FlarePath

Advanced Vehicle Fire Safety and Monitoring with Rapid Emergency Dispatch Solutions

R24-058

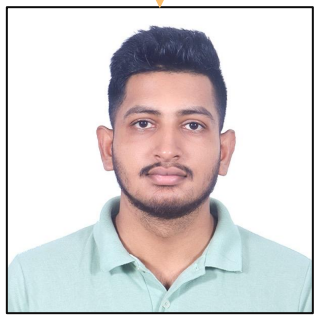
SLIIT
FACULTY OF COMPUTING

Our Team

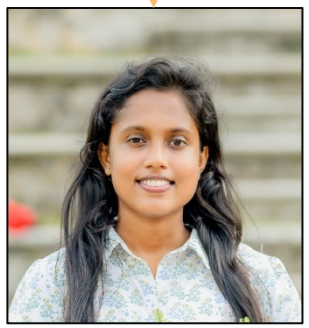
MR.NELUM CHATHURANGA AMARASENA
SUPERVISOR

MR.DEEMANTHA NAYANJITH SIRIWARDANA
CO-SUPERVISOR

MR. W. D. ONRAY SAHINDA
EXTERNAL-SUPERVISOR



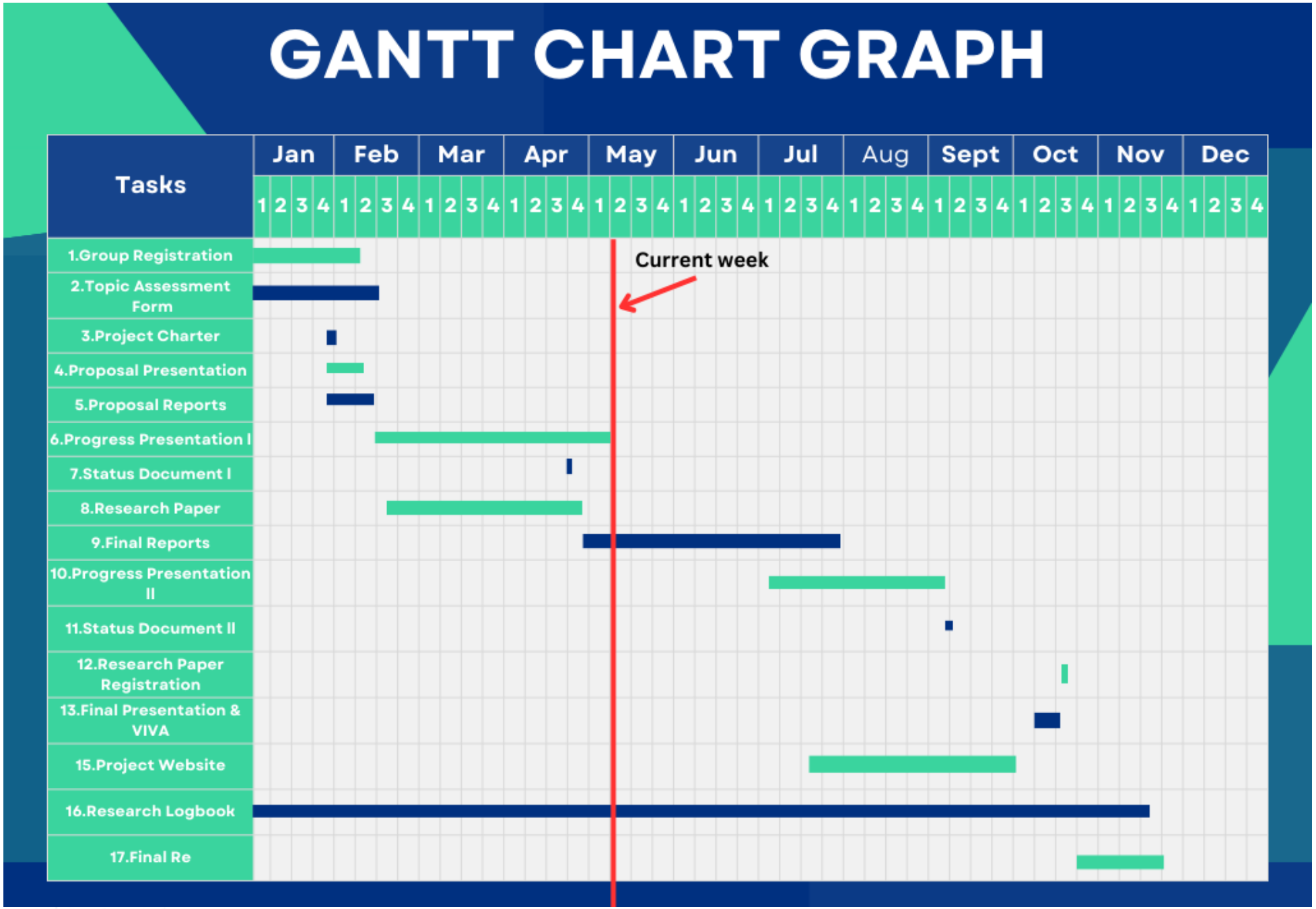
PERAMUNAGE A.N
IT21080562



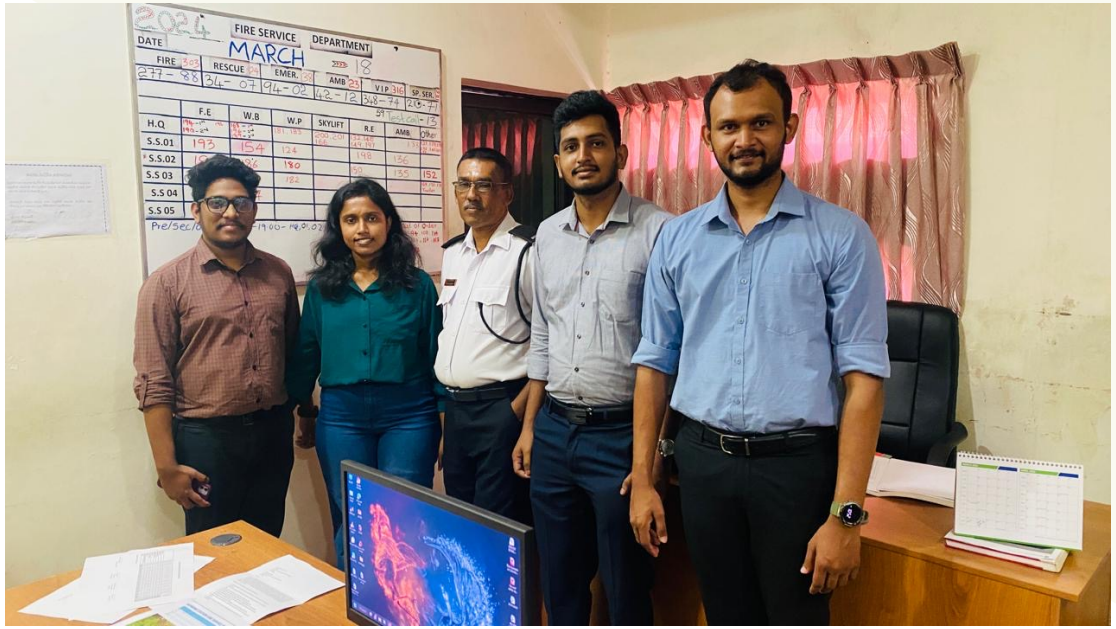
ABEYWARDHANA D.N
IT21133718

DHARMAGUNAWARDANA W.M.P.I
IT21132346

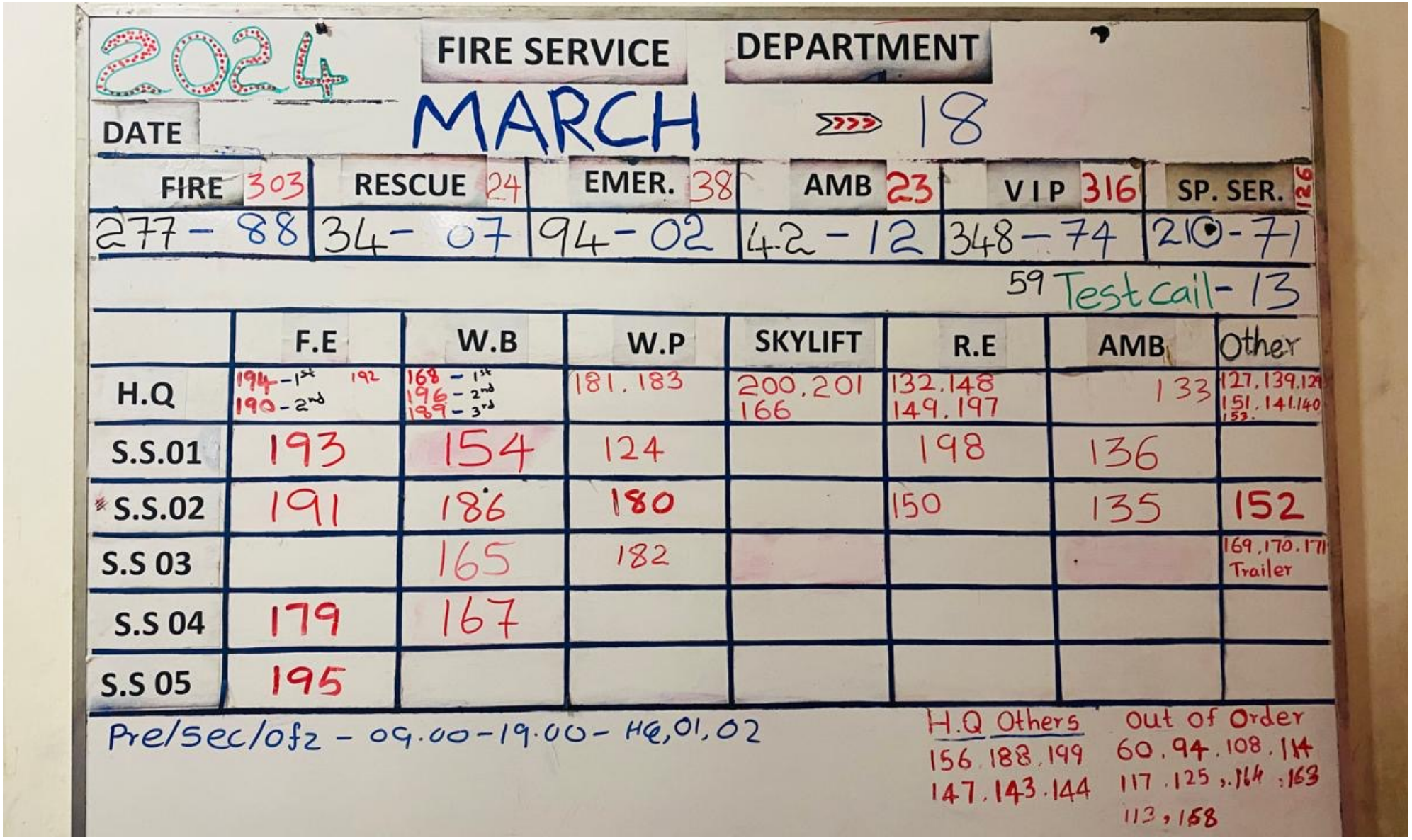
ANTHICK G.N
IT21096266


GANTT CHART GRAPH
Tasks
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Dec
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Current week
1.Group Registration
2.Topic Assessment Form
3.Project Charter
4.Proposal Presentation
5.Proposal Reports
6.Progress Presentation I
7.Status Document I
8.Research Paper
9.Final Reports
10.Progress Presentation II
11.Status Document II
12.Research Paper Registration
13.Final Presentation & VIVA
15.Project Website
16.Research Logbook
17.Final Re

SNAPS FROM THE FIELD VISITS

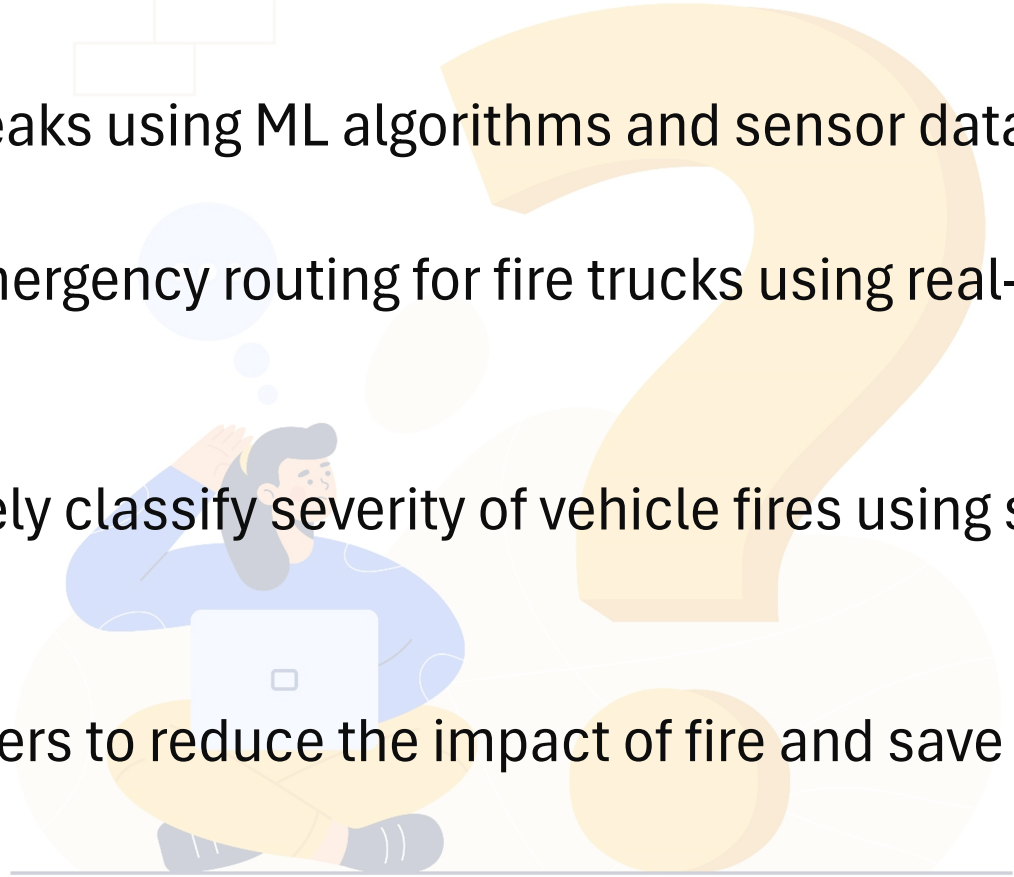


SNAPS FROM THE FIELD VISITS

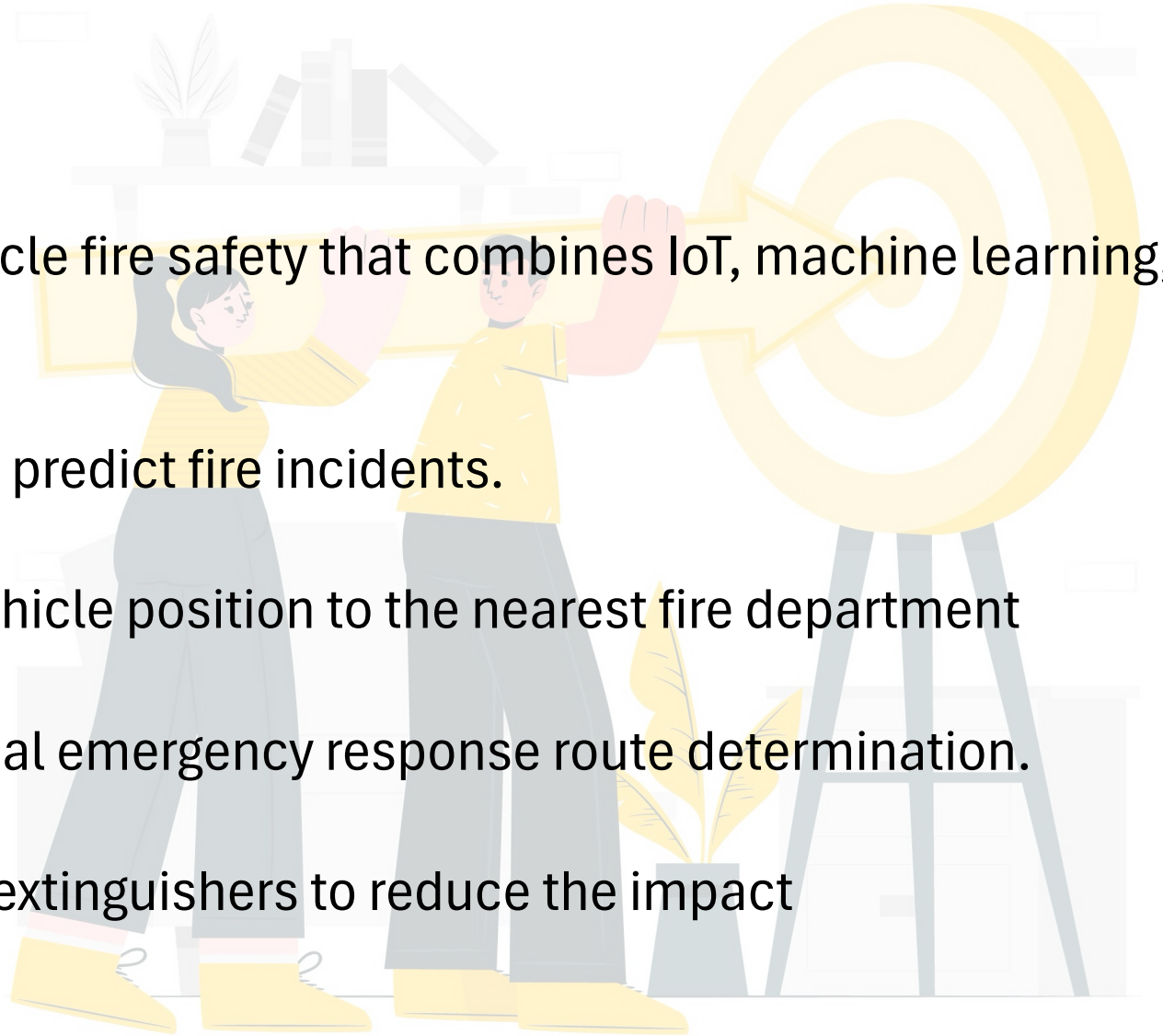

2024
FIRE SERVICE DEPARTMENT
DATE
MARCH
18
FIRE 303
RESCUE 24
EMER. 38
AMB 23
VIP 316
SP. SER. 126
277 - 88
34 - 07
94 - 02
42 - 12
348 - 74
210 - 71
59 Test call - 13
F.E
W.B
W.P
SKYLIFT
R.E
AMB
Other
H.Q
194 - 1st 192
190 - 2nd
168 - 1st
196 - 2nd
189 - 3rd
181, 183
200, 201
166
132, 148
149, 197
133
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?

