1109/icoei48184.2020.9142990

Dharmagunawardana W.M.P.I| IT21132346

Specializing in Information Technology

IoT-based Fire Extinguisher Mechanism



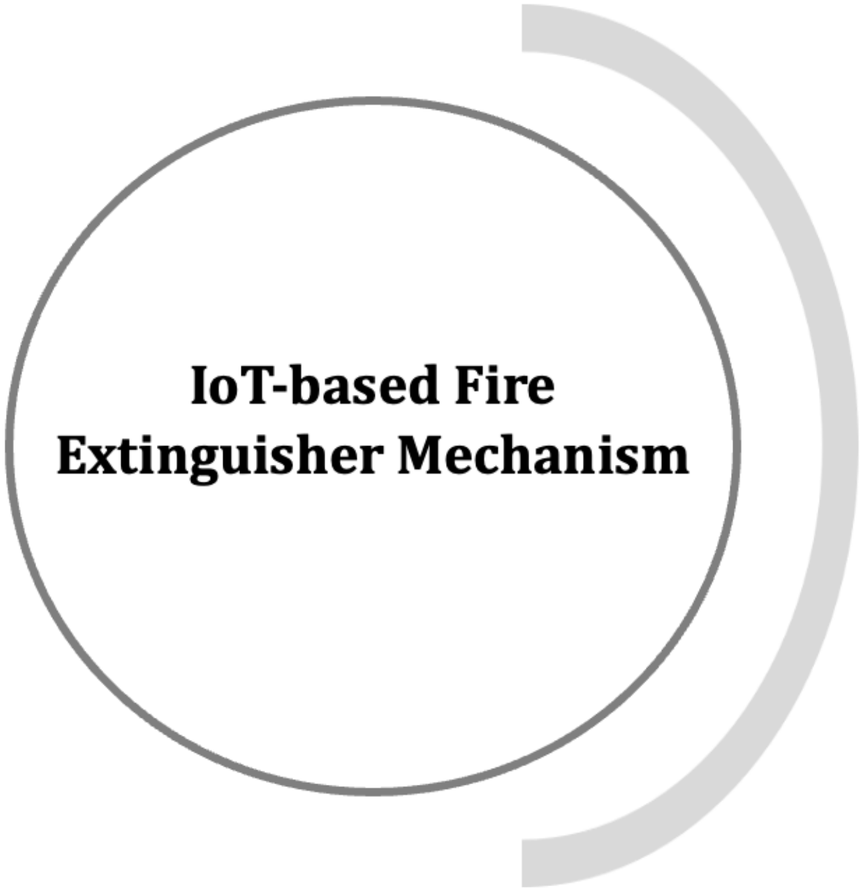
Research Problem

- How To Implement A Such System Inside A Vehicle?
- Ensure Passenger Safety ?
- Ensure Vehicle Safety?
- What Are The Social And Economic Benefits ?

Research Gap

- Installing automatic fire extinguishers into vehicles can revolutionize safety measures.
- To make the IoT-based fire extinguishing system economically viable.
- An IoT-enabled automatic fire extinguisher system significantly minimize the risk and damage.
- An Immediate action is taken, at the moment of potential fire is detected.

Objectives



- IOT device that monitor the vehicle
- Alert the driver about Fire incident
- Give the time out to evacuate the vehicle
- Based on the Response it sprays the Dry Powder

Completion Of The Component

Progress at PP1 – 50%

- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Identify Software & Hardware requirements
- Data collection
- Design System flow chart
- Design mobile app Wireframe
- Design mobile app frontend
- Learning about IOT as a beginner
- Studying IOT Tutorials
- Studying IOT sensors
- Learning about communication protocols (I2C, CAN bus)
- Background studying about micro controllers.
- Background study in sensors before buying