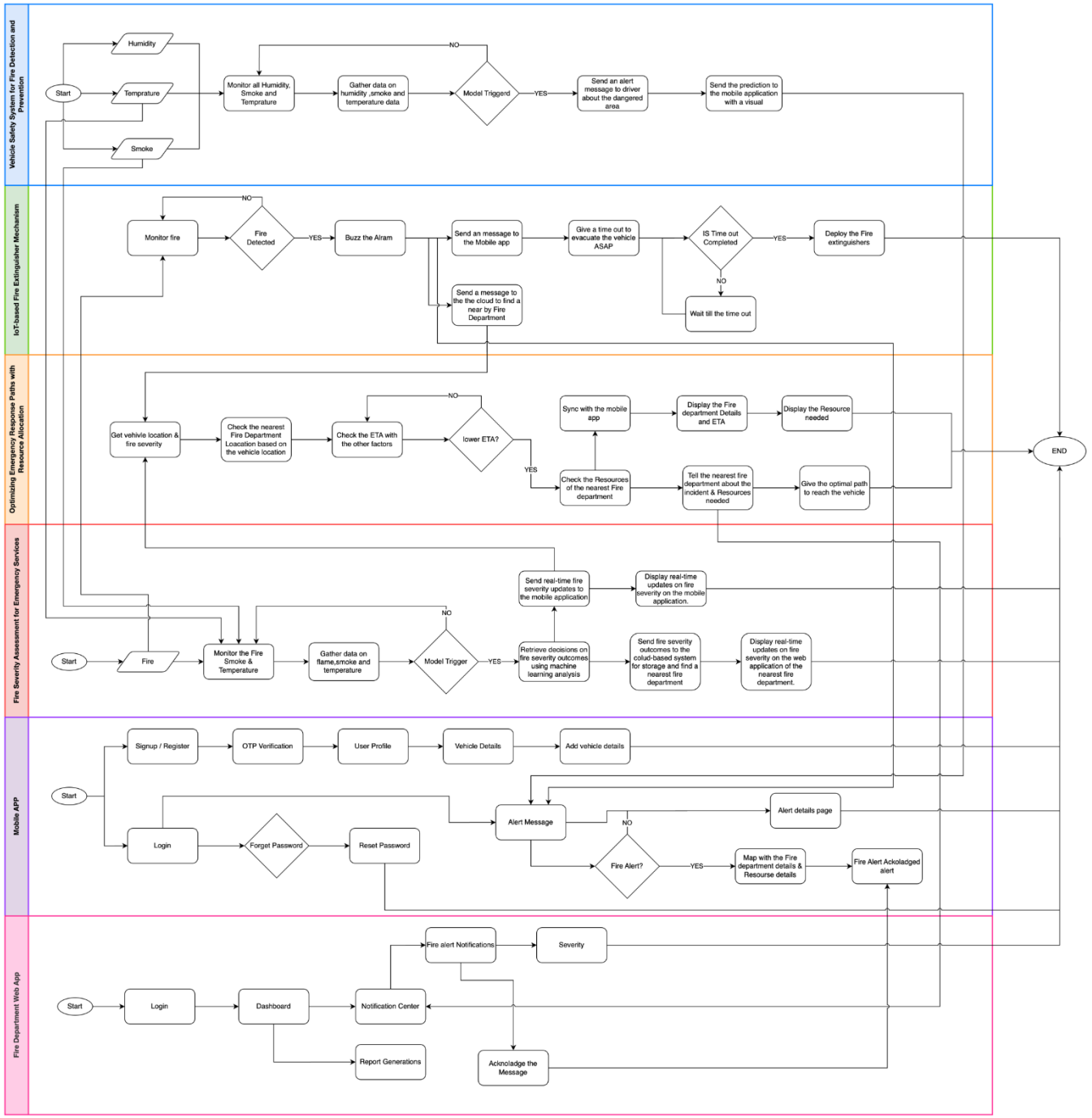


Objectives



- 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

Overall Diagram



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.

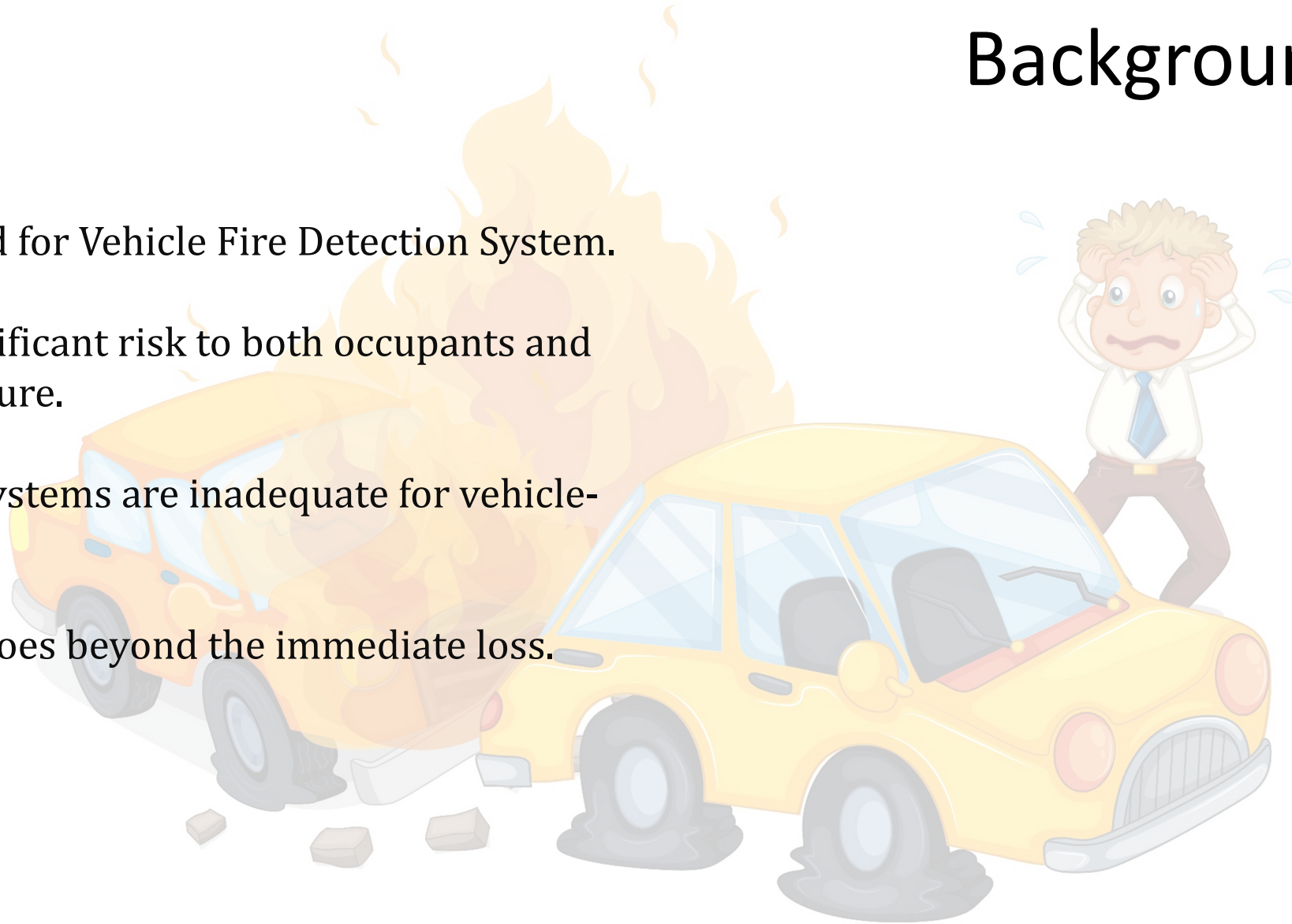
Individual Components

Peramunage A.N | IT21080562
Specializing in Information Technology

Vehicle Safety System for Fire Detection and Prevention

Background

- Understanding the Need for Vehicle Fire Detection System.
- Vehicle fires pose a significant risk to both occupants and surrounding infrastructure.
- Current fire detection systems are inadequate for vehicle-specific scenarios.
- Impact of vehicle fires goes beyond the immediate loss.



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 smoke, temperature, fire, and vehicle vibrations.

- **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

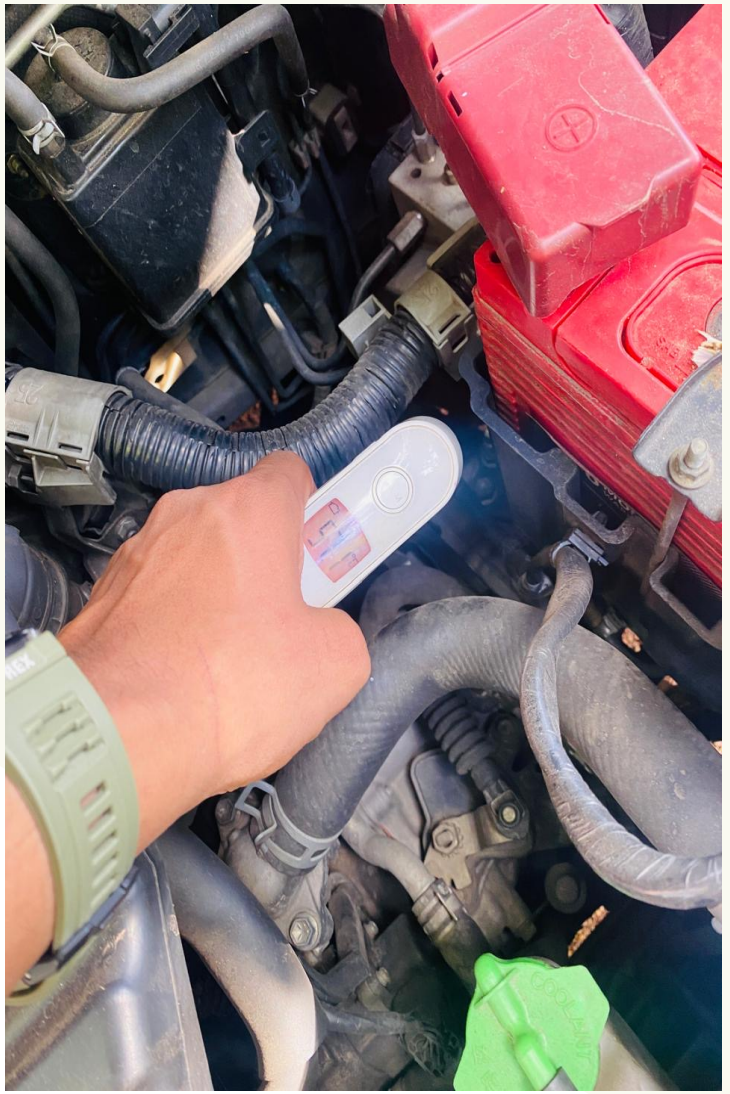
- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Identify software & Hardware requirements
- Collect data manually from vehicles and create datasets
- Collect data from various times with various climates
- Create dataset
- Data pre-processing
- Model Selection
- Design System flow chart
- Design mobile app wireframe
- Design mobile app frontend

Gathered data

A	B	C	D	E	F	G	H	I	J	K	L
1000_rpm	1000_rpm_	2000_rpm	2000_rpm_	3000_rpm	3000_rpm_	4000_rpm	4000_rpm_	5000_rpm	5000_rpm_	cabin_with	rear_witho
1006	63.9	2051	64.4	3017	70.4	4019	74.5	5047	77	35.6	35
1026	62.8	2049	70.9	2991	67.6	4059	75.9	5008	76.8	34.1	34.7
1019	61.8	2037	66.1	3033	67.1	4010	75.5	5030	76.6	35.2	35.3
999	64.7	2025	66	2998	69.9	4018	75.1	4994	75.2	35.9	35.7
1053	63.7	1979	69.7	3005	67.4	4044	73.4	5030	73.8	34.9	36.4
999	63.2	2046	68.6	3004	70.4	4005	75.2	5041	75.1	35	34.8
1039	62.5	2020	67.9	3000	68.7	4030	75.5	5034	74.7	35.2	34.7
992	61.6	2032	67.8	2982	69.8	4065	74.7	5051	75.3	35.7	35.7
1049	63.1	2001	65.2	2982	68.6	4006	75.2	5010	76	34.7	35.3
1040	61.9	1979	65.8	3024	67.4	4008	72.1	4995	75.6	35.8	35.7
974	64.5	1996	65.2	3005	69.8	4049	74.7	5017	74.7	34.7	35.5
1044	62.4	1999	68.2	2994	68.5	4048	73.5	5051	74.7	35.2	35.9
1008	64.9	2014	67.9	3012	71.7	4005	74.8	5012	74.5	34.3	36.2
969	61.2	1997	64.1	3021	68.5	4034	74.5	5013	76.7	35.1	36.4
1008	62.5	2033	64.9	3018	69.4	3996	74.7	5005	75	35.7	35.6
1030	62.3	2052	70.3	3014	70.2	4051	73.9	5042	73.7	34.9	35.7
967	63.1	2045	64.4	3023	67.9	4066	72.1	5006	73.2	34.1	34.3
999	62.9	2044	66.8	2976	71.3	4061	74.9	4990	76.9	34.2	36.5
1017	63	2004	64.7	3009	67.1	4063	73.8	4988	76.3	34.1	35.1

Completion of the Component

Gathered data



CNN Model

```python
import pandas as pd
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 Dense, Conv1D, Flatten, MaxPooling1D, Dropout, BatchNormalization
from tensorflow.keras.utils import to_categorical
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, LearningRateScheduler
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.regularizers import l2
import numpy as np
```

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

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

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

# Normalize features using StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Reshape X to fit the model's expected input
X_train = X_train.reshape(X_train.shape[0], X_train.shape[1], 1)
X_test = X_test.reshape(X_test.shape[0], X_test.shape[1], 1)

# Define the model with additional regularization and layers
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(12, 1), kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
```

Completion of the Component

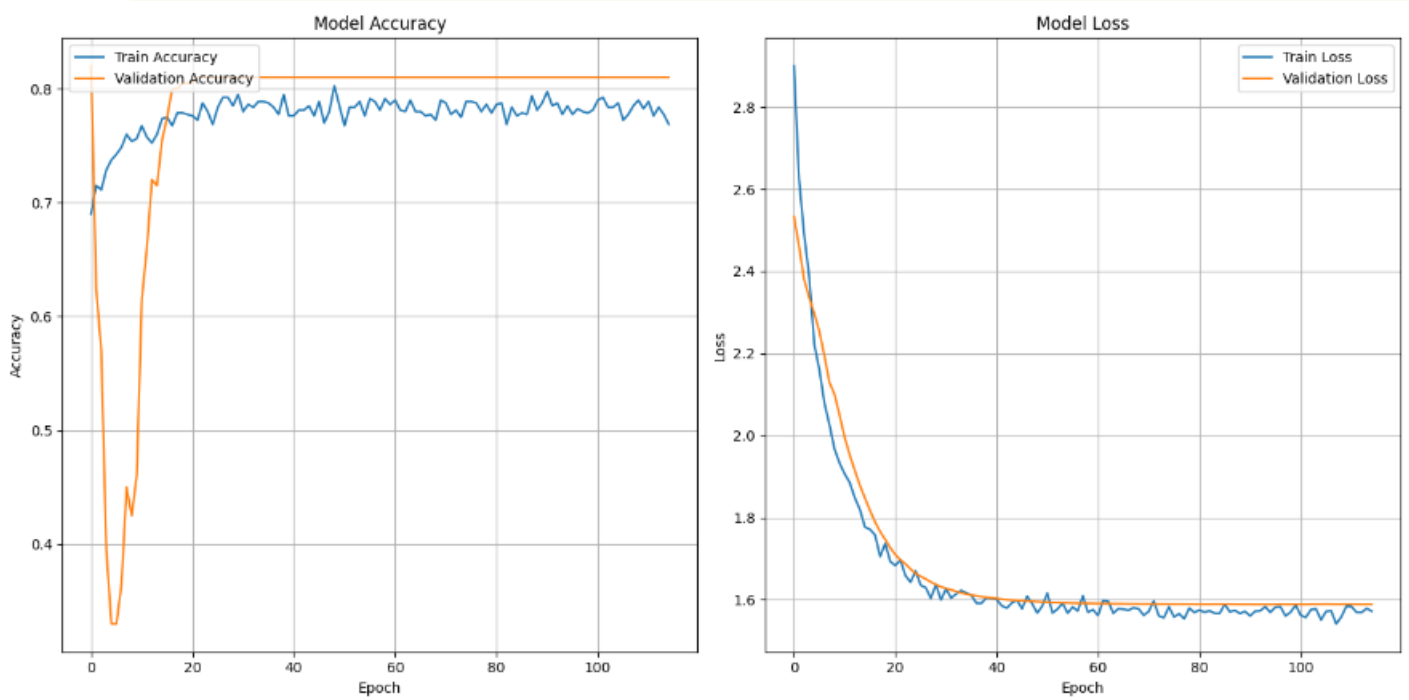
```python
# Define the model with additional regularization and layers
model = Sequential([
    Conv1D(filters=64, kernel_size=3, activation='relu', input_shape=(12, 1), kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Flatten(),
    Dense( units: 100, activation='relu', kernel_regularizer=l2(0.01)),
    Dropout(0.5),
    Dense( units: 2, activation='softmax')
])

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])
```

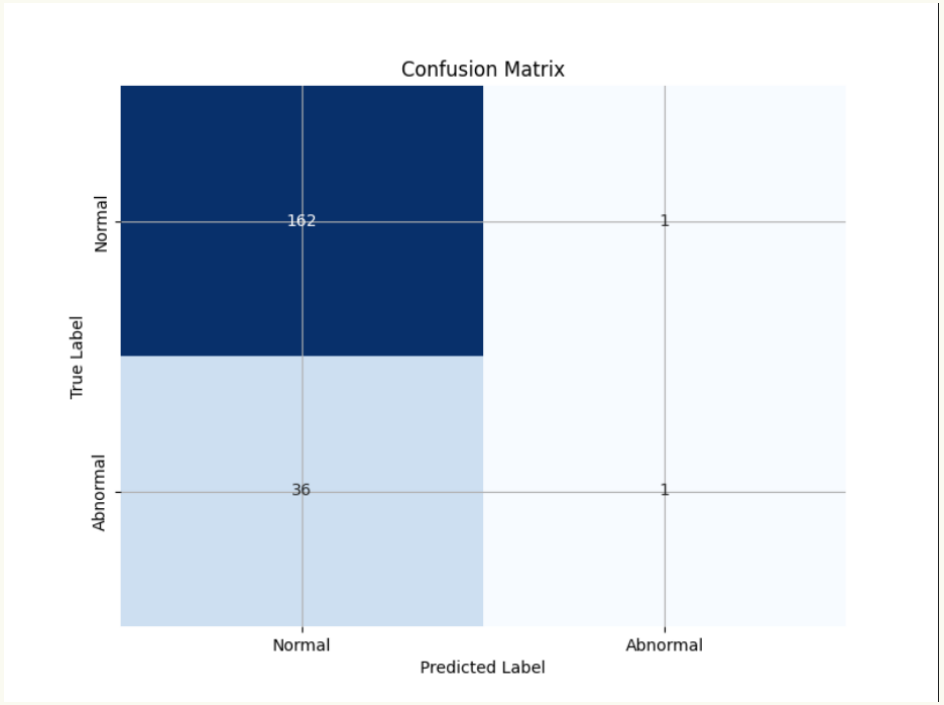
```python
# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=10, min_lr=0.0001, verbose=1)
lr_scheduler = LearningRateScheduler(lambda epoch: 0.001 * np.exp(-0.1 * epoch), verbose=1)