Completion Of The Component

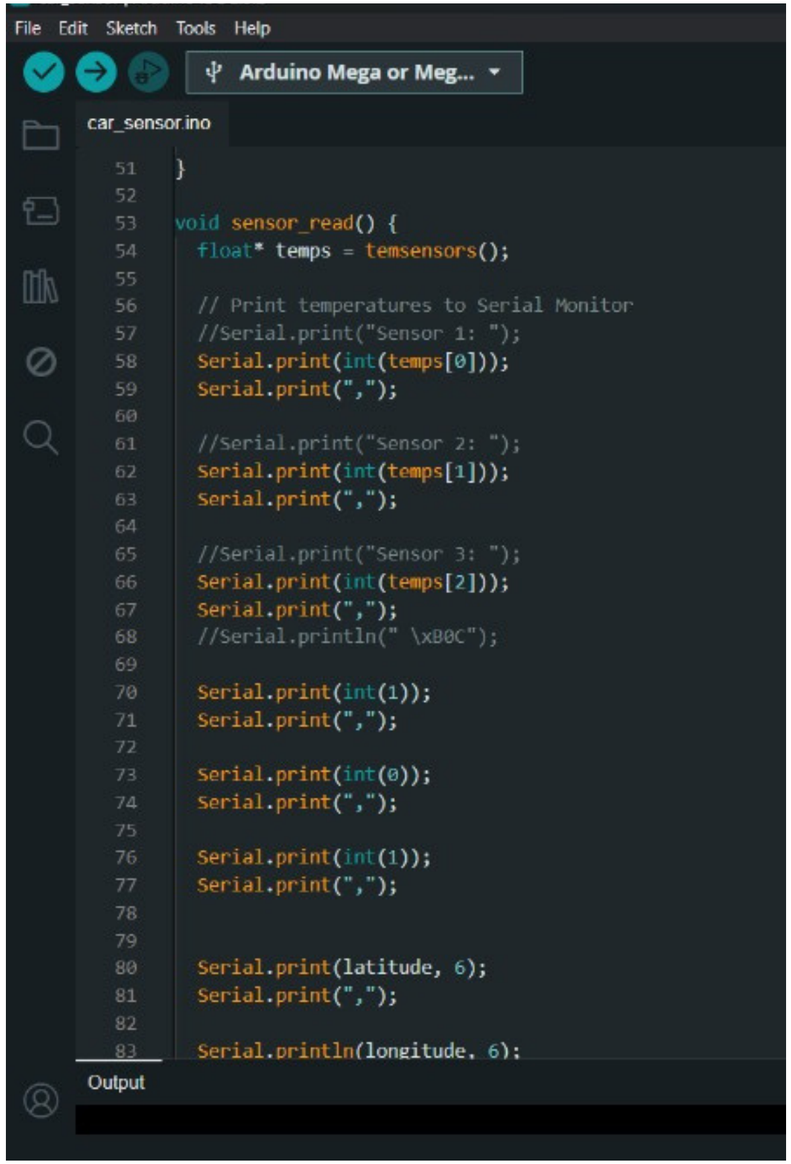
Progress at PP2 – 90%

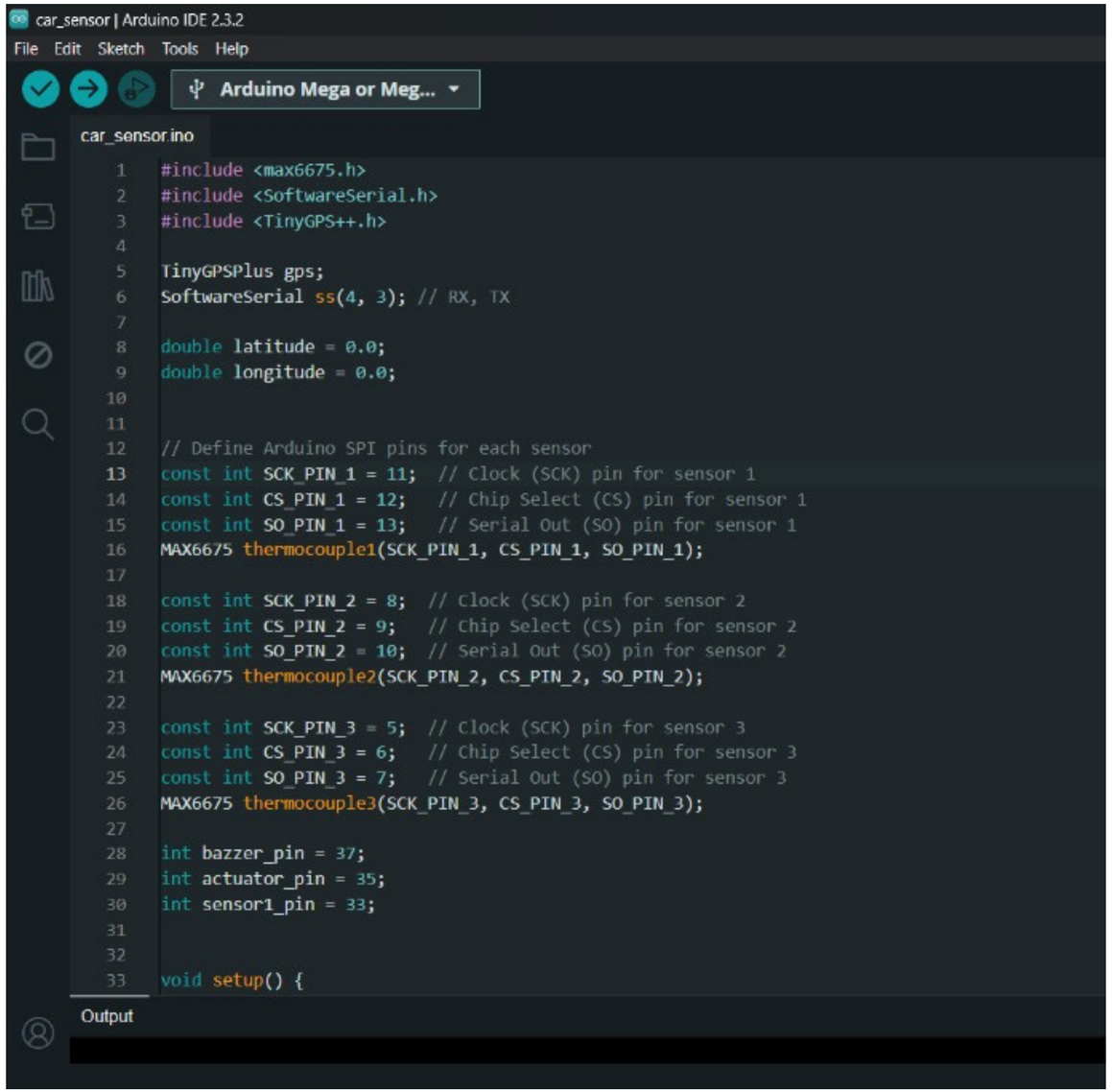
- IOT device design
- IOT device implementation
- Programing using Arduino
- Connecting the IOT device with backend
- Establishing connection for MQTT server
- Establishing connection for backend

In progress – 10%

- Testing

Implementation





Implementation

car_sensor | Arduino IDE 2.3.2

File Edit Sketch Tools Help

Arduino Mega or Meg...

car_sensor.ino

```
31
32
33  void setup() {
34
35
36    digitalWrite(bazzer_pin, LOW);
37    digitalWrite(actuator_pin, HIGH);
38
39    pinMode(bazzer_pin, OUTPUT);     // Set pin 37 as an output
40    pinMode(actuator_pin, OUTPUT);   // Set pin 35 as an output
41    pinMode(sensor1_pin, OUTPUT);    // Set pin 33 as an output
42
43    digitalWrite(bazzer_pin, LOW);
44    digitalWrite(actuator_pin, HIGH);
45
46    delay(200);
47    Serial.begin(9600);
48    Serial1.begin(9600);
49    delay(100);
50
51  }
52
53  void sensor_read() {
54    float* temps = temsensors();
55
56    // Print temperatures to Serial Monitor
57    //Serial.print("Sensor 1: ");
58    Serial.print(int(temps[0]));
59    Serial.print(",");
60
61    //Serial.print("Sensor 2: ");
62    Serial.print(int(temps[1]));
63    Serial.print(",");
```

Output

car_sensor | Arduino IDE 2.3.2

File Edit Sketch Tools Help

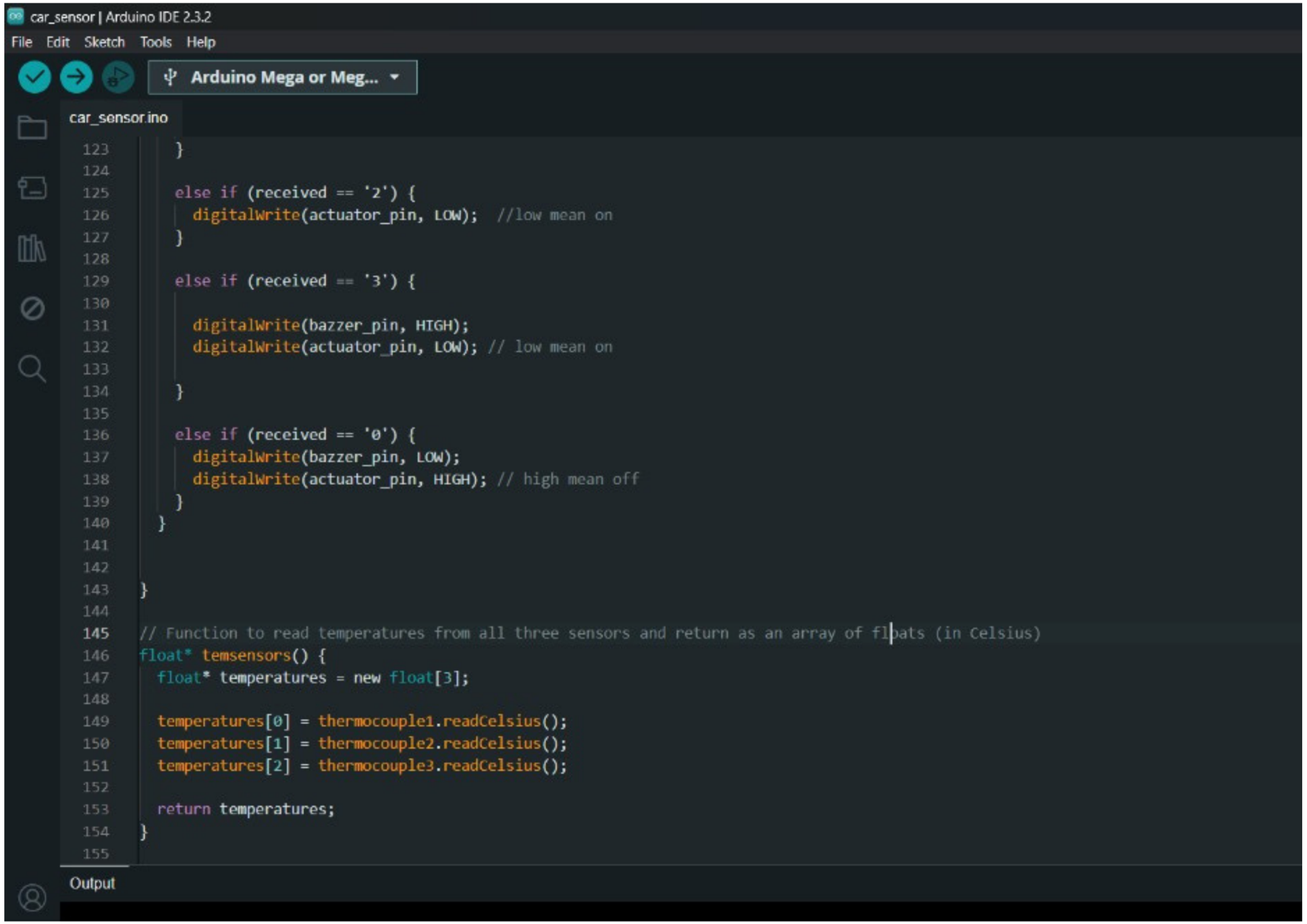
Arduino Mega or Meg...