# Train the model
history = model.fit(X_train, y_train, epochs=150, batch_size=32, validation_data=(X_test, y_test), verbose=2,
                    callbacks=[early_stopping, reduce_lr, lr_scheduler])

# Save the model
model.save('enhanced_sensor_model_v2.h5')
```

Completion of the component

```python
# Visualization of the training process
plt.figure(figsize=(14, 7))
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: history.history['accuracy'], label='Train Accuracy')
plt.plot( *args: history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.grid(True)
plt.legend(loc='upper left')

plt.subplot( *args: 1, 2, 2)
plt.plot( *args: history.history['loss'], label='Train Loss')
plt.plot( *args: history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.grid(True)
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()

# Evaluate the model on test data
test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=2)
print(f"Test Loss: {test_loss}, Test Accuracy: {test_accuracy}")
```

Completion of the Component






```
Epoch 115: LearningRateScheduler setting learning rate to 1.119548484259094e-08.
Epoch 115/150
25/25 - 0s - loss: 1.5723 - accuracy: 0.7688 - val_loss: 1.5887 - val_accuracy: 0.8100 - lr: 1.1195e-08 - 96ms/epoch - 4ms/step
7/7 - 0s - loss: 1.5884 - accuracy: 0.8100 - 39ms/epoch - 6ms/step
Test Loss: 1.5884194374084473, Test Accuracy: 0.8100000023841858
```

RNN Model(LSTM)

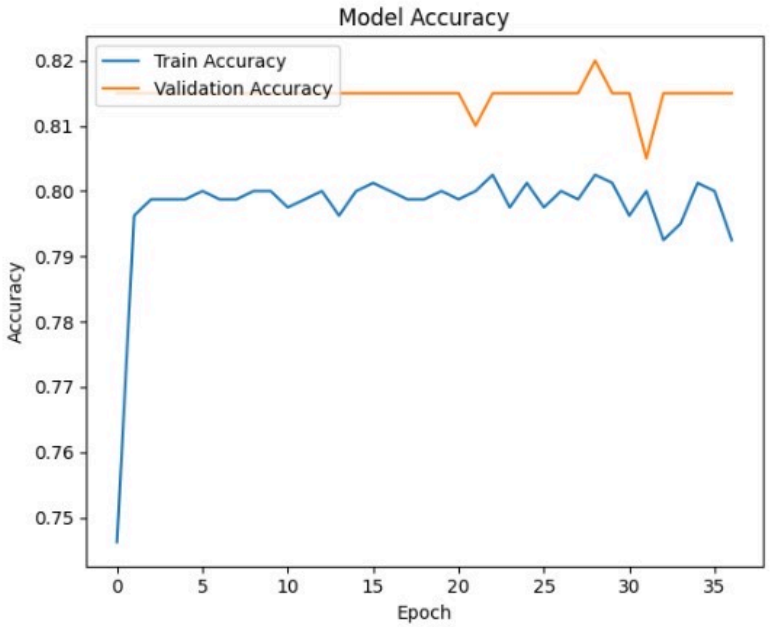
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, BatchNormalization
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)
```

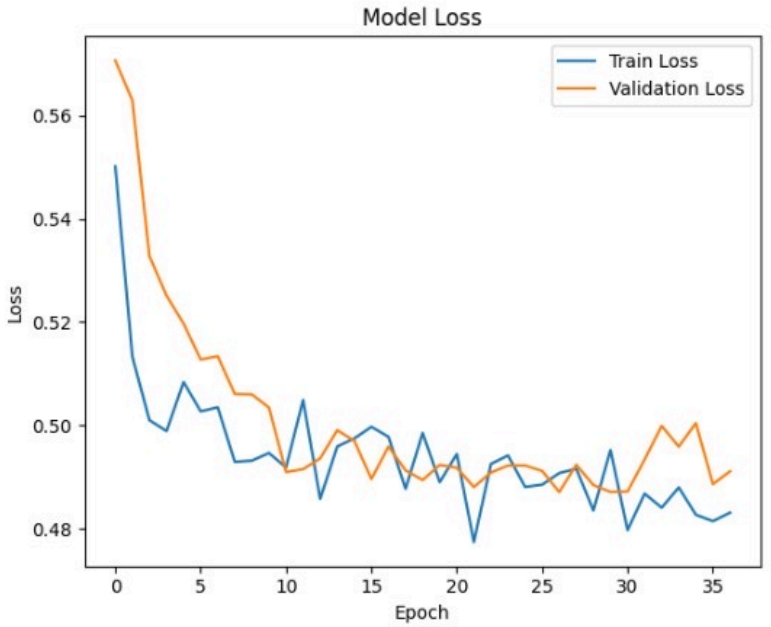
Completion of the component

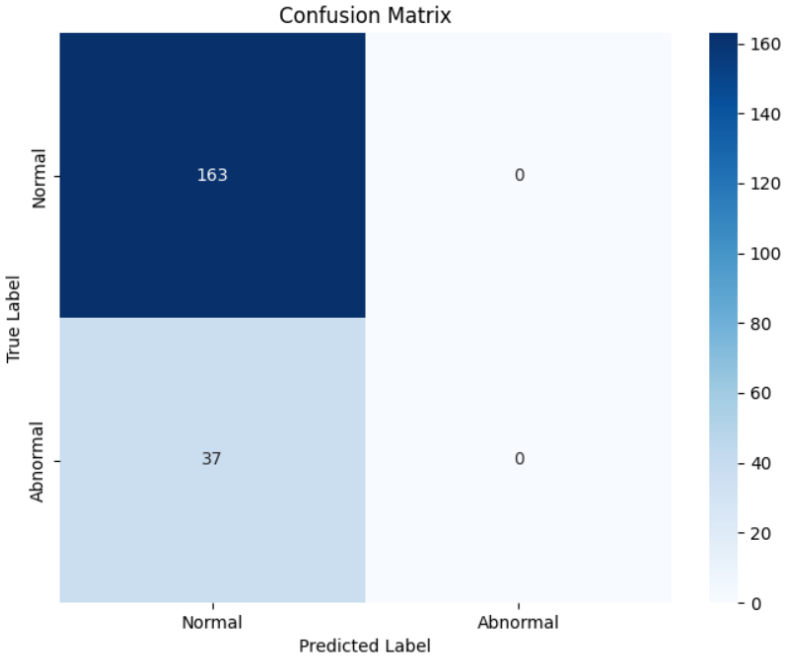
```python
# Define the LSTM model
model = Sequential([
    LSTM( units: 64, return_sequences=True, input_shape=(12, 1)),
    Dropout(0.3),
    BatchNormalization(),
    LSTM( units: 64, return_sequences=False),
    Dropout(0.3),
    Dense( units: 100, activation='relu'),
    Dropout(0.3),
    Dense( units: 2, activation='softmax')
])

# Compile the model with Adam optimizer and categorical crossentropy loss
model.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

# Setup callbacks for early stopping (to prevent overfitting) and best model checkpointing
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)
checkpoint = ModelCheckpoint( filepath: 'best_model.h5', save_best_only=True, monitor='val_loss', mode='min')

# Train the model
history = model.fit(
    X_train, y_train,
    epochs=200,
    batch_size=32,
    validation_data=(X_test, y_test),
    callbacks=[early_stopping, checkpoint],
    verbose=2
)
# Load the best saved model
model.load_weights('best_model.h5')

# Predictions
y_pred = model.predict(X_test)
y_pred_classes = np.argmax(y_pred, axis=1)
y_true = np.argmax(y_test, axis=1)

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

# Classification Report and Accuracy
print("Accuracy: {:.2f}%".format(accuracy_score(y_true, y_pred_classes) * 100))
print("\nClassification Report:\n", classification_report(y_true, y_pred_classes))
```

```python
# Plot Training and Validation Accuracy
plt.figure(figsize=(12, 5))
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: history.history['accuracy'], label='Train Accuracy')
plt.plot( *args: history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(loc='upper left')

# Plot Training and Validation Loss
plt.subplot( *args: 1, 2, 2)
plt.plot( *args: history.history['loss'], label='Train Loss')
plt.plot( *args: history.history['val_loss'], label='Validation Loss')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(loc='upper right')
plt.tight_layout()
plt.show()
```

Completion of the component







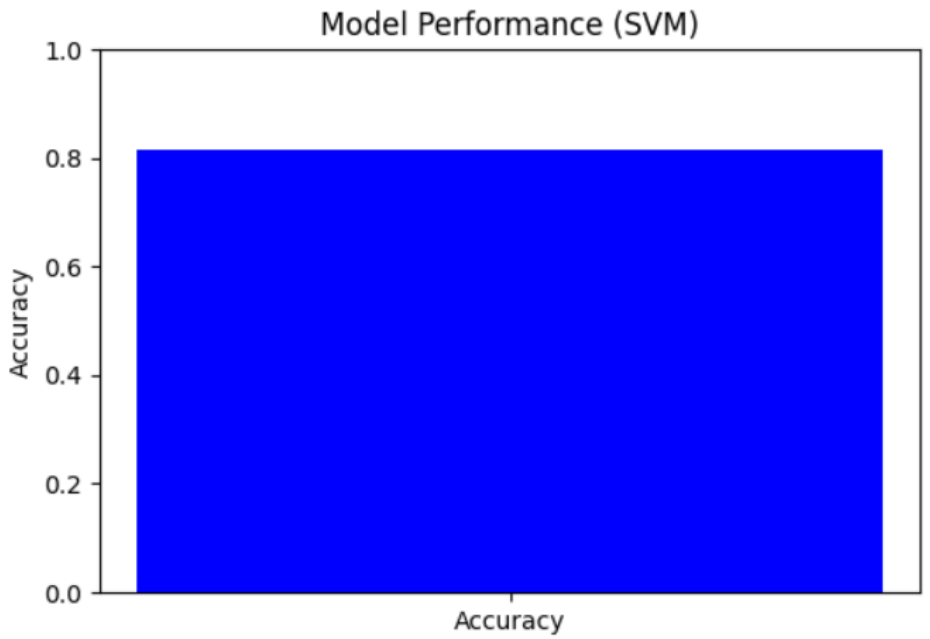

```
25/25 - 0s - loss: 0.4831 - accuracy: 0.7925 - val_loss: 0.4911 - val_accuracy: 0.8150 - 213ms/epoch - 9ms/step
7/7 [==============================] - 1s 3ms/step
Accuracy: 81.50%
```

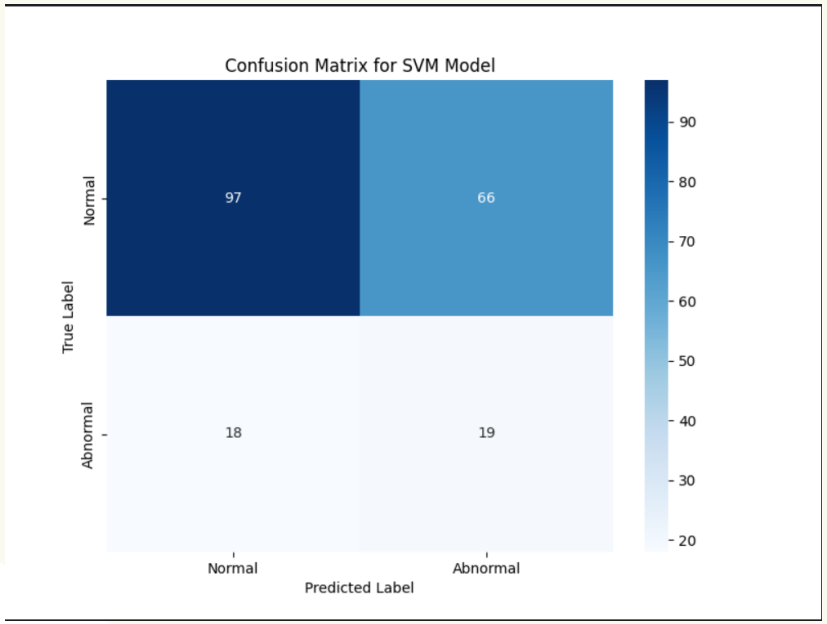
```
Classification Report:
              precision    recall  f1-score   support

           0       0.81      1.00      0.90       163
           1       0.00      0.00      0.00        37

    accuracy                           0.81       200
   macro avg       0.41      0.50      0.45       200
weighted avg       0.66      0.81      0.73       200
```

SVM Model

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score
from sklearn.svm import SVC

# 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

# Normalize features using StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 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)
```

```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))
```



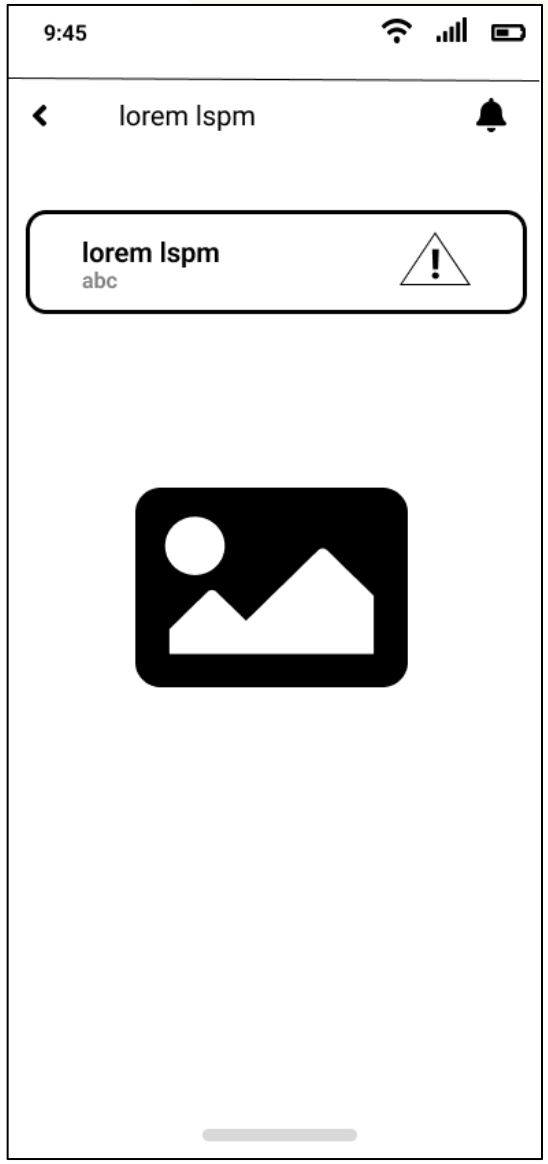


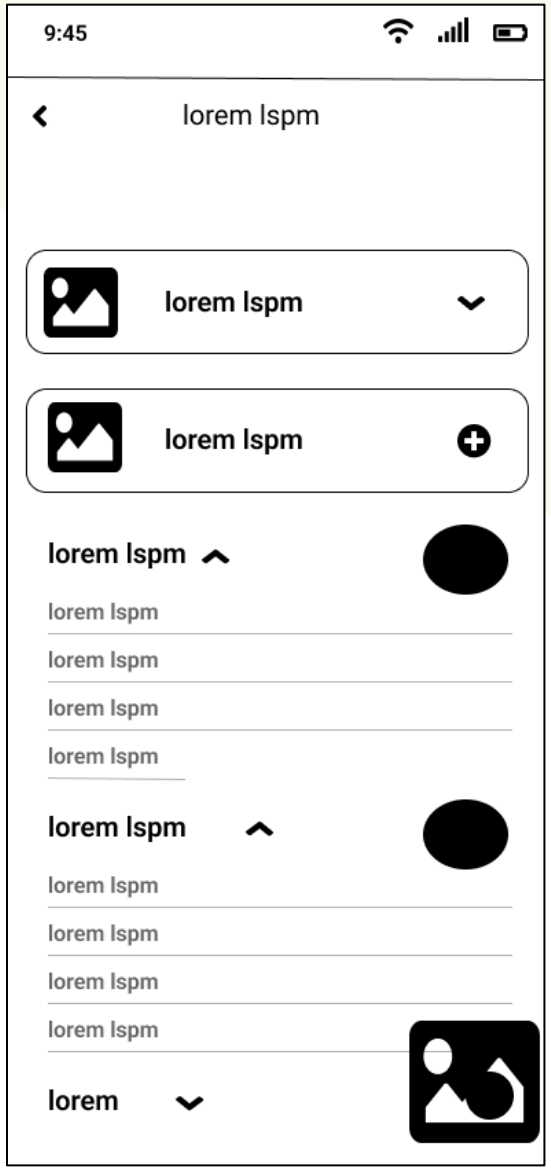