car_sensor.ino

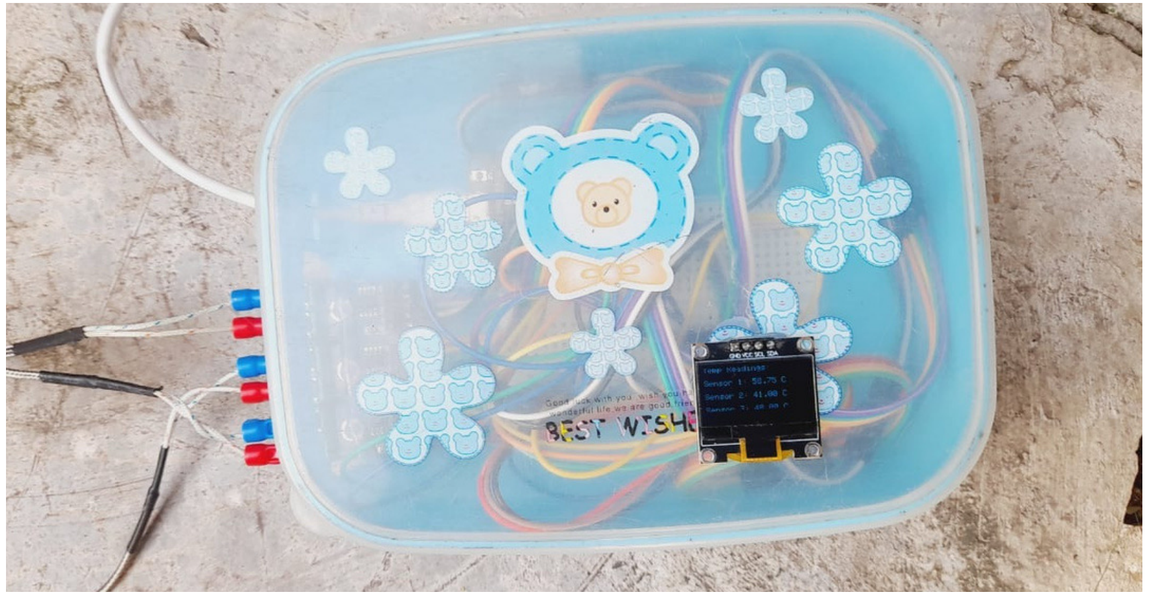
```
91  }
92
93  void gps_data() {
94    while (Serial1.available() > 0) {
95      gps.encode(Serial1.read());
96
97      if (gps.location.isUpdated()) {
98        latitude = gps.location.lat();
99        longitude = gps.location.lng();
100
101       // Serial.print("Latitude= ");
102       // Serial.println(latitude, 6);
103       // Serial.print("Longitude= ");
104       // Serial.println(longitude, 6);
105     }
106   }
107 }
108
109 void loop() {
110
111   sensor_read();
112
113   gps_data();
114
115   if (Serial.available() > 0) {  // Check if data is available to read
116
117     char received = Serial.read();  // Read the incoming byte
118
119     // Serial.println(received);
120
121     if (received == '1') {
122       digitalWrite(bazzer_pin, HIGH);
```