```
SVM Accuracy: 0.58

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
```

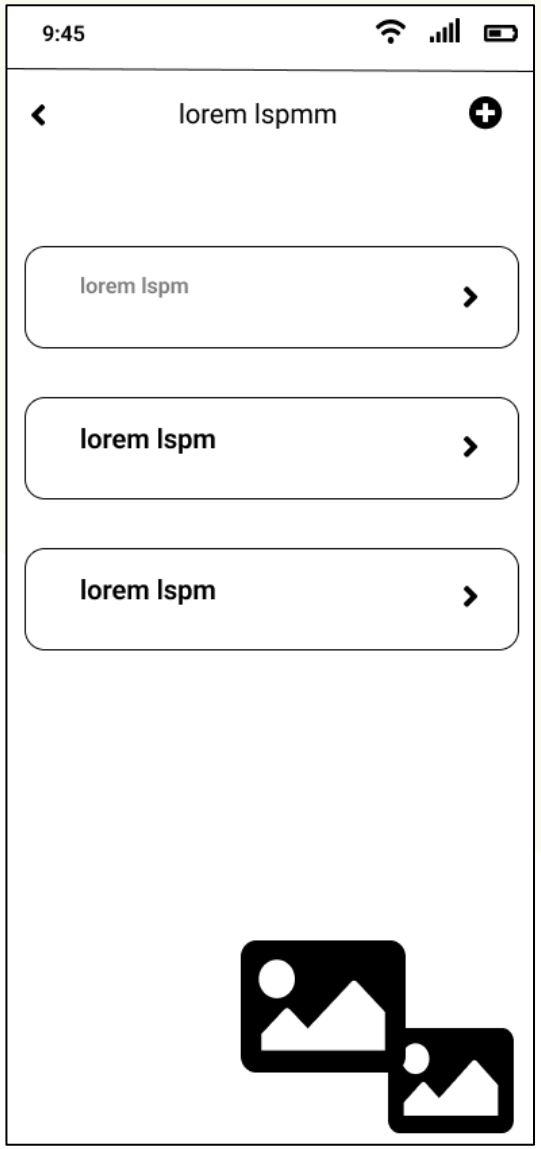
Used Algorithm	Overall Accuracy
CNN Algoritham	82.0%
Support Vector Machine(SVM)	81.50%
RNN Model(LSTM)	81.50%

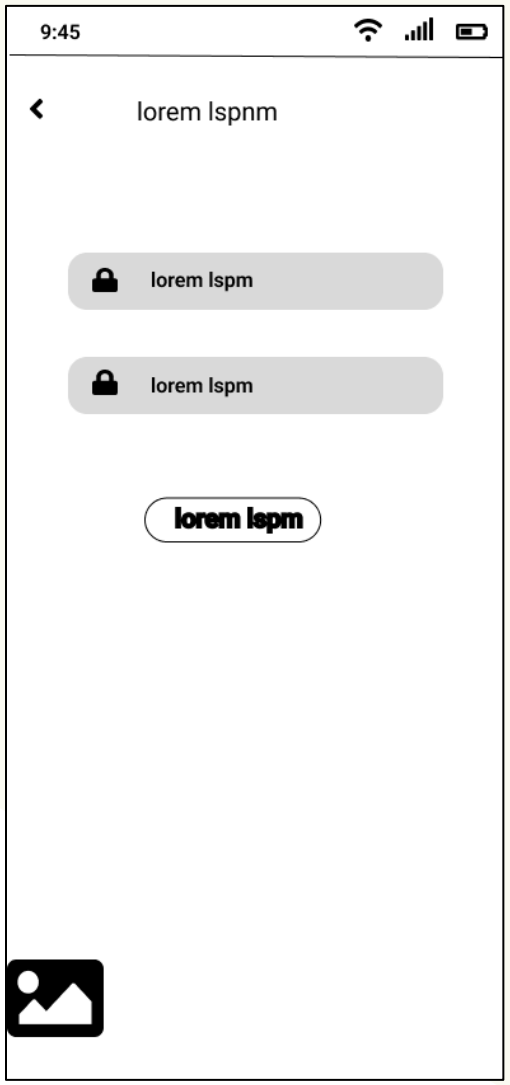
Best Practices of the development

Wireframes



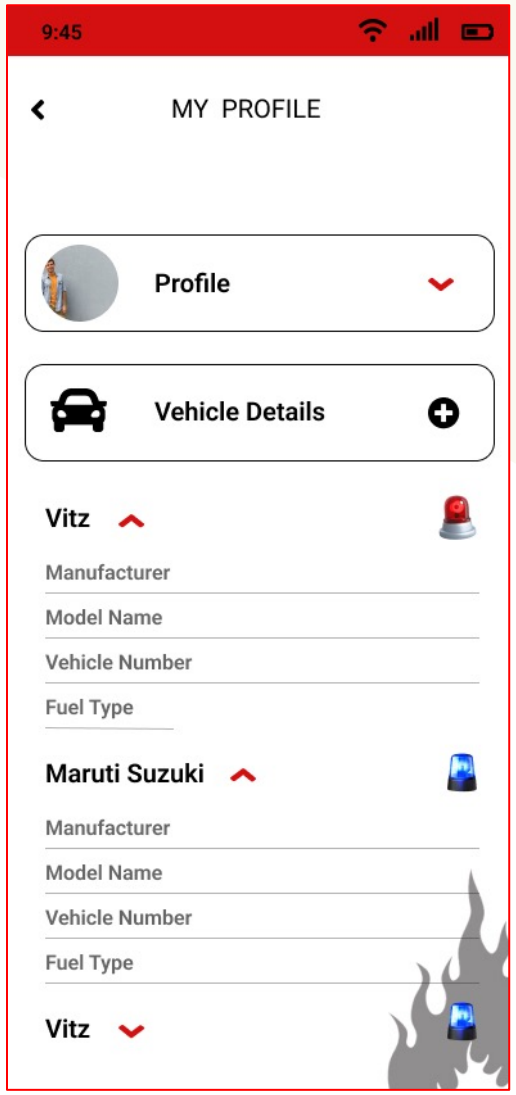


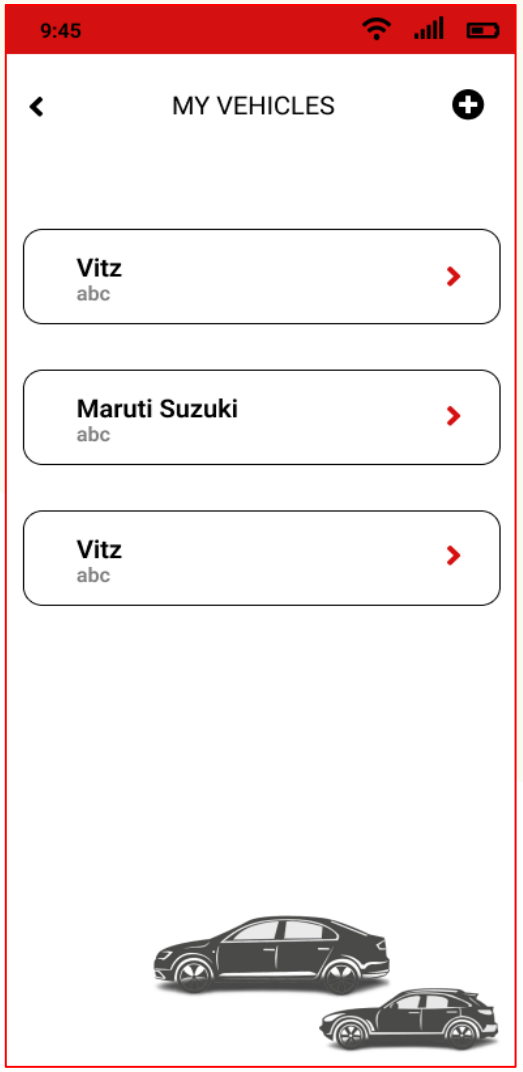


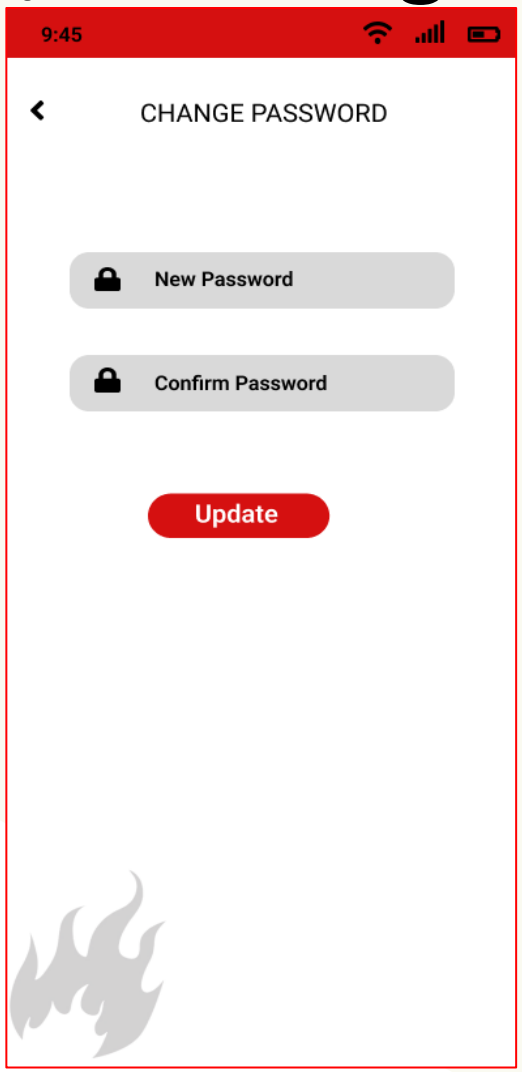


Best Practices of the development

UI/UX Designs







Tools and Technologies

Hardware Tools

- Arduino
- Smoke Detectors
- Flame Sensor
- Temperature sensors
- ESP32
- Carbon Monoxide Detectors
- Gas Sensors

Software Technologies

- MicroPython
- c/c++
- TensorFlow
- AWS/Azure
- PyCharm
- Flutter

Requirement Analysis

Functional Requirements

- Install sensors for smoke, temperature, fire, and vibration detection.
- 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.
- 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.**

Overall Progress

Completed | Ongoing | To be started

- Requirement gathering
- Requirement analysis
- Identify Software & Hardware requirements
- Collect data manually (Measure temperatures/RPM) from vehicles

Overall Progress

Design System flow chart

Design Mobile App wireframe

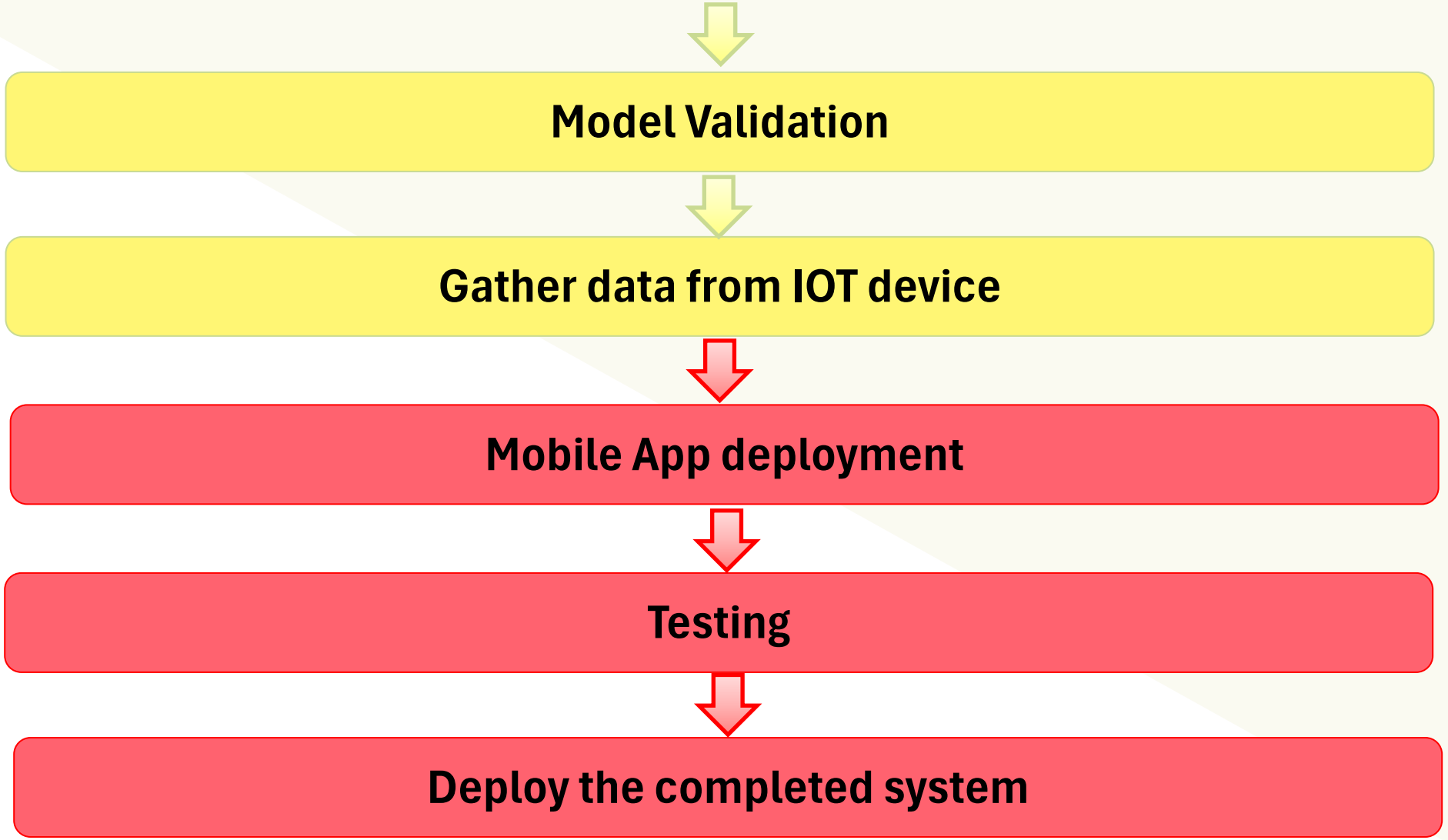
Design Mobile App UI/UX design

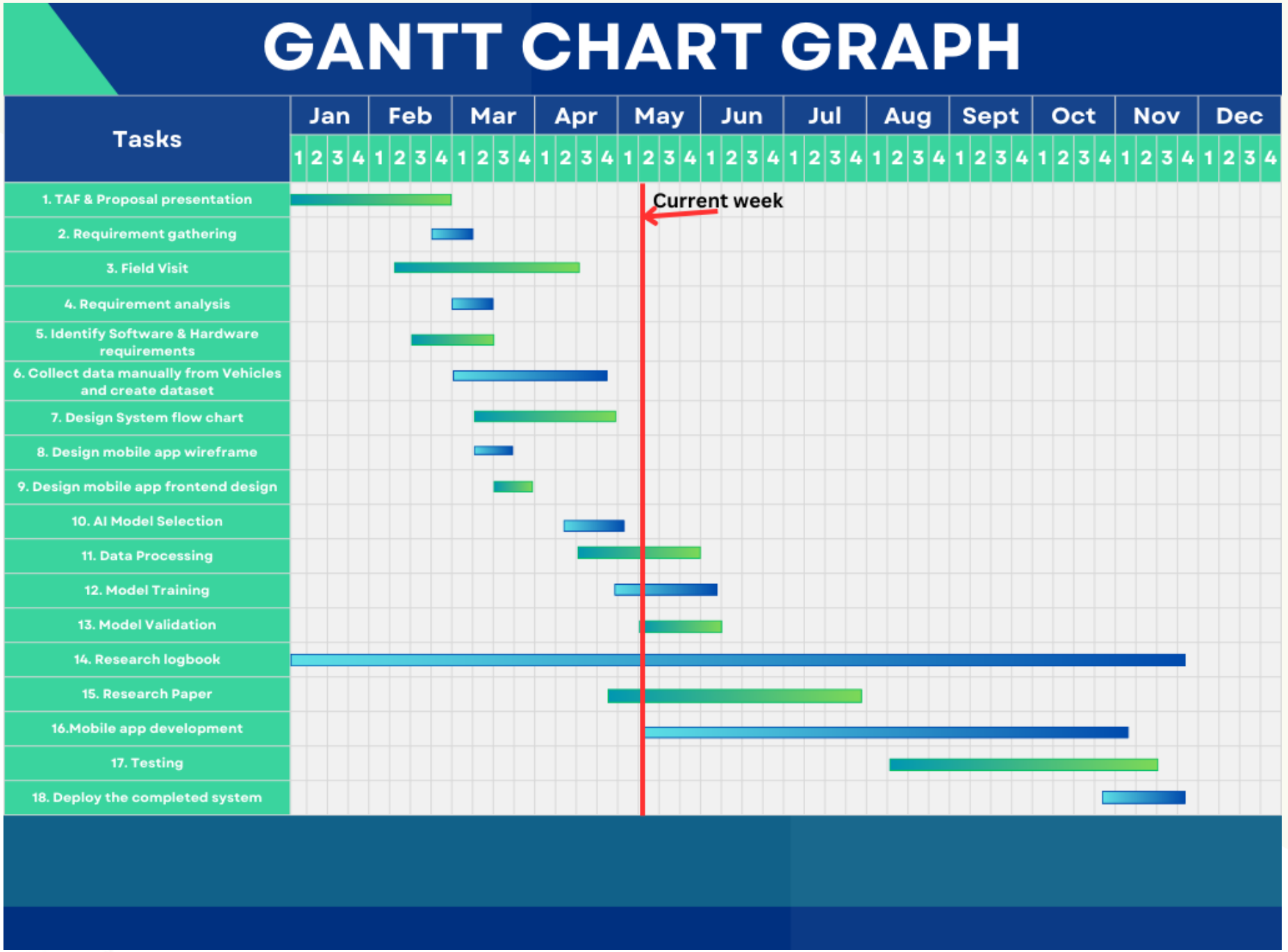
ML Model Selection

Data Processing

Model Training

Overall Progress




GANTT CHART GRAPH
Tasks
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Dec
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Current week
1. TAF & Proposal presentation
2. Requirement gathering
3. Field Visit
4. Requirement analysis
5. Identify Software & Hardware requirements
6. Collect data manually from Vehicles and create dataset
7. Design System flow chart
8. Design mobile app wireframe
9. Design mobile app frontend design
10. AI Model Selection
11. Data Processing
12. Model Training
13. Model Validation
14. Research logbook
15. Research Paper
16.Mobile app development
17. Testing
18. Deploy the completed system

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

Background

- **Fire severity** in vehicle fires refers to the extent of damage or intensity of the fire.
- It ranges from minor, with little damage, to major, causing extensive harm to the vehicle and potentially endangering lives.
- Vehicle fire severity is often determined by factors such as the size ,intensity of the flames, materials(smoke) involved in the fire.
- Vehicle fire severity can be classified into different levels, such as minor, moderate, and severe, with each level representing a different percentage of damage to the vehicle.
- Understanding vehicle fire severity is important for developing effective fire safety measures and emergency response protocols.

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

- Predicting the real time vehicle fire severity levels(High,Medium,Low)with percentages at the moment of the fire.
- 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 level 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.**

Completion of the Component

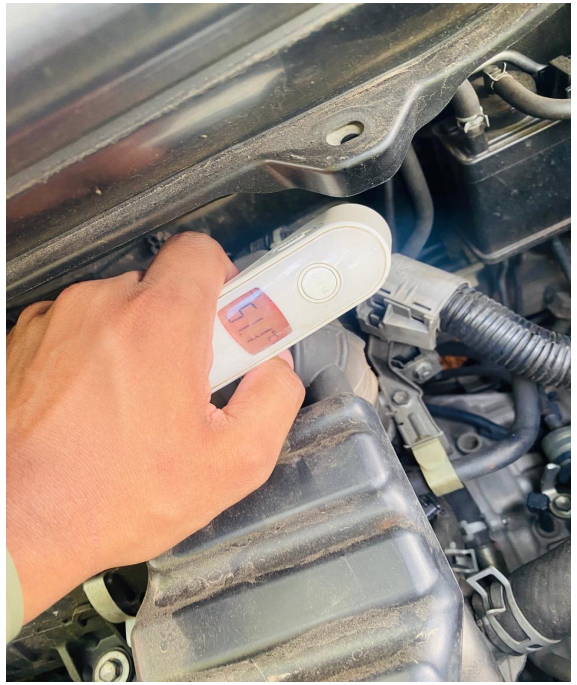
- Background Study
- Identify the research problem
- Identify the research gap
- Identify the solution
- Requirement gathering
- Requirement analysis
- Identify Software requirements
- Collect data manually (Measure temperatures/RPM)
- Create dataset
- Data preprocessing
- Model Selection
- Design System 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
- Complete Mobile App UI/UX flows and Prototypes.

Gathered data

A	B	C	D	E	F	G	H	I	J	K	L
1000_rpm	1000_rpm_	2000_rpm	2000_rpm_	3000_rpm	3000_rpm_	4000_rpm	4000_rpm_	5000_rpm	5000_rpm_	cabin_with	rear_witho
1006	63.9	2051	64.4	3017	70.4	4019	74.5	5047	77	35.6	35
1026	62.8	2049	70.9	2991	67.6	4059	75.9	5008	76.8	34.1	34.7
1019	61.8	2037	66.1	3033	67.1	4010	75.5	5030	76.6	35.2	35.3
999	64.7	2025	66	2998	69.9	4018	75.1	4994	75.2	35.9	35.7
1053	63.7	1979	69.7	3005	67.4	4044	73.4	5030	73.8	34.9	36.4
999	63.2	2046	68.6	3004	70.4	4005	75.2	5041	75.1	35	34.8
1039	62.5	2020	67.9	3000	68.7	4030	75.5	5034	74.7	35.2	34.7
992	61.6	2032	67.8	2982	69.8	4065	74.7	5051	75.3	35.7	35.7
1049	63.1	2001	65.2	2982	68.6	4006	75.2	5010	76	34.7	35.3
1040	61.9	1979	65.8	3024	67.4	4008	72.1	4995	75.6	35.8	35.7
974	64.5	1996	65.2	3005	69.8	4049	74.7	5017	74.7	34.7	35.5
1044	62.4	1999	68.2	2994	68.5	4048	73.5	5051	74.7	35.2	35.9
1008	64.9	2014	67.9	3012	71.7	4005	74.8	5012	74.5	34.3	36.2
969	61.2	1997	64.1	3021	68.5	4034	74.5	5013	76.7	35.1	36.4
1008	62.5	2033	64.9	3018	69.4	3996	74.7	5005	75	35.7	35.6
1030	62.3	2052	70.3	3014	70.2	4051	73.9	5042	73.7	34.9	35.7
967	63.1	2045	64.4	3023	67.9	4066	72.1	5006	73.2	34.1	34.3
999	62.9	2044	66.8	2976	71.3	4061	74.9	4990	76.9	34.2	36.5
1017	63	2004	64.7	3009	67.1	4063	73.8	4988	76.3	34.1	35.1

Completion of the component

Gathered data



Model Selection

Import Libraries and Data Preprocessing

jupyter Linear Regression 1.2 Last Checkpoint: 44 minutes ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [1]:
```
#Import Dependencies
import pandas as pd   #Data manupulation $ Analysis
import numpy as np    #Numerical computation
import matplotlib.pyplot as plt  #Visualizations
import seaborn as sns   #Informative statistical graph
```

In [2]:
```
#Load Data To the pandas frame
vehicle_data = pd.read_csv('rpm_temp_dataset.csv')
```

In [5]:
```
# Display the first five rows
print(vehicle_data.head())
```

```
   1000_rpm  1000_rpm_temprature  2000_rpm  2000_rpm_temprature  3000_rpm  \
0      1006                 63.9      2051                 64.4      3017
1      1026                 62.8      2049                 70.9      2991
2      1019                 61.8      2037                 66.1      3033
3       999                 64.7      2025                 66.0      2998
4      1053                 63.7      1979                 69.7      3005

   3000_rpm_temprature  4000_rpm  4000_rpm_temprature  5000_rpm  \
0                 70.4      4019                 74.5      5047
1                 67.6      4059                 75.9      5008
2                 67.1      4010                 75.5      5030
3                 69.9      4018                 75.1      4994
4                 67.4      4044                 73.4      5030

   5000_rpm_temprature  cabin_without_ac_sunny_day  rear_without_ac_sunny_day
0                 77.0                        35.6                       35.0
1                 76.8                        34.1                       34.7
2                 76.6                        35.2                       35.3
3                 75.2                        35.9                       35.7
4                 73.8                        34.9                       36.4
```

In [7]:
```
#To view the last few rows
vehicle_data.tail()
```

jupyter Linear Regression 1.2 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [7]:
```
vehicle_data.tail()
```

Out[7]:

	1000_rpm	1000_rpm_temprature	2000_rpm	2000_rpm_temprature	3000_rpm	3000_rpm_temprature	4000_rpm	4000_rpm_temprature	5000_rpm	5000_rpm_
995	990	63.5	1999	64.4	2996	68.2	4038	73.3	5009	
996	968	63.5	2039	66.6	2996	67.5	3999	73.0	5036	
997	996	64.4	2016	65.7	2978	71.8	4011	75.4	4989	
998	1014	63.4	2053	65.6	3003	68.3	4023	74.6	5012	
999	1007	61.7	2052	67.4	2976	70.7	4056	73.2	5030	

In [9]:
```
#Get the column headings of the data set
vehicle_data.columns.values
```

Out[9]:
```
array(['1000_rpm', '1000_rpm_temprature', '2000_rpm',
       '2000_rpm_temprature', '3000_rpm', '3000_rpm_temprature',
       '4000_rpm', '4000_rpm_temprature', '5000_rpm',
       '5000_rpm_temprature', 'cabin_without_ac_sunny_day',
       'rear_without_ac_sunny_day'], dtype=object)
```

In [11]:
```
#Explore the data types of the columns
vehicle_data.dtypes
```

Out[11]:
```
1000_rpm                      int64
1000_rpm_temprature          float64
2000_rpm                      int64
2000_rpm_temprature          float64
3000_rpm                      int64
3000_rpm_temprature          float64
4000_rpm                      int64
4000_rpm_temprature          float64
5000_rpm                      int64
5000_rpm_temprature          float64
cabin_without_ac_sunny_day   float64
rear_without_ac_sunny_day    float64
dtype: object
```

Import Libraries and Data Preprocessing

jupyter Linear Regression 1.2 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [12]:
```
# Check for missing values
missing_data = vehicle_data.isnull().sum()
print(missing_data)
```

```
1000_rpm                    0
1000_rpm_temprature         0
2000_rpm                    0
2000_rpm_temprature         0
3000_rpm                    0
3000_rpm_temprature         0
4000_rpm                    0
4000_rpm_temprature         0
5000_rpm                    0
5000_rpm_temprature         0
cabin_without_ac_sunny_day  0
rear_without_ac_sunny_day   0
dtype: int64
```

In [13]:
```
# Drop rows with missing values if any
# if missing_data.sum() > 0:
#     vehicle_data = data.dropna()
```

In [15]:
```
#Describe the data set
vehicle_data.describe()
```

Out[15]:

	1000_rpm	1000_rpm_temprature	2000_rpm	2000_rpm_temprature	3000_rpm	3000_rpm_temprature	4000_rpm	4000_rpm_temprature	5000_rp
count	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.000000	1000.00000	1000.0000
mean	1011.881000	63.040800	2014.489000	67.539100	3004.837000	69.467800	4026.409000	73.98490	5022.6410
std	25.545595	1.134788	22.030916	2.056952	17.258455	1.445902	24.302929	1.13687	20.7429
min	967.000000	61.000000	1976.000000	64.000000	2976.000000	67.000000	3986.000000	72.00000	4987.0000
25%	990.000000	62.100000	1996.000000	65.800000	2990.000000	68.275000	4006.000000	73.00000	5005.0000
50%	1012.000000	63.100000	2013.500000	67.600000	3005.000000	69.400000	4026.000000	73.90000	5023.0000
75%	1035.000000	64.000000	2034.000000	69.400000	3020.000000	70.700000	4048.000000	75.00000	5040.0000
max	1054.000000	65.000000	2053.000000	71.000000	3035.000000	72.000000	4068.000000	76.00000	5058.0000

Build the model and train

jupyter Linear Regression 1.2 Last Checkpoint: an hour ago (unsaved changes)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [19]:
```
#Import libraries for scalling data and transform
from sklearn.preprocessing import StandardScaler

# Scaling the features
scaler = StandardScaler()
features_scaled = scaler.fit_transform(features)
```

In [22]:
```
#Import libraries fore training an testing subsets of data
from sklearn.model_selection import train_test_split

# Splitting the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target, test
```

In [33]:
```
#Import libraries for linear regression class
from sklearn.linear_model import LinearRegression

# Initialize and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[33]:
```
▾ LinearRegression
LinearRegression()
```

In [24]:
```
#Import metrics modules
from sklearn.metrics import mean_squared_error, r2_score

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')   #Average of the squares of the errors
```

jupyter Linear Regression 1.2 Last Checkpoint: an hour ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help

Not Trusted | Python 3 (ipykernel)

Run Code

In [24]:
```
#Import metrics modules
from sklearn.metrics import mean_squared_error, r2_score

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')   #Average of the squares of the errors
print(f'R-squared: {r2}')    #How close the data are to the fitted regression line
```

```
Mean Squared Error: 0.3004216803413622
R-squared: -0.019211221171745008
```

In [25]:
```
# Calculate maximum and minimum of actual values
max_y = np.max(y_test)
min_y = np.min(y_test)

# Calculate range
range_y = max_y - min_y

# Normalized MSE as a fraction of the range
nmse = mse / (range_y ** 2)

accuracy_score = (1 - nmse) * 100
print(f'Accuracy-Score: {accuracy_score:.2f}%')
```

```
Accuracy-Score: 92.49%
```

In [32]:
```
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import pandas as pd

print(vehicle_data.columns)
```

In [12]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features_scaled, target, test_size=0.2, random_state=0)
```

In [13]:
```
# Initialize and train the SVM model
svm_model = SVR(kernel='rbf')  # You can experiment with different kernels like 'linear', 'poly', 'rbf'
svm_model.fit(X_train, y_train)
```

Out[13]:
```
SVR
SVR()
```

In [14]:
```
# Predict on the test set
y_pred = svm_model.predict(X_test)
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 0.3364807877582887
R-squared: -0.14154542442567974
```

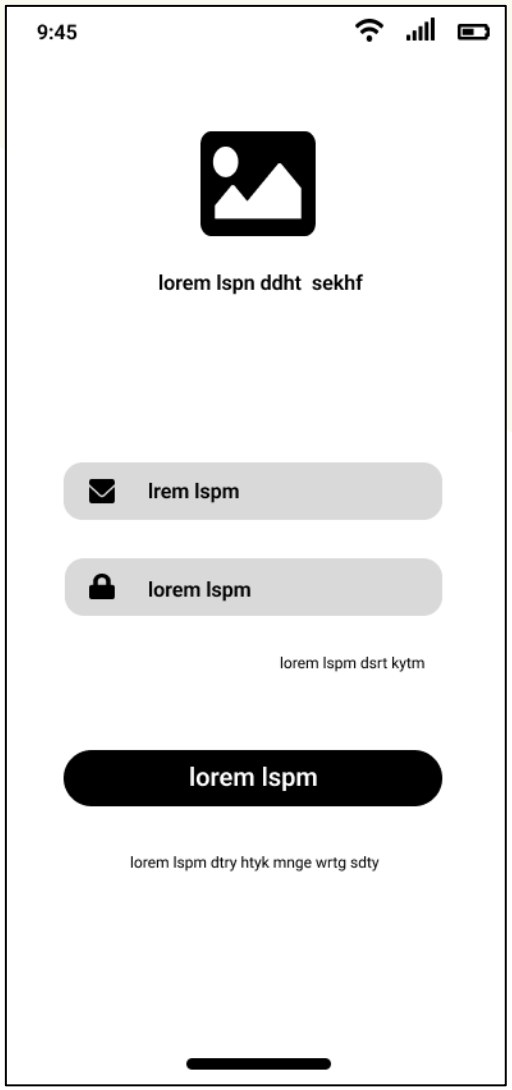
In [15]:
```
# Calculate the range of actual values
max_y = np.max(y_test)
min_y = np.min(y_test)
range_y = max_y - min_y