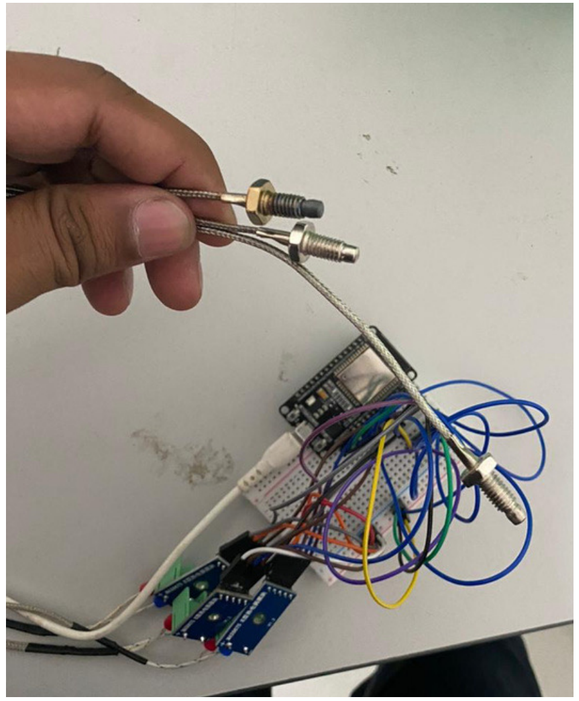
Output

Implementation

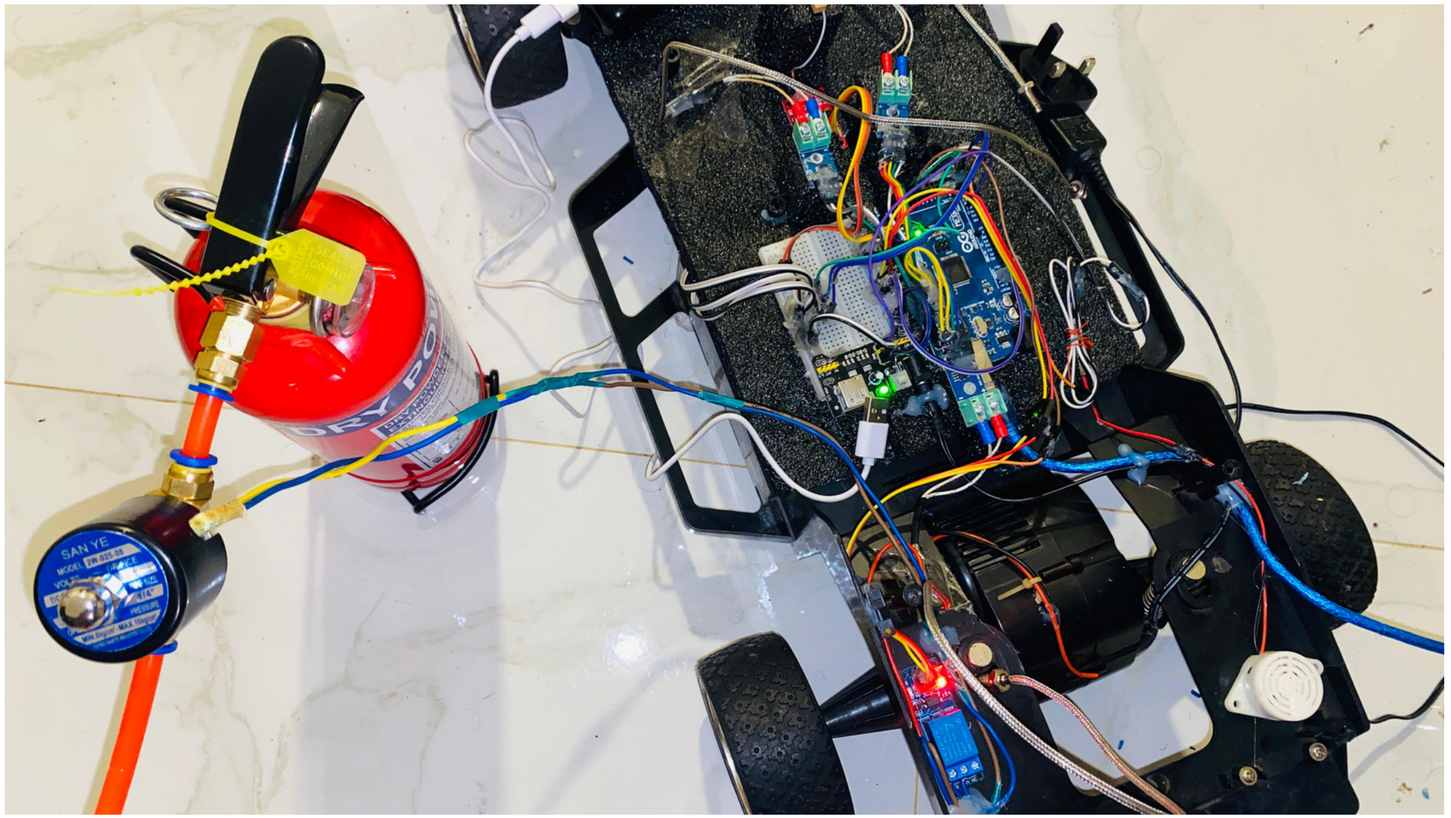


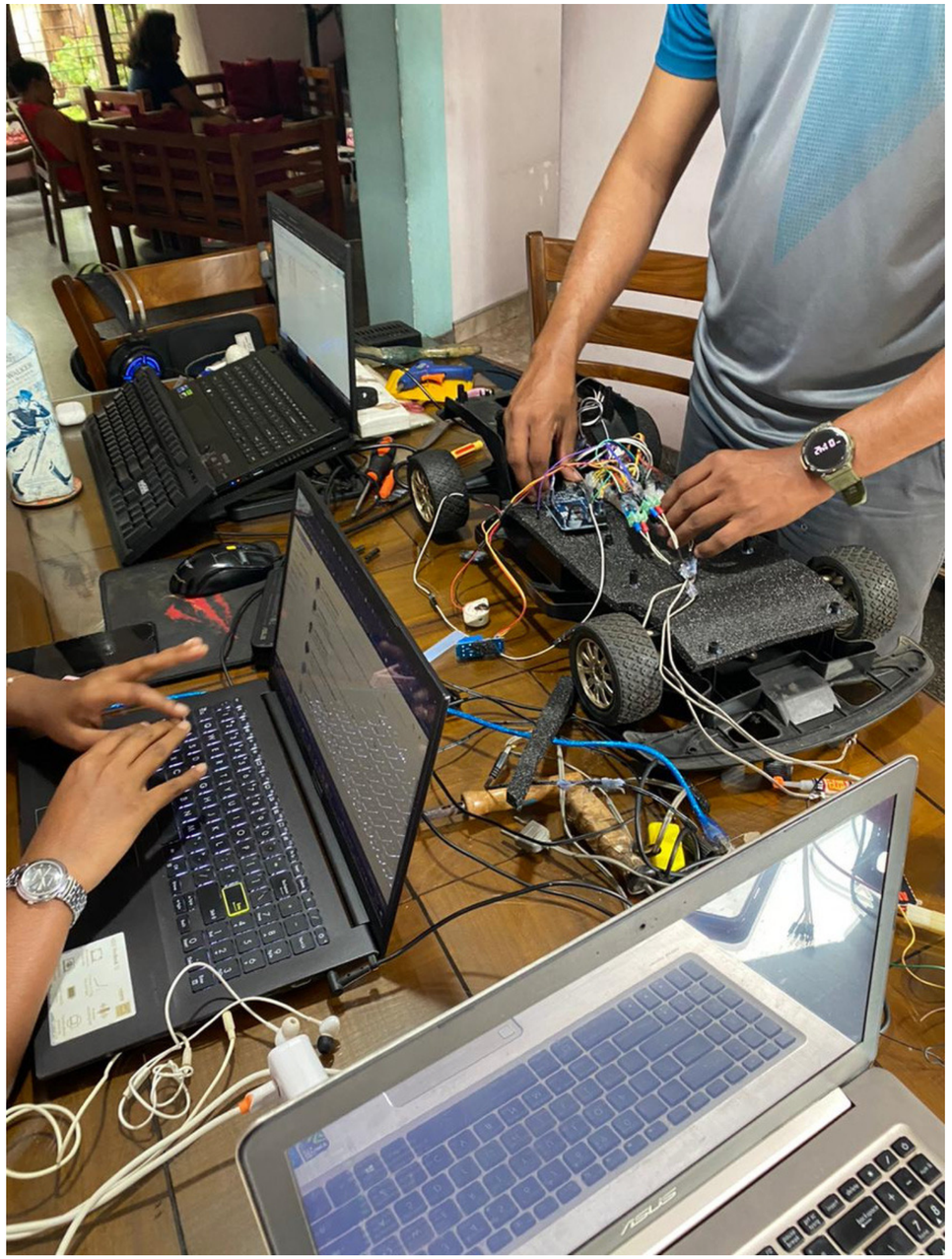
Implementation



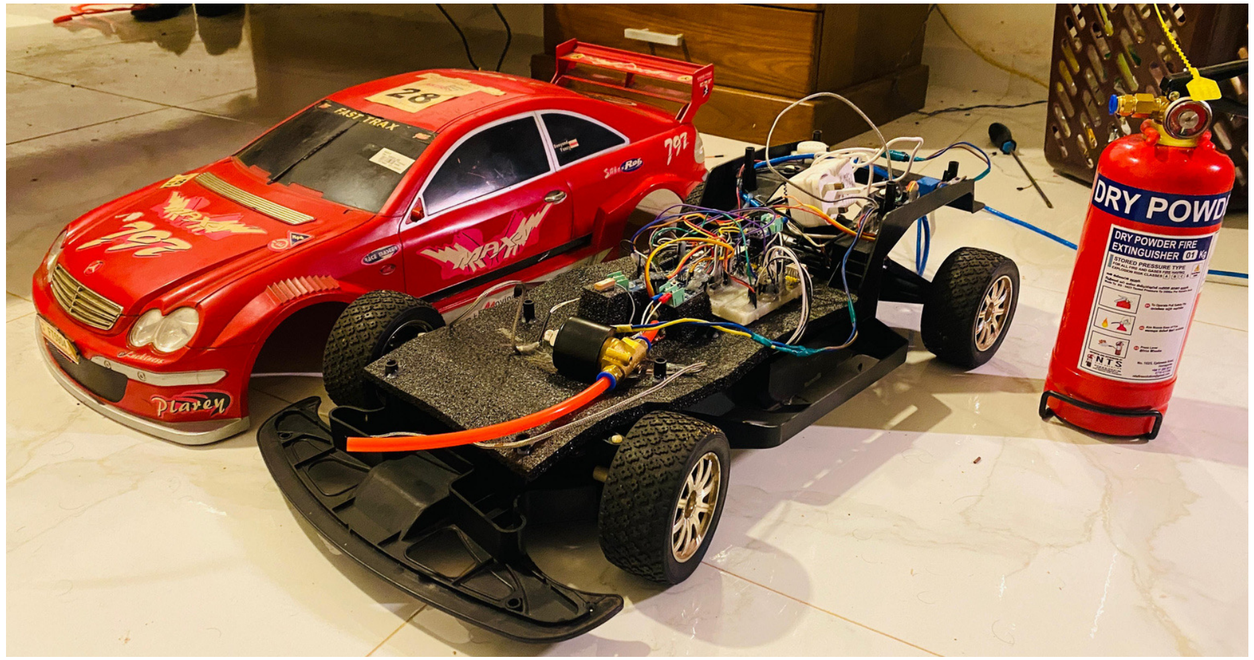


Implementation





Implementation



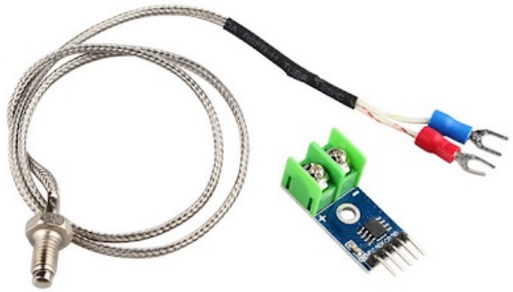
Tools and Technologies

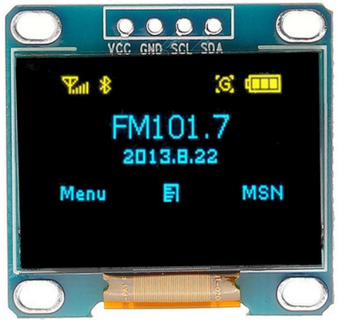
Hardware Tools

- ESP32
- Arduino mega board
- Arduino UNO board
- max6675 K type thermocouple
- Temperature Sensor
- Mini Ublox neo 6m gps module
- Alarm Buzzer
- OLED 128X64 0.96 inch Display Module
- 12V Power supply
- Bread board
- Bread board power supply
- Fire Extinguisher
- Solenoid valve
- 1 channel relay module

Software Tools

- Micro Python
- c/c++
- Arduino IDE
- Pycharm
- Python





Requirement Analysis

Functional Requirements

- The system must transmit the collected data securely to a cloud.
- The system must provide real-time feedback to drivers based on the situation.
- The system must include a user-friendly interface.
- The system must be capable of integrating with existing vehicle systems.
- The Fire extinguishers must be easy to install without modifying

Non-Functional Requirements

- The system must be reliable with minimal downtime.
- The solution should be cost-effective
- The system must process and analyze data in real-time.
- The system should be easy to maintain.