# Normalized MSE as a fraction of the range
nmse = mse / (range_y ** 2)
accuracy_score = (1 - nmse) * 100
print(f'Accuracy-Score: {accuracy_score:.2f}%')
```

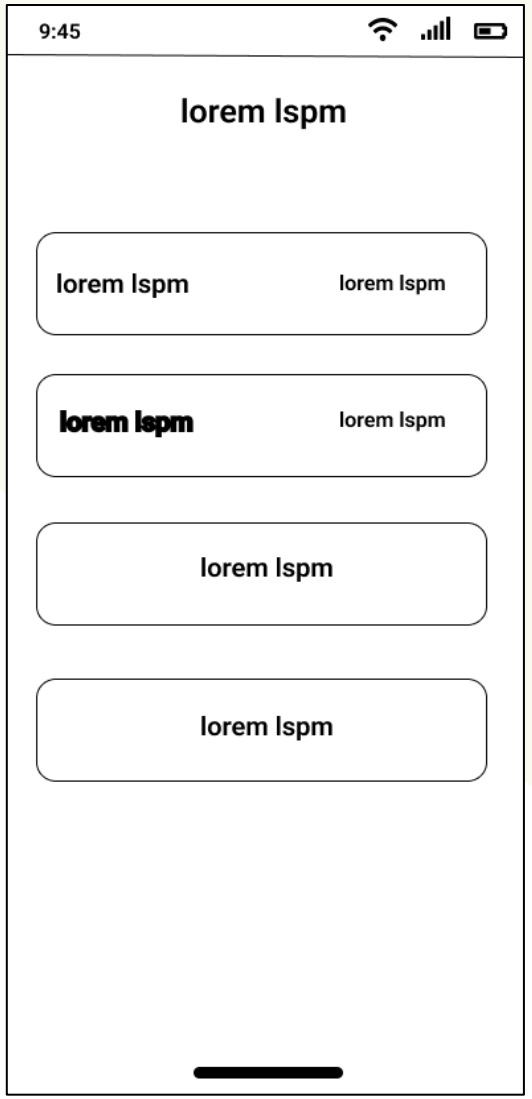
```
Accuracy-Score: 91.59%
```

Used Algorithm	Overall Accuracy
Linear Regression	92.49%
Support Vector Machine(SVM)	91.59%
Decision Trees	Plan to implement
Logistic Regression	Plan to implement

Wireframes


9:45
lorem lspn ddht sekhf

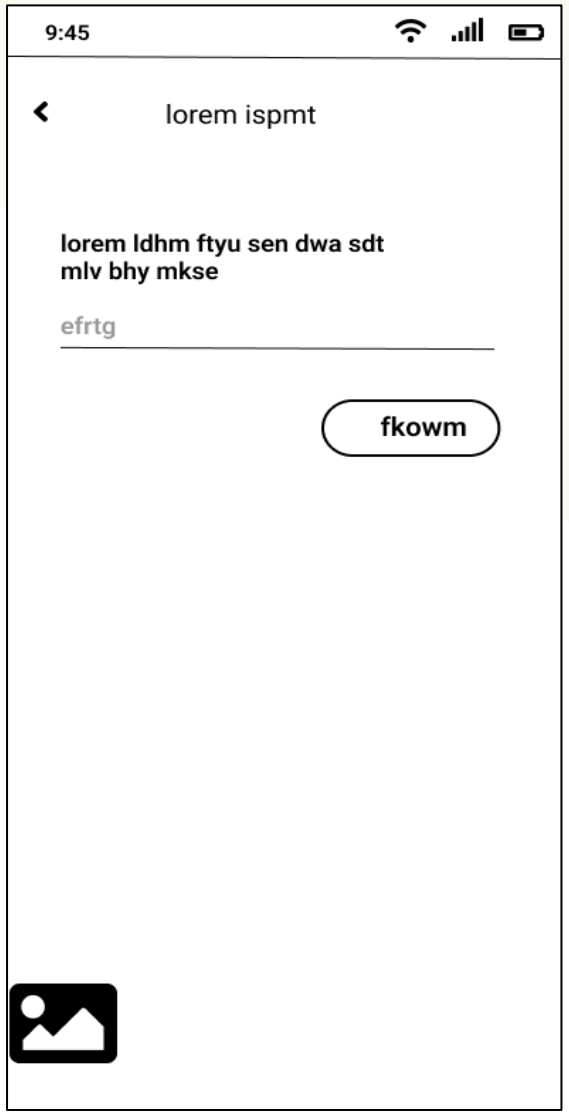

9:45
lorem lspn ddht sekhf
lrem lspm
lorem lspm
lorem lspm dsrt kytm
lorem lspm
lorem lspm dtry htyk mnge wrtg sdty

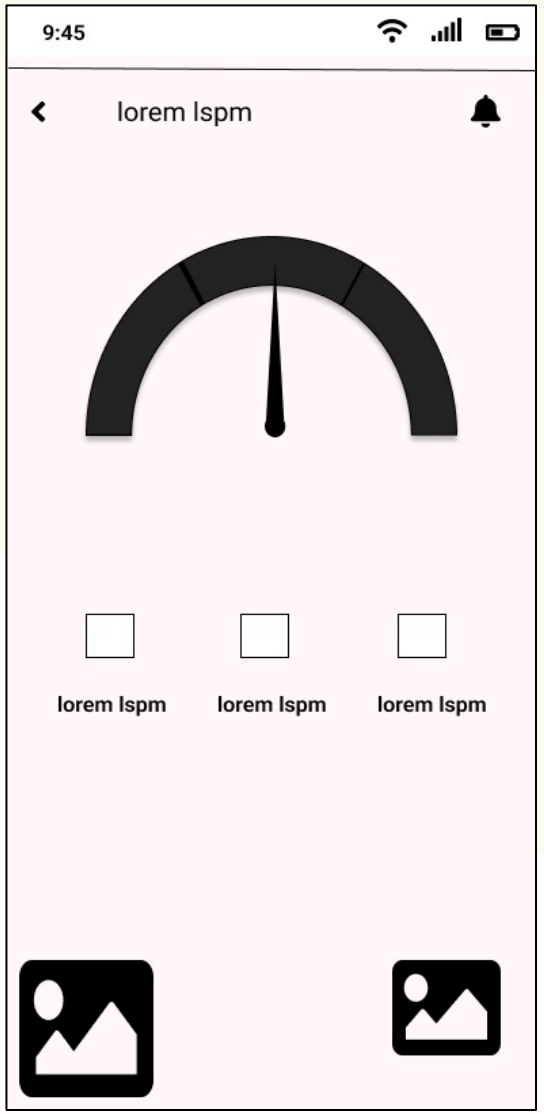

9:45
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm

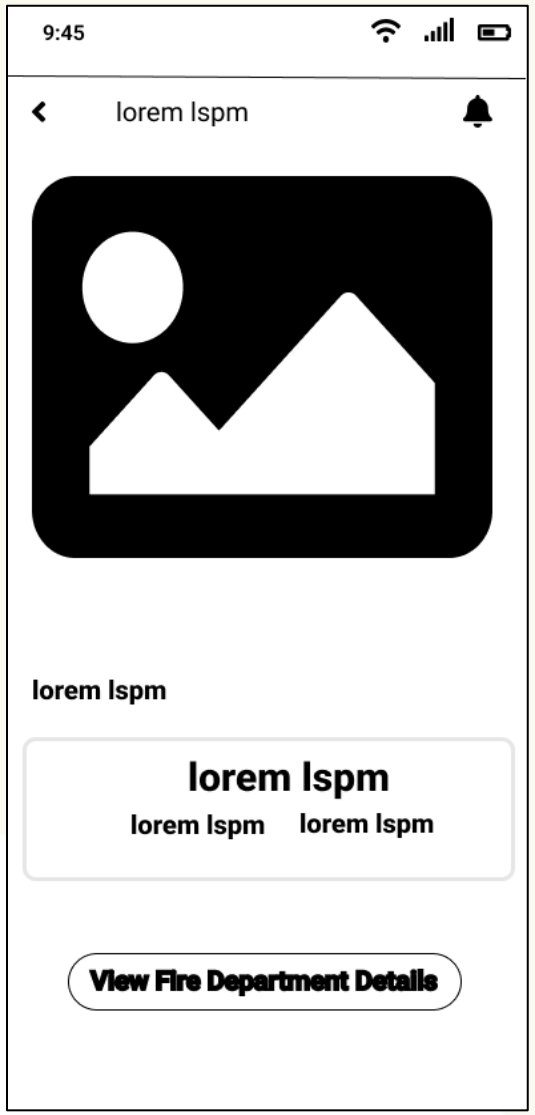

9:45
lorem lspm
lorem lspm dyht
lorem lspm dtr htrmk
lorem lspm fty wegtr
lorem lspm fty
lorem dtyb ghkiy sln
lorem kmhn ghm tyg drew awdf blonh
lorem fgt sdw vcbn

Best Practices of the development

Wireframes

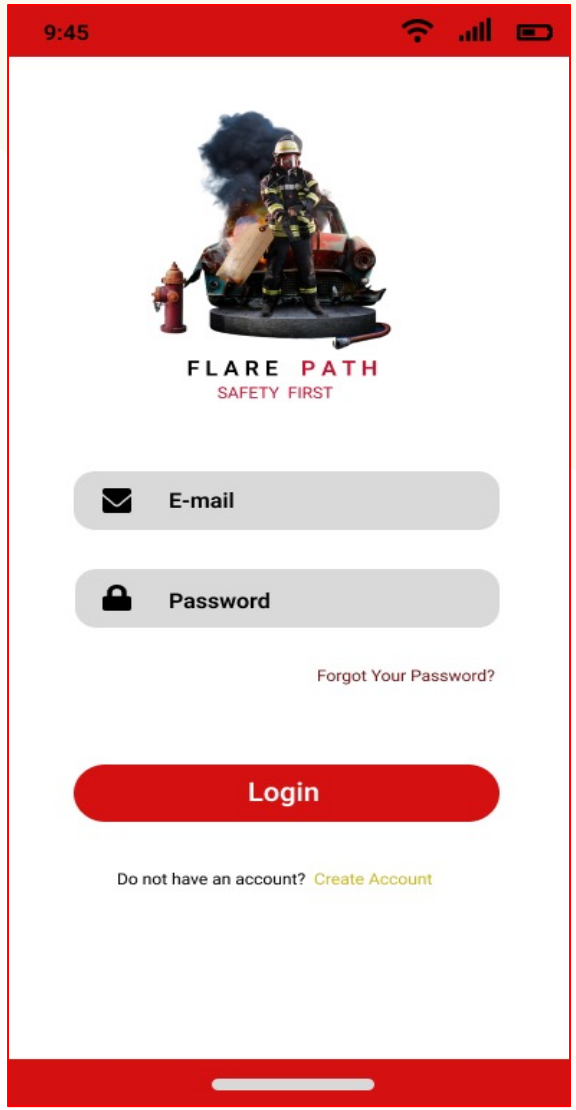


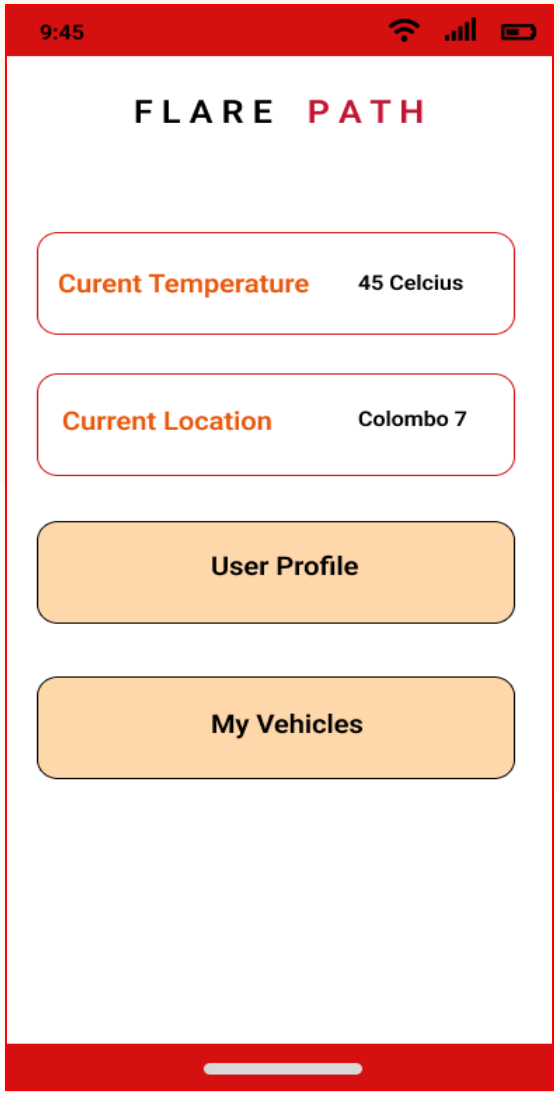


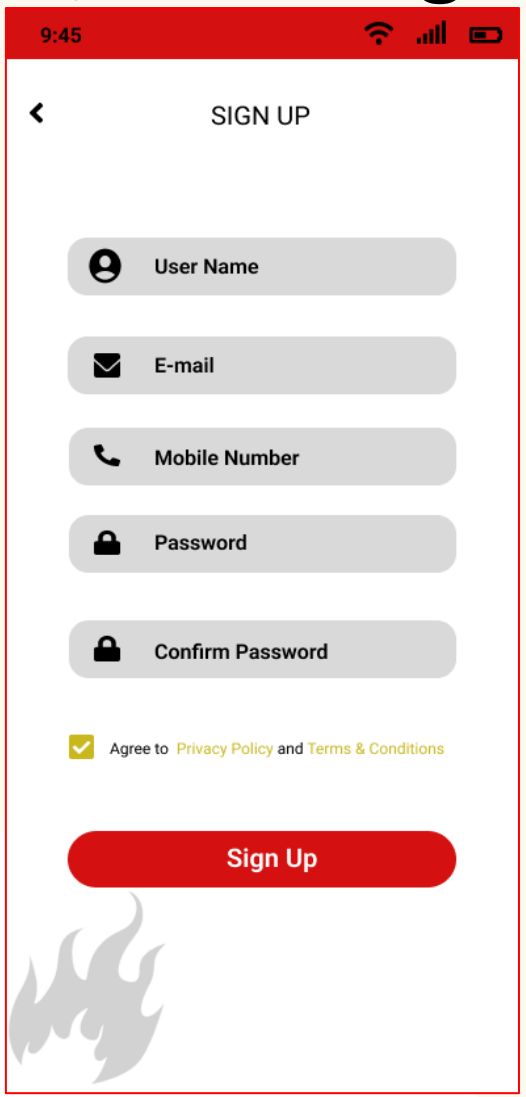


Best Practices of the development

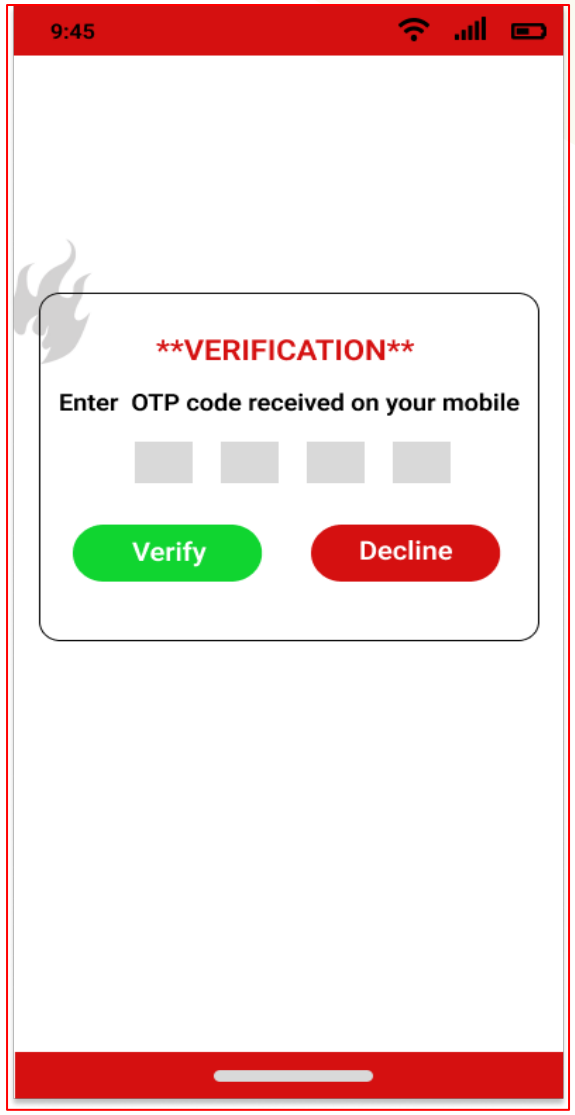
UI/UX Designs

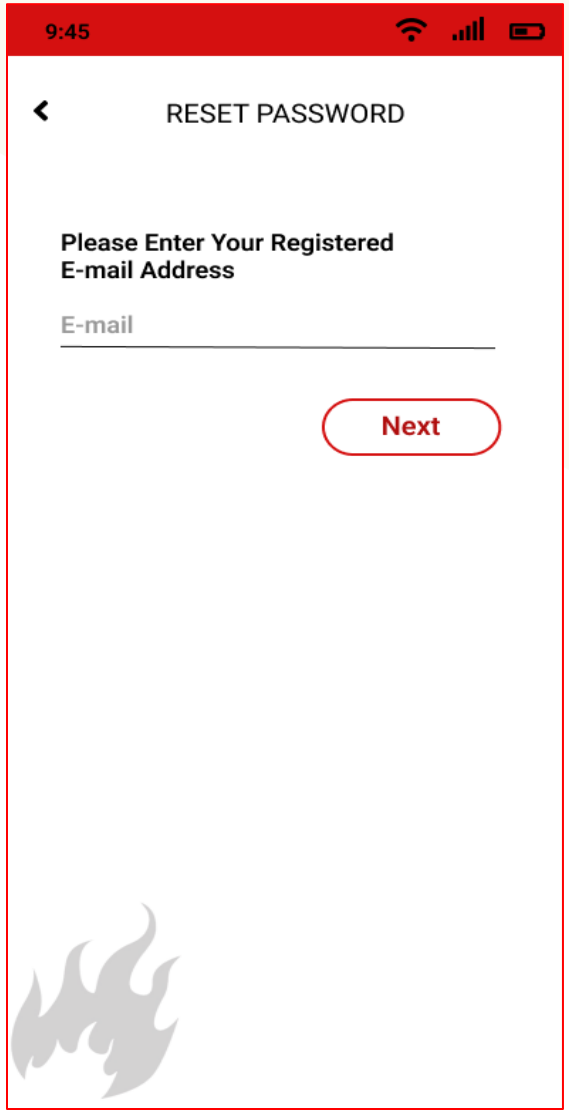


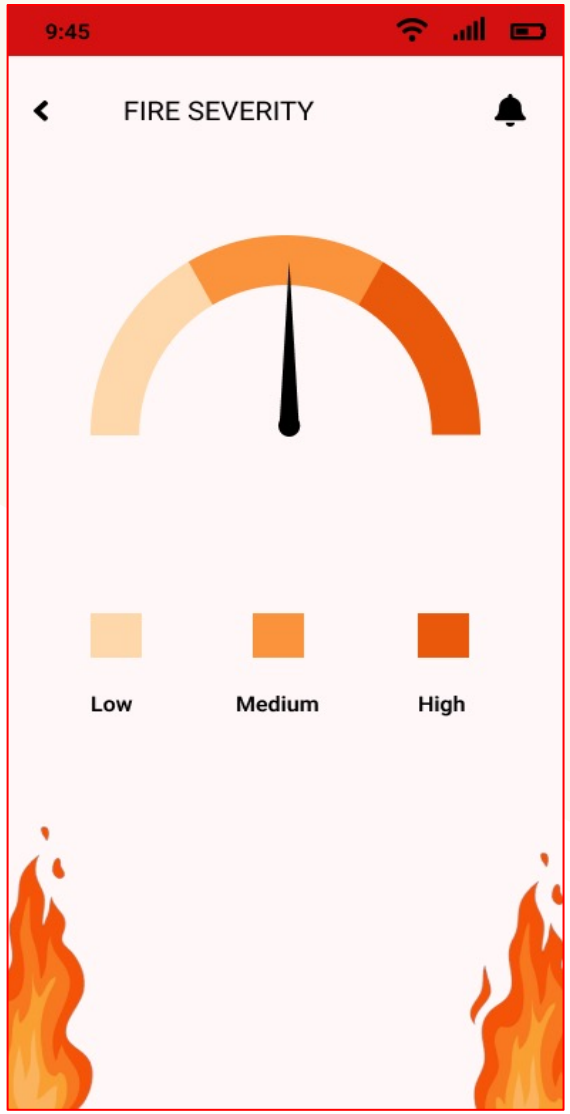


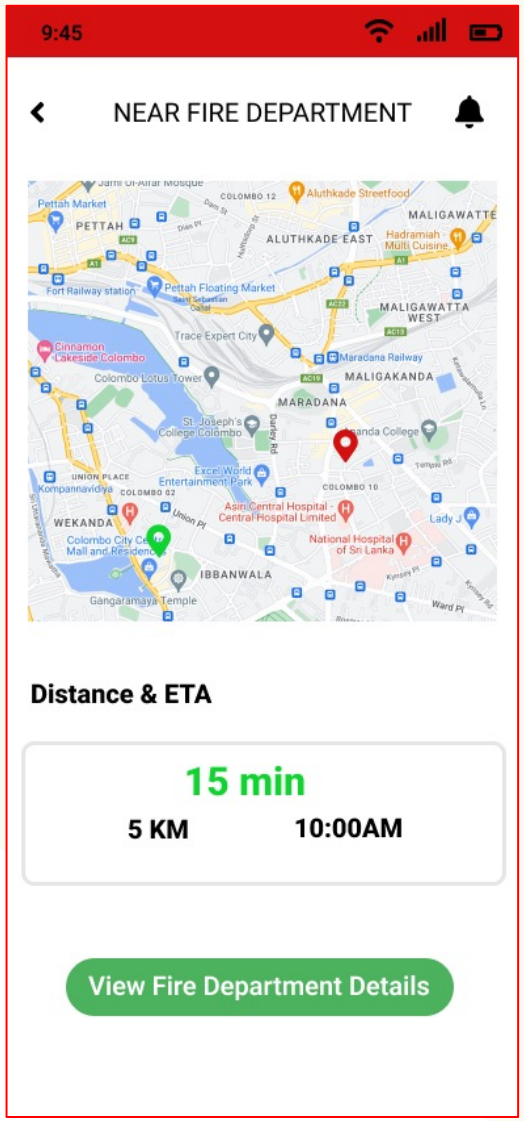


Best Practices of the development
UI/UX Designs

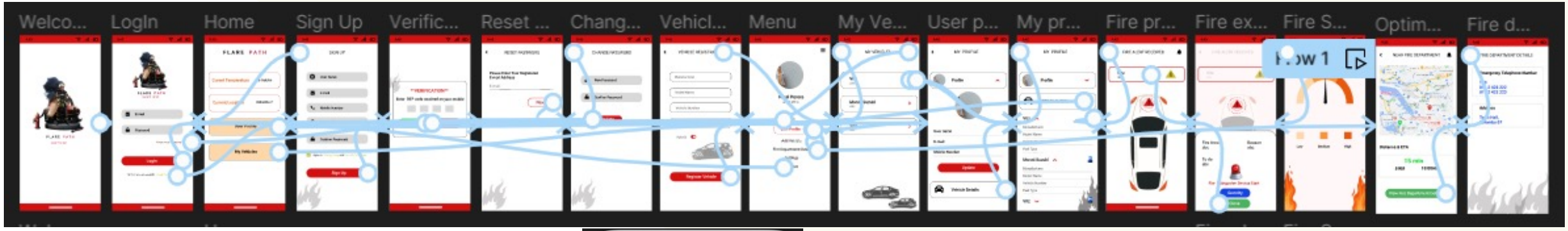


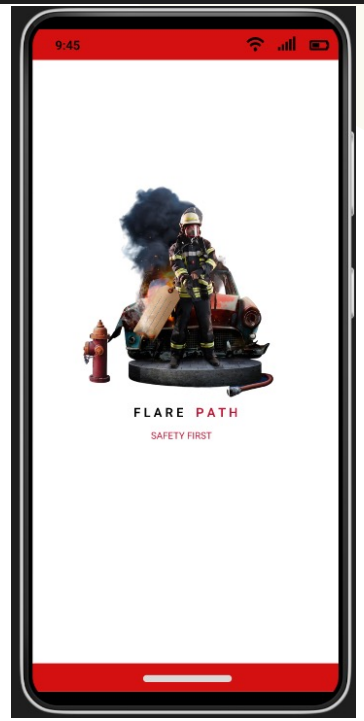






Best Practices of the development UI/UX Designs Flows and Prototype





Tools and Technologies

Hardware Tools

- Arduino UNO
- MQTT Broker
- Smoke Detectors
- Temperature sensors'
- Flame sensors
- Arduino Board



Software Technologies

- Python
- Jupyter Notebook
- Figma
- Flutter
- TensorFlow
- AWS/Azure

Requirement Analysis

Functional Requirements

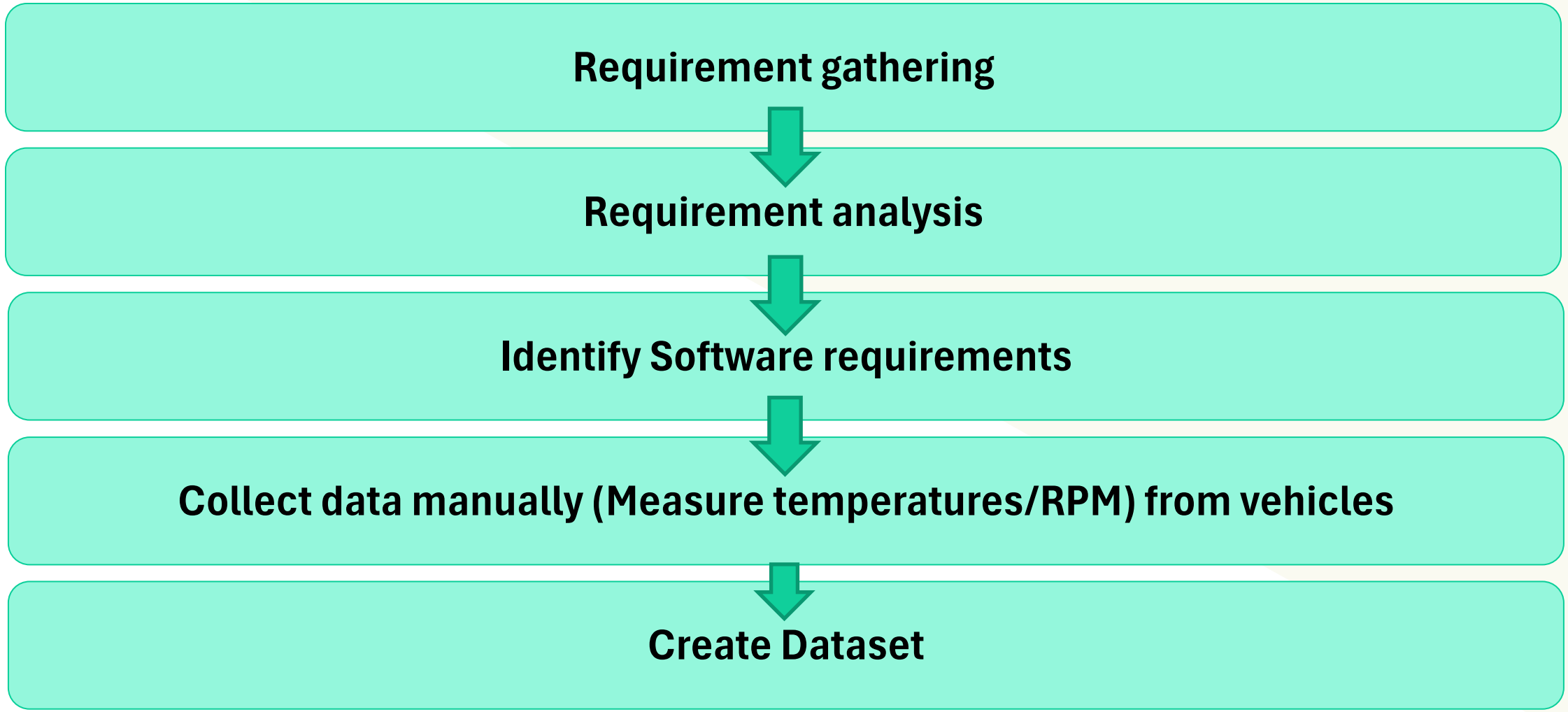
- System should be able to collect and store real-time data under various conditions from IoT-enabled sensors.
- System should be able to send real time fire severity levels to the mobile app without any delay.
- System should be able to send real time fire severity levels to the fire department dashboard without any delay.

Non-Functional Requirements

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

Overall Progress

Completed | Ongoing | To be started



Overall Progress

Design System flow chart

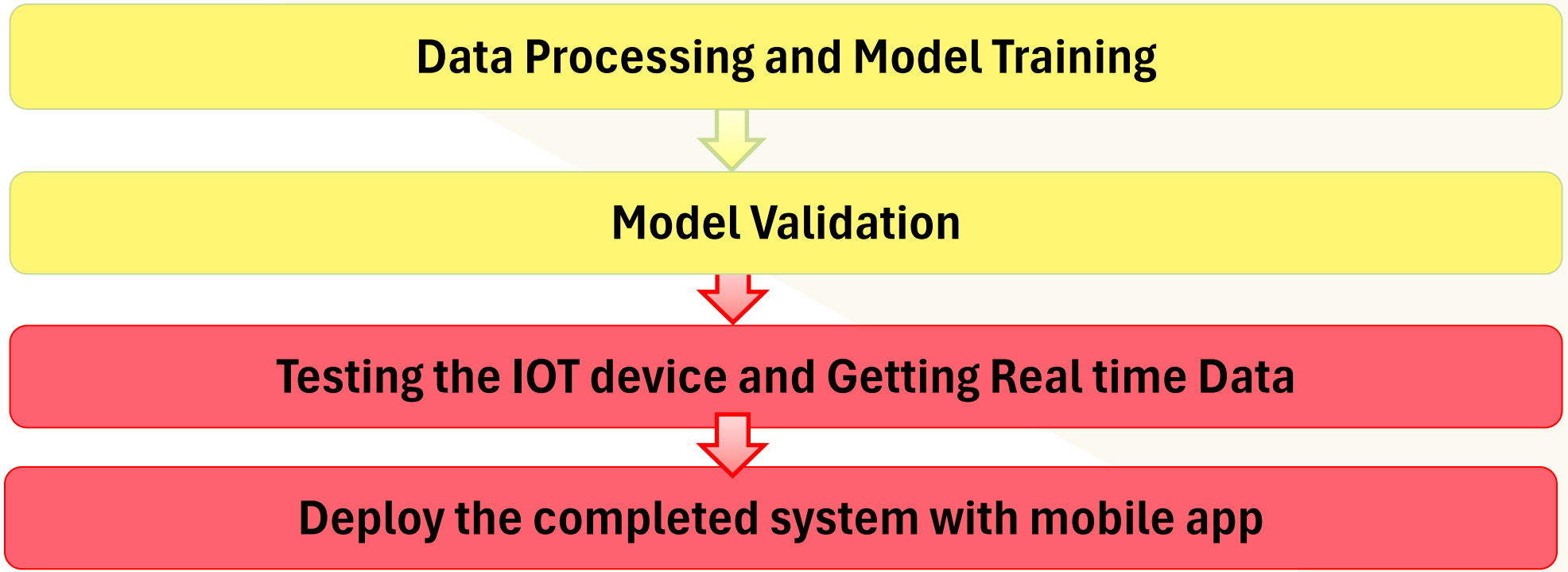
Design Mobile App wireframe

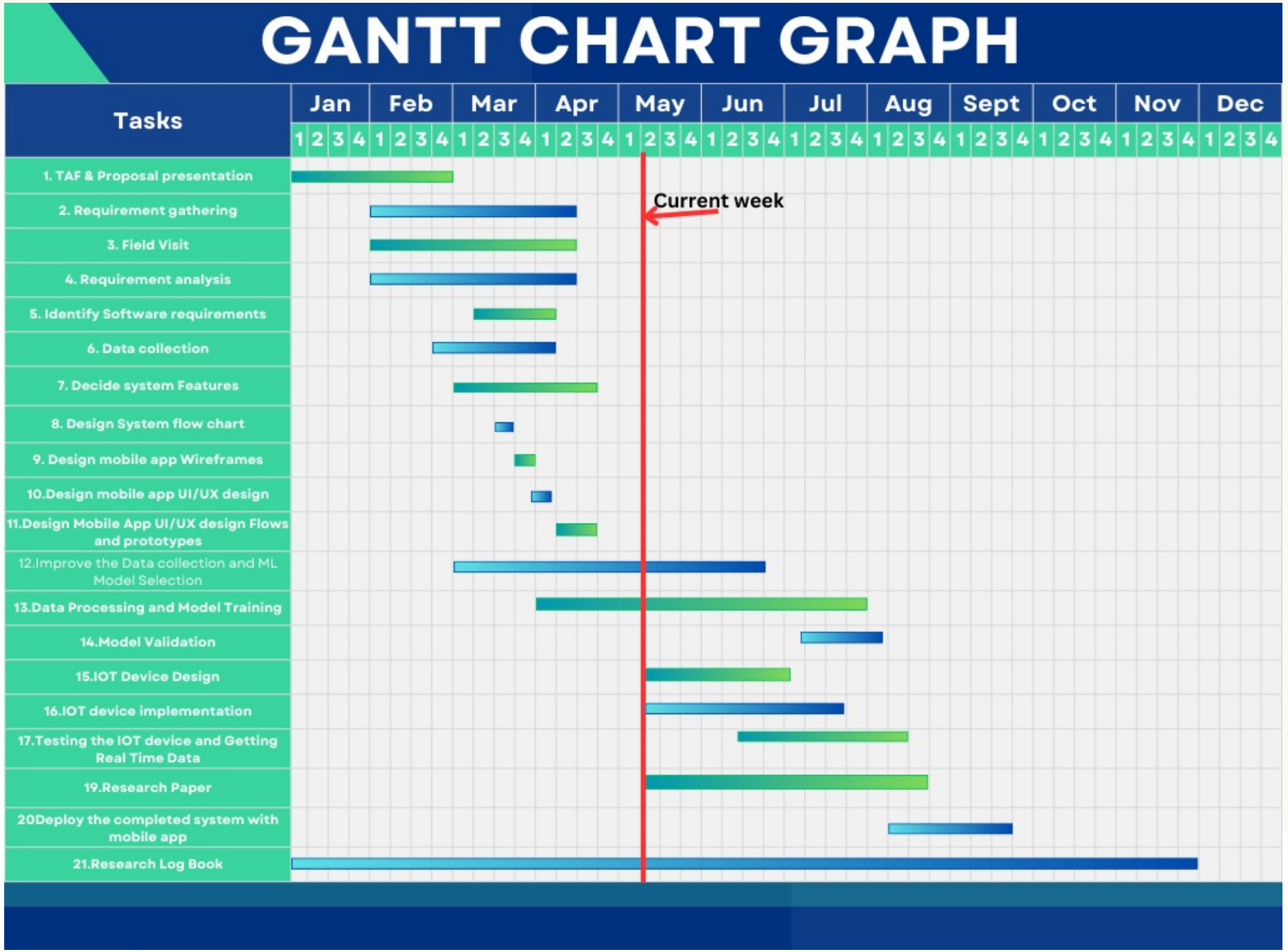
Design Mobile App UI/UX design

Design Mobile App UI/UX design Flows and protypes

Improve the Data collection and ML Model Selection

Overall Progress




GANTT CHART GRAPH
Tasks
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Dec
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
1. TAF & Proposal presentation
2. Requirement gathering
3. Field Visit
4. Requirement analysis
5. Identify Software requirements
6. Data collection
7. Decide system Features
8. Design System flow chart
9. Design mobile app Wireframes
10.Design mobile app UI/UX design
11.Design Mobile App UI/UX design Flows and prototypes
12.Improve the Data collection and ML Model Selection
13.Data Processing and Model Training
14.Model Validation
15.IOT Device Design
16.IOT device implementation
17.Testing the IOT device and Getting Real Time Data
19.Research Paper
20Deploy the completed system with mobile app
21.Research Log Book
Current week

References

[1] Sowah, R., Ampadu, K. O., Ofoli, A., Koumadi, K., Mills, G. A., & Nortey, J. (2016, October). Design and implementation of a fire detection and control system for automobiles using fuzzy logic. *2016 IEEE Industry Applications Society Annual Meeting*. https://doi.org/10.1109/ias.2016.7731880

[2] 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

[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.2018.2875189

Dharmagunawardana W.M.P.I| IT21132346
Specializing in Information Technology

IoT-based Fire Extinguisher Mechanism

Background

- Vehicle fire cause significant damage to properties.
- Current systems do not effectively communicate with emergency services.
- Annually, more than 40 vehicle fires are reported in Sri Lanka.
- Alert the driver about the fire Incidence.
- Introduce the concept of the Internet of Things (IoT) and its growing application to automobile industry.
- Environmental pollution from burning materials and chemicals.

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-based Fire Extinguisher Mechanism

- IOT device that monitor the vehicle
- Alert the driver about Fire incident
- Give the time out to evacuate the vehicle
- Based on the location it sprays the CO2

Completion of the Component

- 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
- IOT device design
- IOT device implementation

Completion of the component

Implementation

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

Fire_IOT.ino