References

[1] Habib, M. R., Khan, N., Ahmed, K., Kiran, M. R., Asif, A., Bhuiyan, M. I., & Farrok, O. (2019, September). Quick Fire Sensing Model and Extinguishing by Using an Arduino Based Fire Protection Device. 2019 5th International Conference on Advances in Electrical Engineering (ICAEE). https://doi.org/10.1109/icaee48663.2019.8975538

[2] Kumar, D. D., Bharathraj, B., Vishak, V. N., Jasith, S., & Raja, L. (2023, March 23). IoT Based Fire Protection System. 2023 4th International Conference on Signal Processing and Communication (ICSPC). https://doi.org/10.1109/icspc57692.2023.10125807.

[3] Mathavan, J. J., Faslan, A., Basith, N. U. A., & Wanigasinghe, W. (2020, June). Hardware Implementation of Fire Detection, Control and Automatic Door Unlocking System for Automobiles. 2020 4th International Conference on Trends in Electronics and Informatics (ICOEI)(48184). https://doi.org/10.1109/icoei48184.2020.9142990

[4] P, C., Venusamy, K., S, N., EL, J., & Vickyath, S. (2023, May 4). Design and Implementation of IoT based Multi Degree Rotating Fire Extinguisher System. 2023 2nd International Conference on Applied Artificial Intelligence and Computing (ICAAIC). https://doi.org/10.1109/icaaic56838.2023.10140798

Anthick.G.N| IT21096266
Specializing in Information Technology

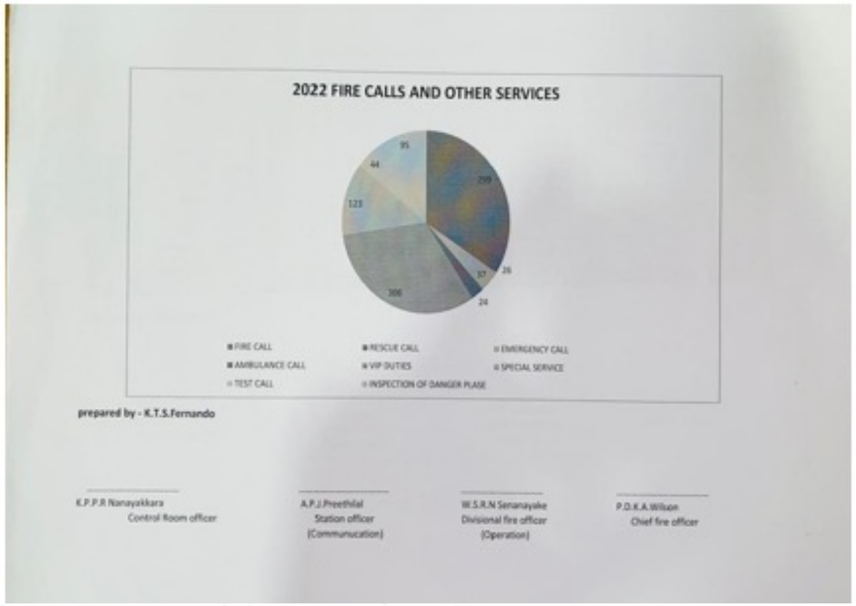
Optimizing Emergency Response Paths with Resource Allocation

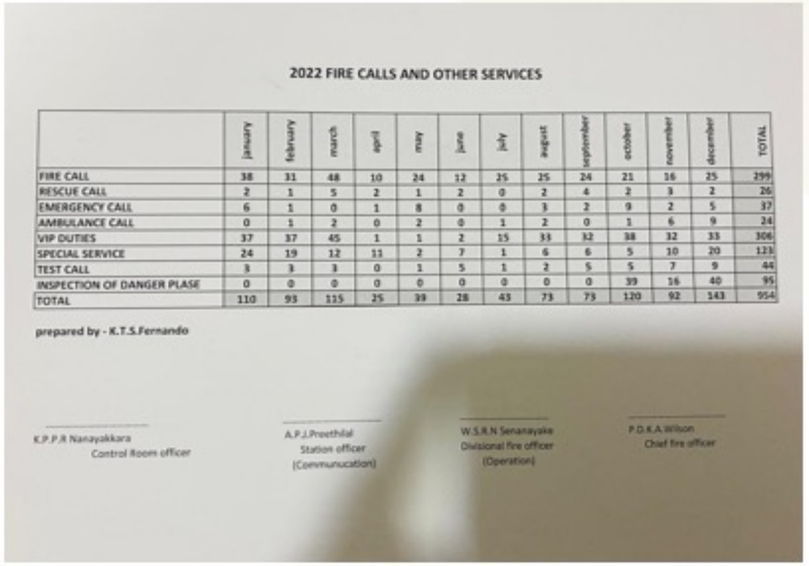
Research Problem

- Inefficiency of Manual Call Verification
- Lack of Real-Time Response Information
- Caller Helplessness During Emergencies

Introduction

Research Problem





2022 FIRE CALLS AND OTHER SERVICES

	january	febrsury	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	79	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

Research Gap

•Essential for reducing response times in vehicle fire emergencies.

•Absence of real-time support for both emergency responders and vehicle owners, hindering prompt responses.

•Vehicle owners lack immediate access to crucial emergency support like nearby fire extinguishers. This research aims to address these gaps by implementing a comprehensive, integrated solution.

Objectives

Enhance emergency response times for vehicle fires a system that facilitates real-time data sharing between fire departments and vehicle owners.

- Develop an automated system to detect and report vehicle fire locations to the cloud.
- Implement an algorithm to automatically identify and notify the nearest fire department with details of the incident.
- Create an model to inform vehicle owners about nearby fire.

Implementation

Flask Server

current.json | station_main.py | server.py | TestScreen.js | GPS_calculations.py

Backend > server.py > real_station

```python
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)
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS

```
    self._target(*self._args, **self._kwargs)
  File "C:\Users\ACER NITRO\Desktop\Station\Backend\server.py", line 125, in send_new_station_periodically
    current_data = json.load(f)
  File "C:\Users\ACER NITRO\AppData\Local\Programs\Python\Python310\lib\json\__init__.py", line 293, in load
    return loads(fp.read(),
  File "C:\Users\ACER NITRO\AppData\Local\Programs\Python\Python310\lib\json\__init__.py", line 346, in loads
    return _default_decoder.decode(s)
  File "C:\Users\ACER NITRO\AppData\Local\Programs\Python\Python310\lib\json\decoder.py", line 337, in decode
    obj, end = self.raw_decode(s, idx=_w(s, 0).end())
  File "C:\Users\ACER NITRO\AppData\Local\Programs\Python\Python310\lib\json\decoder.py", line 355, in raw_decode
    raise JSONDecodeError("Expecting value", s, err.value) from None
json.decoder.JSONDecodeError: Expecting value: line 1 column 1 (char 0)
```

Frontend Code

frontend > src > components > screencomponents > TestScreen > TestScreen.js > TestScreen > map() callback

```javascript
import React, { useState, useEffect, useRef } from "react";
import {FaPhone, FaMapMarkerAlt, FaCar, FaFire, FaRegCopy,} from "react-icons/fa";
import Swal from "sweetalert2"; // Import SweetAlert2
import { io } from "socket.io-client";
import { collection, getDocs, onSnapshot, doc, updateDoc } from "firebase/firestore"; // Import Firestore functions
import { db } from "../../../firebase/firebase.js";
import { GoogleMap, LoadScript, Marker } from '@react-google-maps/api';

import "./TestScreen.css";

// const socket = io("http://localhost:5000"); // Adjust the URL as needed

const TestScreen = () => {
  const [selectedStation, setSelectedStation] = useState(null);
  const [actionedStations, setActionedStations] = useState([]);
  const [currentStations, setCurrentStations] = useState([]);
  const [vehiclesLeft, setVehiclesLeft] = useState([]);
  const [tab, setTab] = useState("current");
  const [loading, setLoading] = useState(false);
  const [cardsLoading, setCardsLoading] = useState(true);
  const userInteractedRef = useRef(null);

  const newItemRef = useRef(null);
  const intervalRef = useRef(null);

  useEffect(() => {
    // Add an event listener to capture user interaction
    const handleUserInteraction = () => {
      console.log("has is done");
      userInteractedRef.current = true;
      // Remove the event listener after the first interaction
      document.removeEventListener("click", handleUserInteraction);
      document.removeEventListener("keydown", handleUserInteraction);
      document.removeEventListener("scroll", handleUserInteraction);
    };

    // Attach the event listeners
    document.addEventListener("click", handleUserInteraction);
    document.addEventListener("keydown", handleUserInteraction);
    document.addEventListener("scroll", handleUserInteraction);
```

Implementation

```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

```

Calculating Nearest Extinguishers

```
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: 161

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: 139

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: 102

Extinguisher Details:
no: 4
name: Pizza Hut - Kothalawala
Latitude: 6.916317544
Longitude: 79.97236758
water: y
```

Implementation

Extinguishers near SLIIT 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

Implementation

Calculating Nearest Fire sub station

Backend > GPS_calculations.py > find_optimal_route

```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_ZSQQ7qESG6TPvKviBf0XUIlIcGd84I4'



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
```

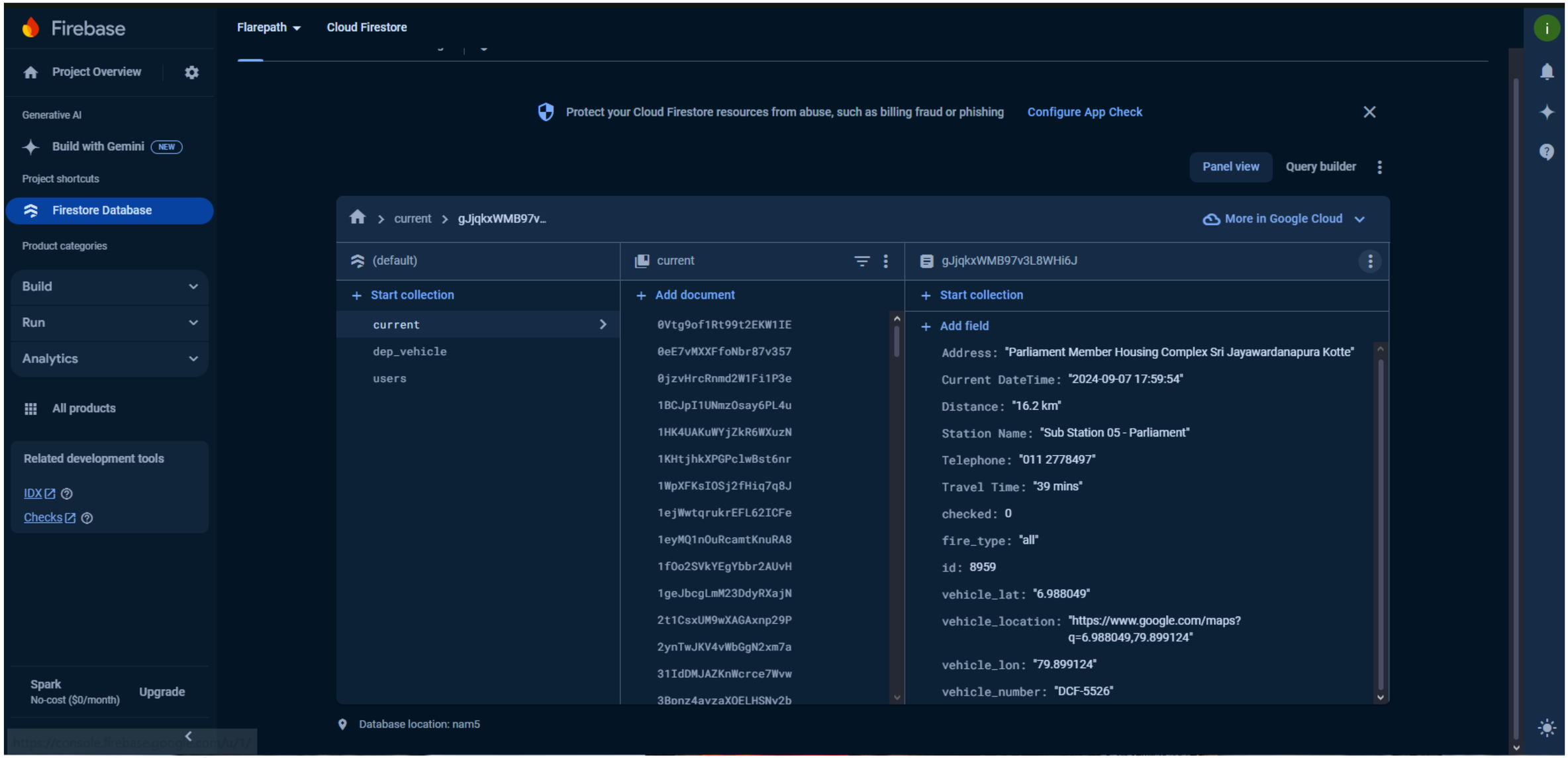
Implementation

Nearest Fire sub station Output

```
PS C:\Users\ACER NITRO\Desktop\Station\Backend> python GPS_calculations_main.py
Station: Head Quarters - Maradana, Distance: 2185m, Duration: 354s
Combined Weight for Head Quarters - Maradana: 1635.7
Station: Sub Station 01 - Hettiyawaththa, Distance: 4643m, Duration: 692s
Combined Weight for Sub Station 01 - Hettiyawaththa: 3457.7
Station: Sub Station 02 - Gaspaha, Distance: 3089m, Duration: 469s
Combined Weight for Sub Station 02 - Gaspaha: 2302.9999999999995
Station: Sub Station 03 - Wellawaththa, Distance: 8075m, Duration: 1045s
Combined Weight for Sub Station 03 - Wellawaththa: 5966.0
Station: Sub Station 04 - Pettah, Distance: 5535m, Duration: 776s
Combined Weight for Sub Station 04 - Pettah: 4107.299999999999
Station: Sub Station 05 - Parliament, Distance: 12925m, Duration: 1494s
Combined Weight for Sub Station 05 - Parliament: 9495.7

Shortest path by combined weight (in order):
Station: Head Quarters - Maradana, Combined Weight: 1635.7
Station: Sub Station 01 - Hettiyawaththa, Combined Weight: 3457.7
Station: Sub Station 02 - Gaspaha, Combined Weight: 2302.9999999999995
Station: Sub Station 03 - Wellawaththa, Combined Weight: 5966.0
Station: Sub Station 04 - Pettah, Combined Weight: 4107.299999999999
Station: Sub Station 05 - Parliament, Combined Weight: 9495.7