```
#include <WiFi.h> //This library is used to manage the WiFi functionality of Arduino boards

#include <WiFiMulti.h>  // This library is useful when your device might connect to multiple networks over time
                        //allows an ESP32 device to store WiFi access credentials (SSID and password) and attempt to connect to them

#include <PubSubClient.h>   //This library is used for implementing MQTT communication in Arduino sketches.
                            //MQTT (Message Queuing Telemetry Transport) is a popular lightweight messaging protocol

#include <WiFiClientSecure.h> //This library is particularly useful for connecting to APIs and web services that require secure communication,
                              //ensuring data transmitted over the network is encrypted.

#include <DHT.h>  //The library provides methods to initialize the sensor, read temperature, and read humidity.

#define DHT_SENSOR_PIN 21 //This specify the DATA pin number on the microcontroller (21)

#define DHT_SENSOR_TYPE DHT11 //specifies the type of DHT sensor being used

#define Fire_Sensor   15

#define Alarm_Pin   4

DHT dht_sensor(DHT_SENSOR_PIN, DHT_SENSOR_TYPE);


//########################################## wifi_setup ##########################################

//---- WiFi settings
const char* ssid = "Dialog 4G";
const char* password = "L2LAYH89HHL";

```

Output

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

Fire_IOT.ino

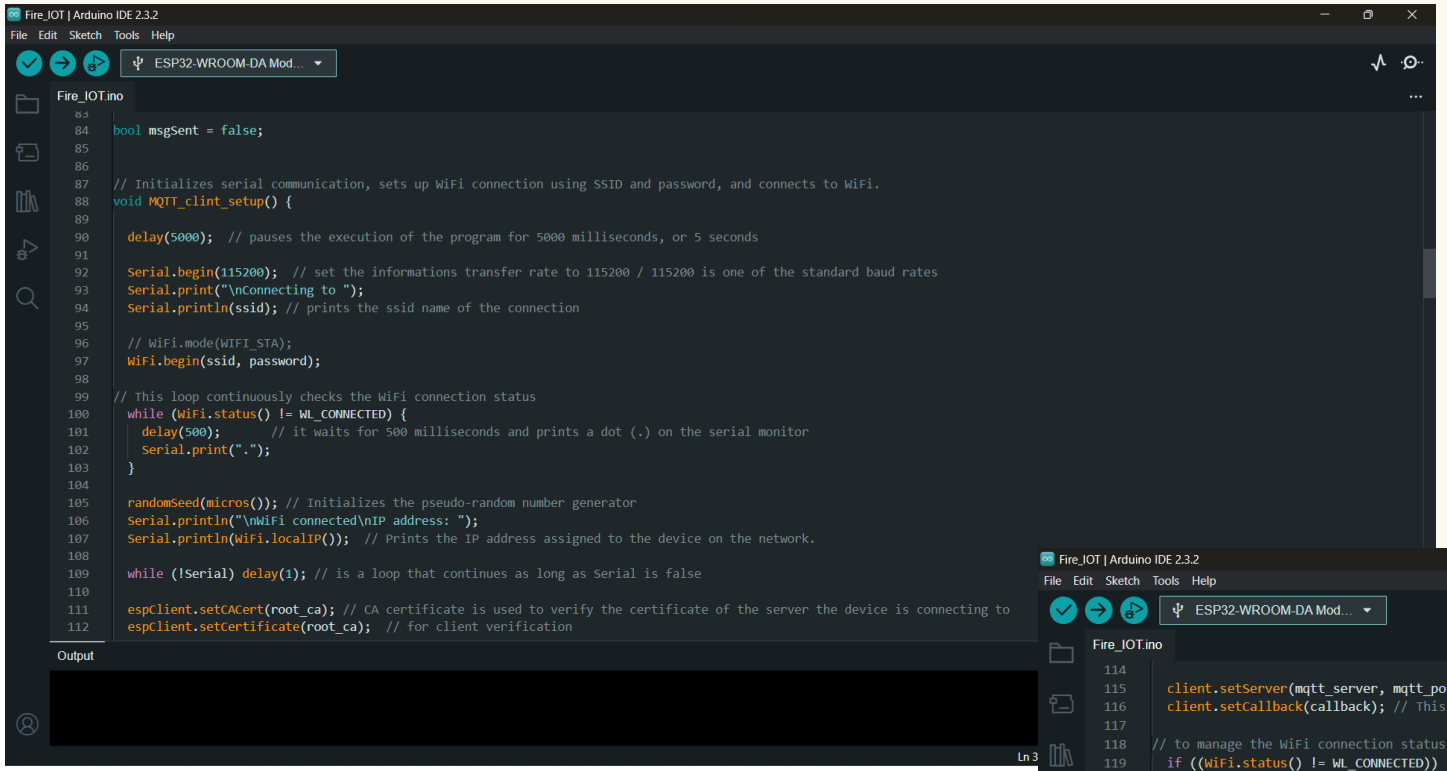
```
const int mqtt_port = 8883; //port number

WiFiClientSecure espClient; // an instance of the WiFiClientSecure class, establish secure SSL/TLS connections over WiFi,

PubSubClient client(espClient); //how to use the PubSubClient library in combination with the WiFiClientSecure class for secure MQTT communication

unsigned long lastMsg = 0;  //used to store the time (in milliseconds) at which the last message was sent