Nearest Station Info: {'Station_Name': 'Head Quarters - Maradana', 'Distance': 2185, 'Travel_Time': 354, 'Address': 'T.B. Jaya Mawatha, Colombo 10', 'Telephone': '011-4222222', '
Current_DateTime': '2024-09-08 21:19:04'}
PS C:\Users\ACER NITRO\Desktop\Station\Backend>
```

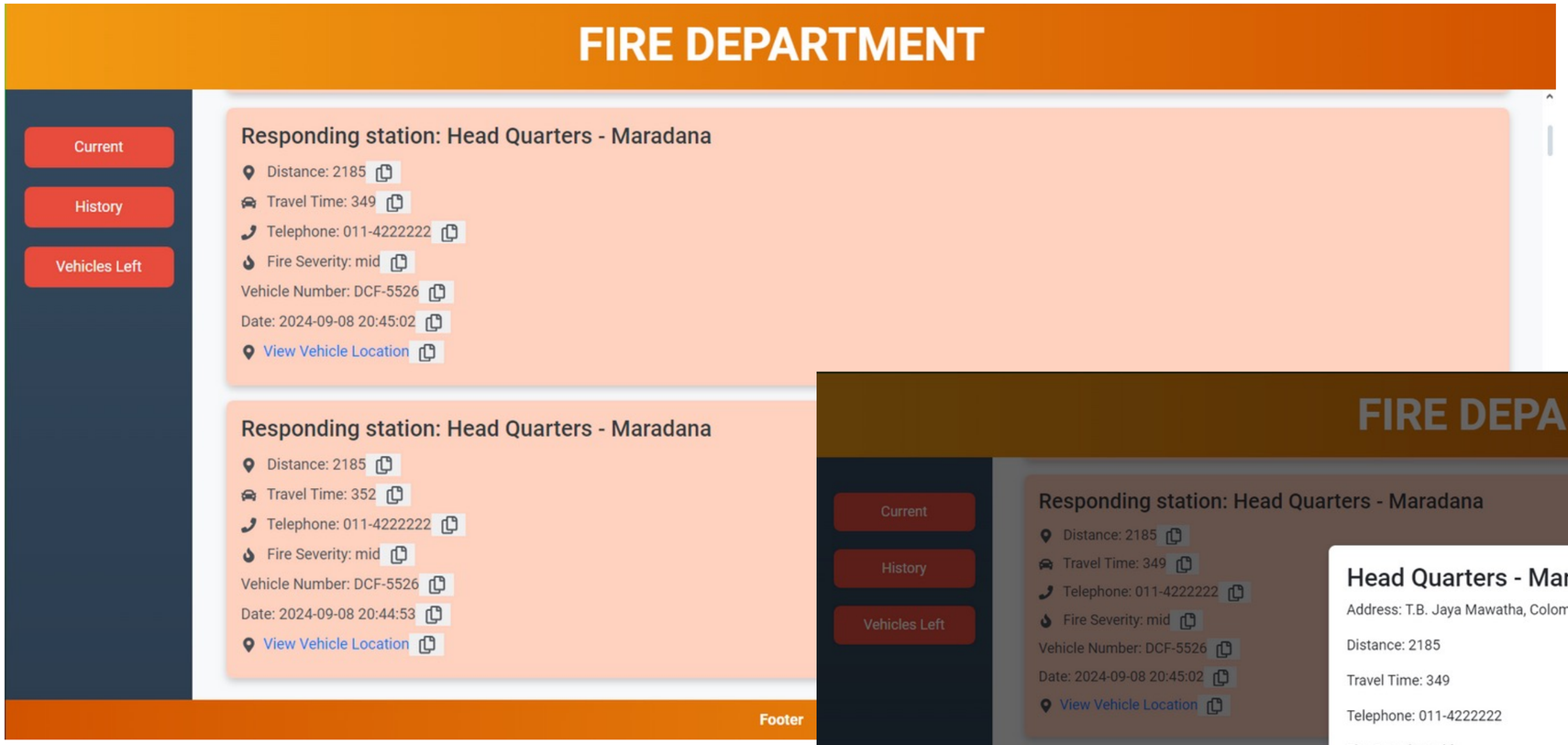
Implementation

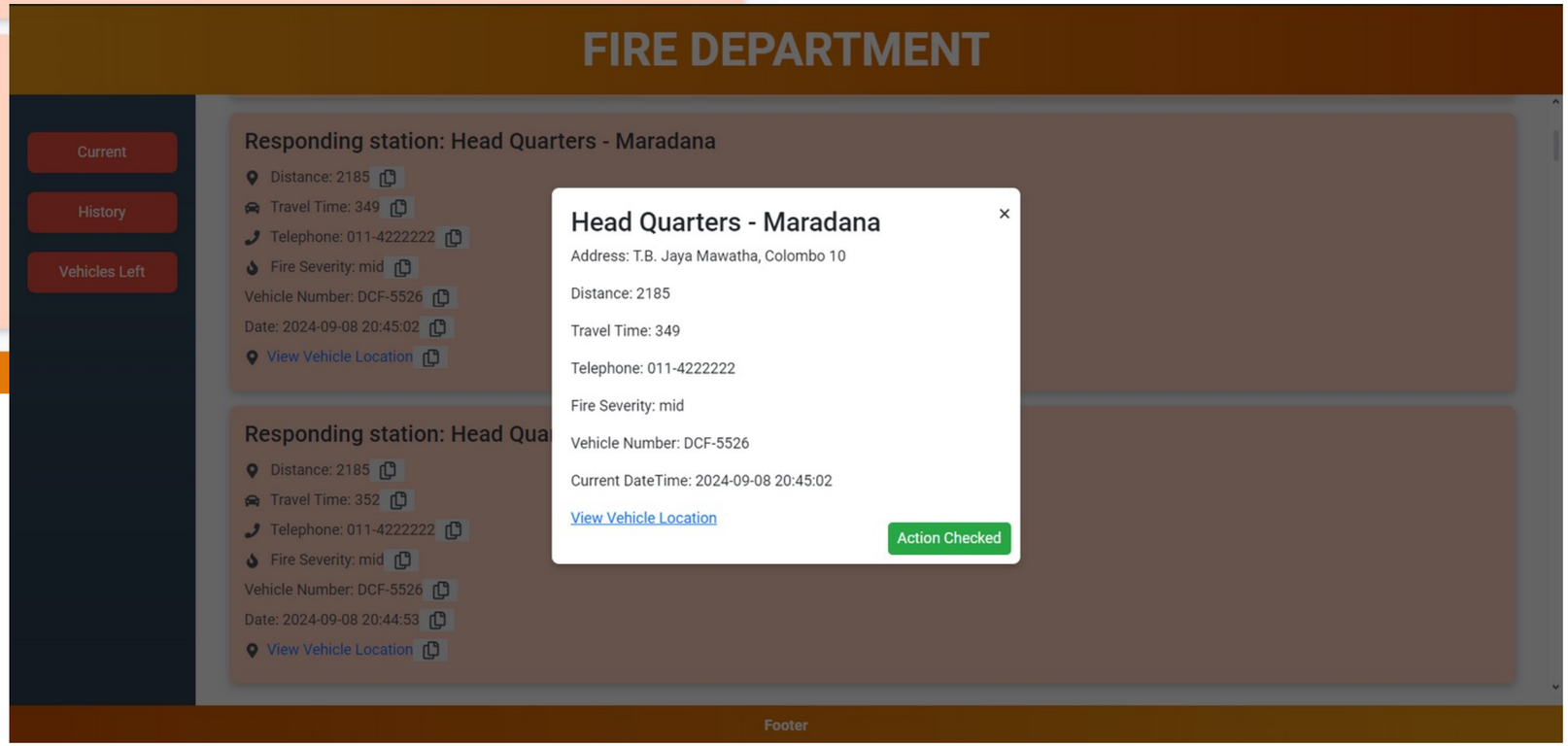
Firebase (Firestore Database)



Implementation

Dashboard for Fire Department





Completion Of The Component

Progress at PP1 – 50%

- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Identify Software requirements
- Collect data manually from RDA and create dataset
- Collect data manually from Fire department and create dataset
- Design System flow chart
- Design Dashboard, Active Incidents, Active Incidents view and Logs wireframe
- Design Dashboard, active Incidents, active Incidents view and logs frontend design

Completion Of The Component

Progress at PP2 – 90%

- Collect data manually about extinguishers near the SLIIT
- Nearest fire station calculating
- Calculating extinguishers within 300m radius
- Creating the backend of the ‘Server’
- Connecting MQTT server and the backend and done testing
- Creating the frontend for the Fire Department
- Testing, Validation and dashboard completion of the frontend
- UI/UX improvements for mobile app and web app

In progress – 10%

- Connecting “Extinguishers within the vicinity” to the mobile app

Tools and Technologies

Software Technologies

- Python
- Pandas , NumPy
- Flask
- Firebase
- networkx
- NodeJS
- React Native

Requirement Analysis

Functional Requirements

- Detect the location of a vehicle on fire and transmit this data
- Identify and notify the nearest fire department about the incident, including sending detailed information about the location and severity of the fire.
- Display the available emergency resources, including fire trucks and their operational status.
- Provide real-time updates to vehicle owners about the proximity and availability of fire extinguishers and other emergency resources.
- Ensure system compatibility with emergency dispatch protocols.

Non-Functional Requirements

- Interfaces should be User-friendly
- Application should be reliable
- Application should be able to give fast results

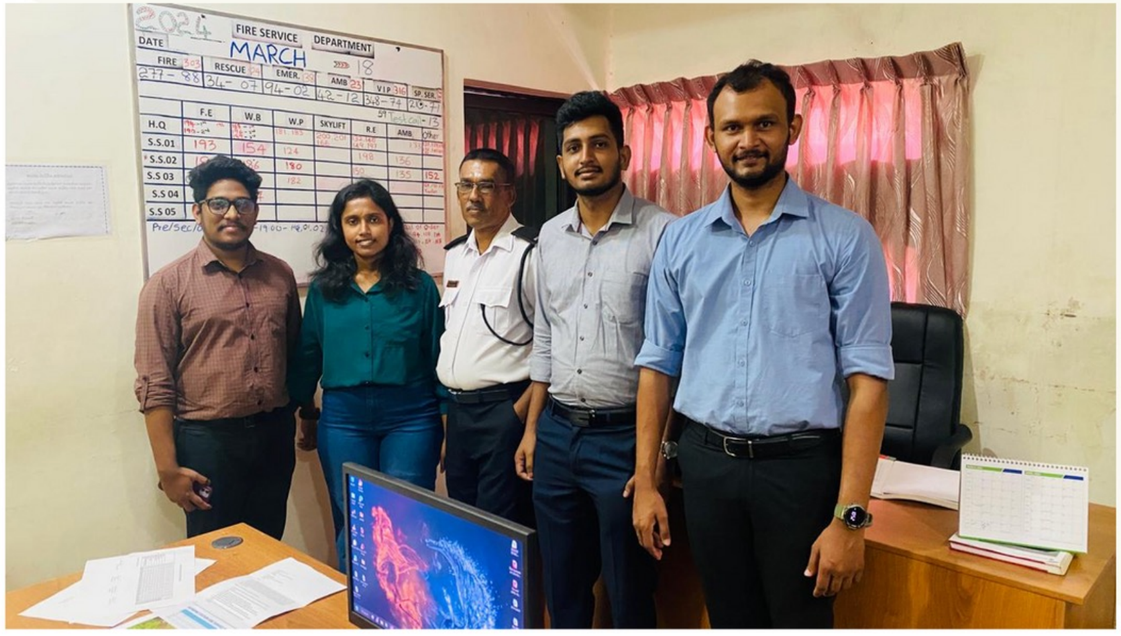
References

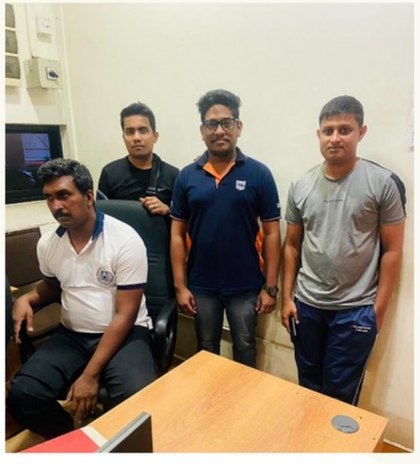
[1] J. Zhang et al., “Vehicle routing in urban areas based on the Oil Consumption Weight -Dijkstra algorithm,” IET Intelligent Transport Systems, vol. 10, no. 7, pp. 495–502, Sep. 2016, doi: 10.1049/iet-its.2015.0168.

[2] A. Candra, M. A. Budiman, and K. Hartanto, “Dijkstra’s and A-Star in Finding the Shortest Path: a Tutorial,” 2020 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA), Jul. 2020, Published, doi: 10.1109/databia50434.2020.9190342.

[3] D. Fan and P. Shi, “Improvement of Dijkstra’s algorithm and its application in route planning,” 2010 Seventh International Conference on Fuzzy Systems and Knowledge Discovery, Aug. 2010, Published, doi: 10.1109/fskd.2010.5569452.

SNAPS FROM THE FIELD VISITS



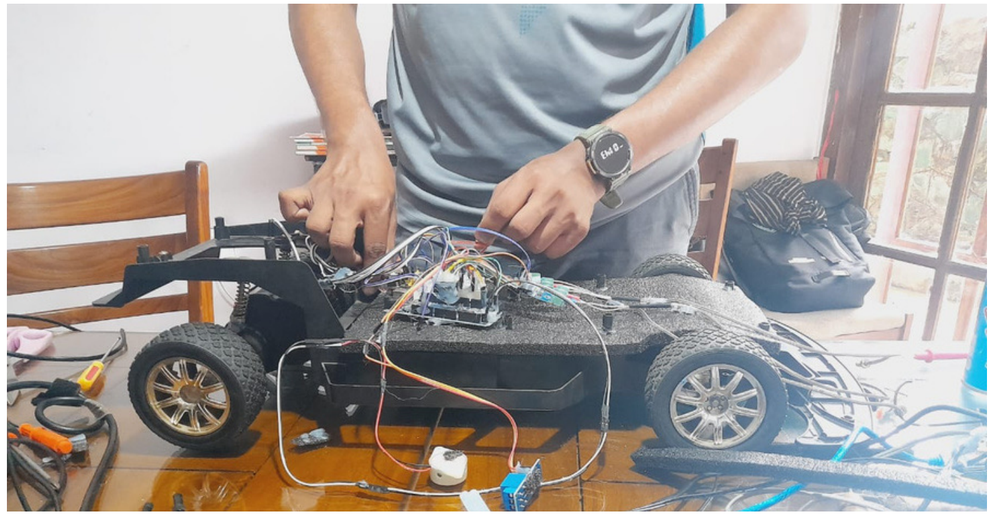


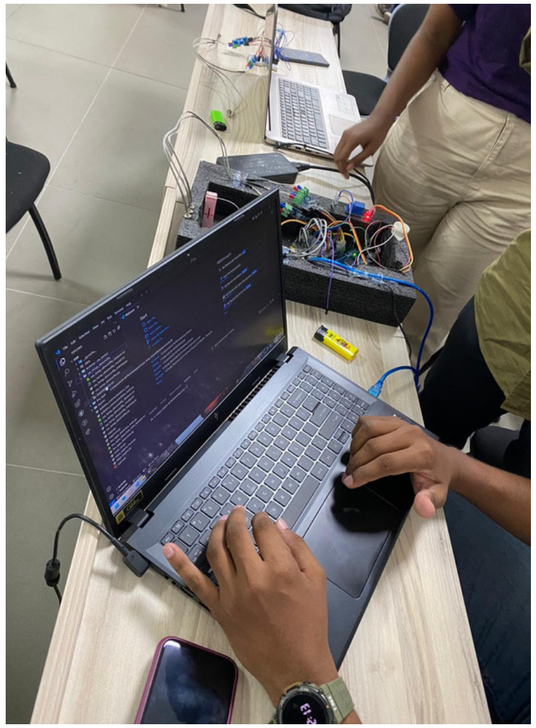
SLIIT
FACULTY OF COMPUTING

SNAPS FROM THE FIELD VISITS

SLIIT
FACULTY OF COMPUTING

SLIIT
FACULTY OF COMPUTING





SLIIT
FACULTY OF COMPUTING

Commercialization

- Collaborate with automobile manufacturers to install the system directly into new vehicles.

- Utilize collected data on fire events and responses to provide consultancy services to automobile manufacturers, assisting them in designing safer vehicles.

Thank You !