#define MSG_BUFFER_SIZE (50)  //size of the message
char msg[MSG_BUFFER_SIZE];


const char* sensor1_topic = "fire_dep_inform";

const char* root_ca =
  "-----BEGIN CERTIFICATE-----\n"
  "MIIFazCCA1OgAwIBAgIRAIIQz7DSQONZRGPgu2OCiwAwDQYJKoZIhvcNAQELBQAw\n"
  "TzELMAkGA1UEBhMCVVMxKTAnBgNVBAoTIEludGVybmV0IFNlY3VyaXR5IFJlc2Vh\n"
  "cmNoIEdyb3VwMRUwEwYDVQQDEwxJU1JHIFJvb3QgWDEwHhcNMTUwNjA0MTEwNDM4\n"
  "WhcNMzUwNjA0MTEwNDM4WjBPMQswCQYDVQQGEwJVUzEpMCcGA1UEChMgSW50ZXJu\n"
  "ZXQgU2VjdXJpdHkgUmVzZWFyY2ggR3JvdXAxFTATBgNVBAMTDElTUkcgUm9vdCBY\n"
  "MTCCAiIwDQYJKoZIhvcNAQEBBQADggIPADCCAgoCggIBAK3oJHP0FDfzm54rVygc\n"
  "h77ct984kIxuPOZXoHj3dcKi/vVqbvYATyjb3miGbESTtrFj/RQSa78f0uoxmyF+\n"
  "0TM8ukj13Xnfs7j/EvEhmkvBioZxaUpmZmyPfjxwv60pIgbz5MDmgK7iS4+3mX6U\n"
  "A5/TR5d8mUgjU+g4rk8Kb4Mu0UlXjIB0ttov0DiNewNwIRt18jA8+o+u3dpjq+sW\n"
  "T8KOEUt+zwvo/7V3LvSye0rgTBIlDHCNAymg4VMk7BPZ7hm/ELNKjD+Jo2FR3qyH\n"
  "B5T0Y3HsLuJvW5iB4YlcNHlsdu87kGJ55tukmi8mxdAQ4Q7e2RCOFvu396j3x+UC\n"
  "B5iPNgiV5+I3lg02dZ77DnKxHZu8A/lJBdiB3QW0KtZB6awBdpUKD9jf1b0SHzUv\n"
```

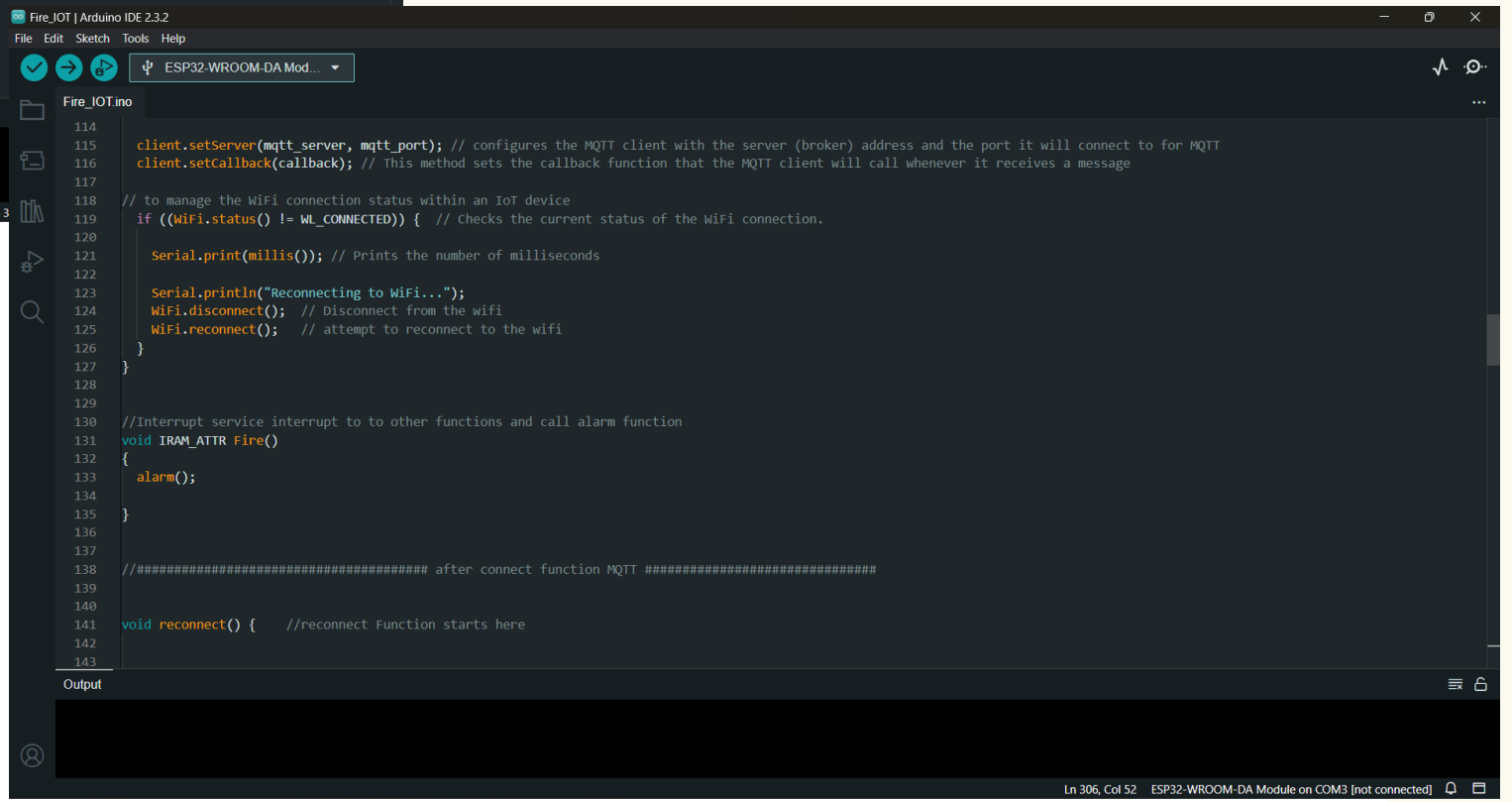
Output

Ln 306, Col 52 ESP32-WROOM-DA Module on COM3 [not connected]

Completion of the component

Implementation





Completion of the component

Implementation

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

Fire_IOT.ino

```
139
140
141  void reconnect() {    //reconnect Function starts here
142
143
144    // Loop until we're reconnected
145    while (!client.connected()) {   // Initiates a loop that continues as long as the MQTT client is not connected.
146      Serial.print("Attempting MQTT connection…");
147      String clientId = "esp-client-";  //  Creates a unique client ID for the MQTT connection.
148      boolean cleanSession = true;
149      clientId += String(random(0xffff), HEX);  // generates a random number between 0 and ffff
150
151
152      // Attempts to connect to the MQTT broker using the generated client ID and stored credentials.
153      if (client.connect(clientId.c_str(), mqtt_username, mqtt_password)) {
154
155        Serial.println("connected");
156
157        client.subscribe(sensor1_topic);  // subscribe the topics here
158
159      } else {    // Provides feedback and manages actions if the connection attempt fails.
160        Serial.print("failed, rc=");
161        Serial.print(client.state()); // prints display the connection failure reason with an error code
162        Serial.println(" try again in 5 seconds");  // Wait 5 seconds before retrying
163        delay(5000);  // pauses the loop for 5000 milliseconds (5 seconds
164      }
165    }
166  }
167
168
```

Output

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

Fire_IOT.ino

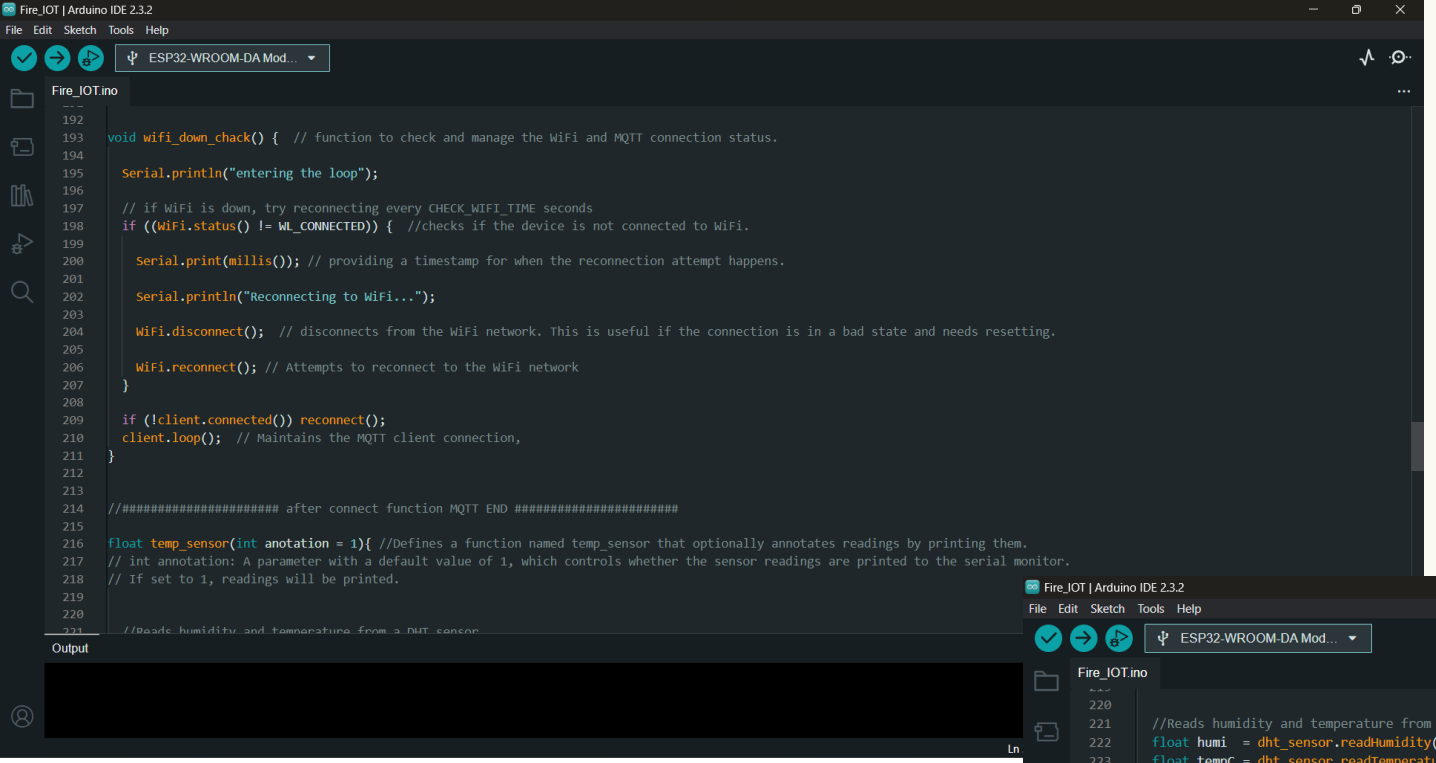
```
167
168
169  void callback(char* topic, byte* payload, unsigned int length) {  // Defines the function that the MQTT client will call upon receiving a message.
170
171    String incommingMessage = ""; // Initializes an empty string to store the incoming message
172
173    for (int i = 0; i < length; i++) incommingMessage += (char)payload[i];  //// Converts the payload from an array to a string.
174
175    Serial.println("Message arrived [" + String(topic) + "]" + incommingMessage); // Logs the received message along with its topic
176
177    String intopic = String(topic); // Converts the topic from a C-style string (char*) to an Arduino String object.
178
179    if_message_resv(intopic, incommingMessage); // Further processes the received message by calling another function
180  }
181
182
183  //===================================== publising as string
184  void publishMessage(const char* topic, String payload, boolean retained) {  // Defines a function to publish messages to an MQTT topic
185
186    if (client.publish(topic, payload.c_str(), true)) // Attempts to publish a message to the specified MQTT topic.
187
188      Serial.println("Message published [" + String(topic) + "]: " + payload); // Logs the successful publication of the message.
189  }
190
191
192
193  void wifi_down_chack() {  // function to check and manage the WiFi and MQTT connection status.
194
195    Serial.println("entering the loop");
196
```

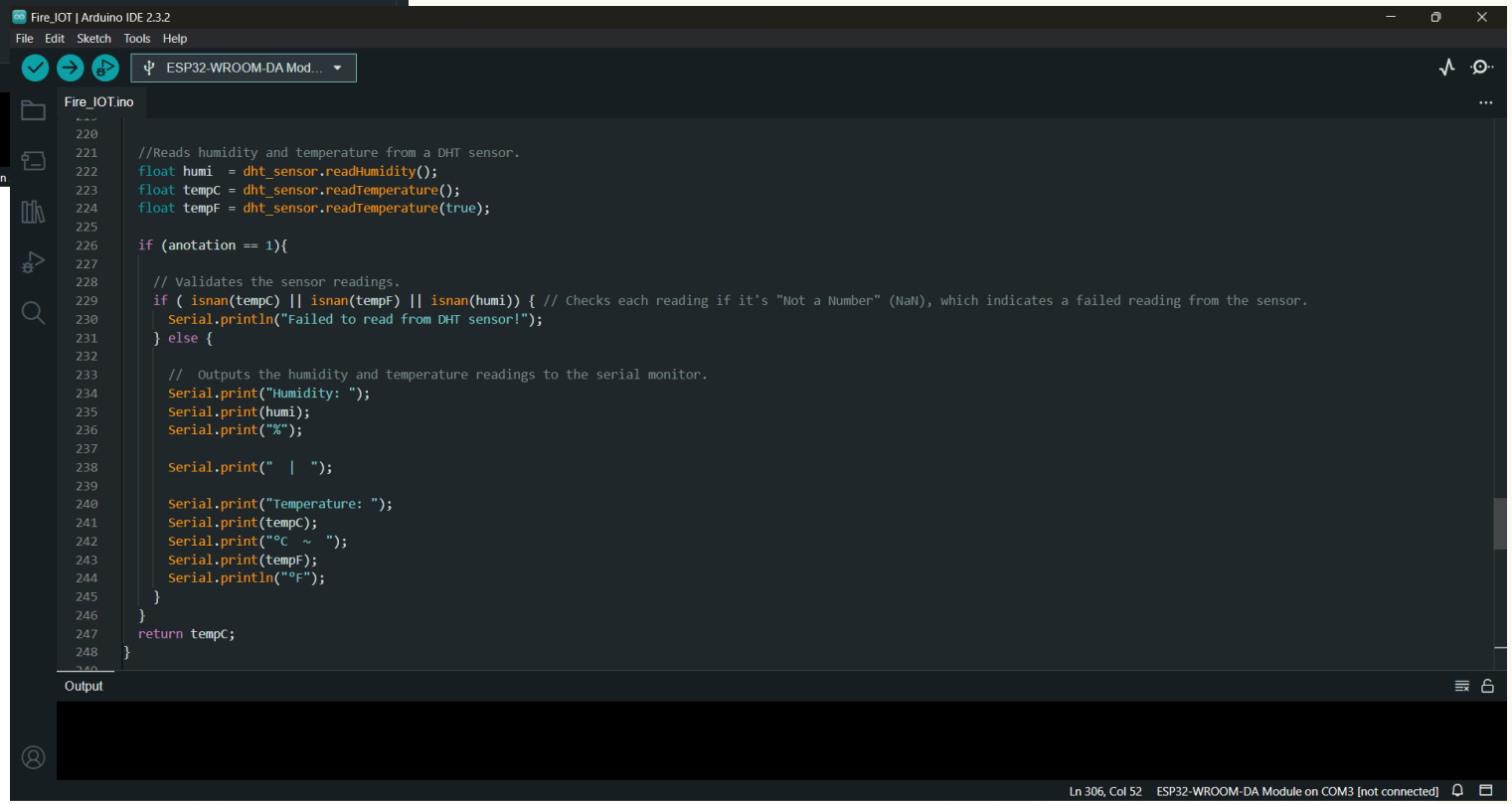
Output

Ln 306, Col 52 ESP32-WROOM-DA Module on COM3 [not connected]

Completion of the component

Implementation





Completion of the component

Implementation

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

Fire_IOT.ino

```
250
251  //############################## if msg reseved then do ###################
252  // This void is called every time we have a message from the broker
253
254  void if_message_resv(String topic, String msg) {
255
256
257    //Outputs the received message along with the topic to the serial monitor.
258    Serial.print("serial workssssssssssssss  - ");
259    Serial.print(topic); //print the topic of the serial
260    Serial.print("  -  ");
261    Serial.println(msg);
262  }
263
264  //#############################################################################
265
266
267  // Main Finctions, all the arduino logics will call through this functions / setup() & loop() functions
268  void setup() {
269
270    MQTT_clint_setup();
271    dht_sensor.begin(); //Initializes the DHT sensor for reading temperature and humidity data.
272
273    pinMode(Fire_Sensor, INPUT_PULLUP);
274
275    attachInterrupt(Fire_Sensor, Fire, RISING); //Rising edge -buttom push
276
277    pinMode(Alarm_Pin, OUTPUT);
278
279  }
```

Output

Fire_IOT | Arduino IDE 2.3.2

File Edit Sketch Tools Help

ESP32-WROOM-DA Mod...

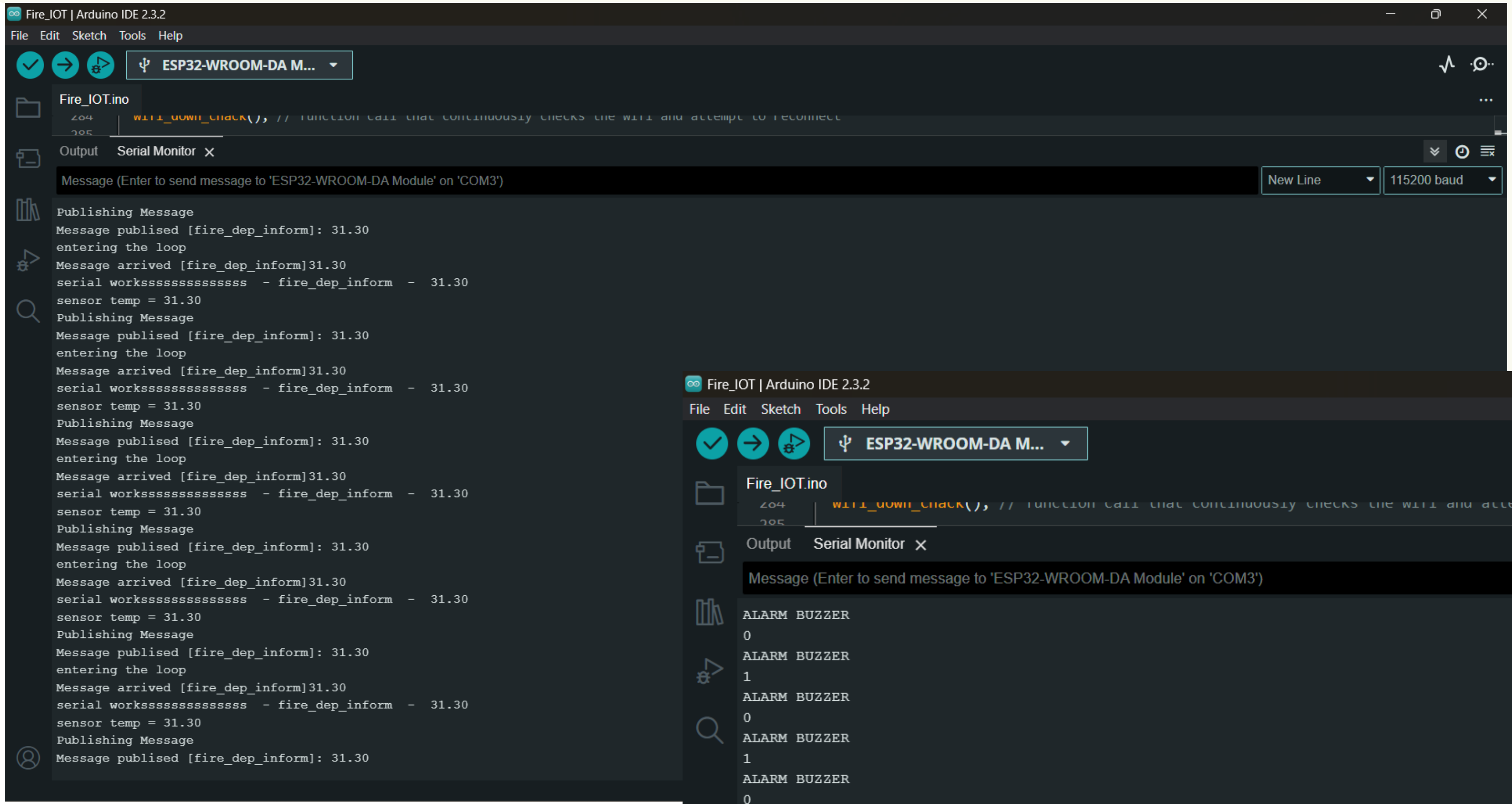
Fire_IOT.ino

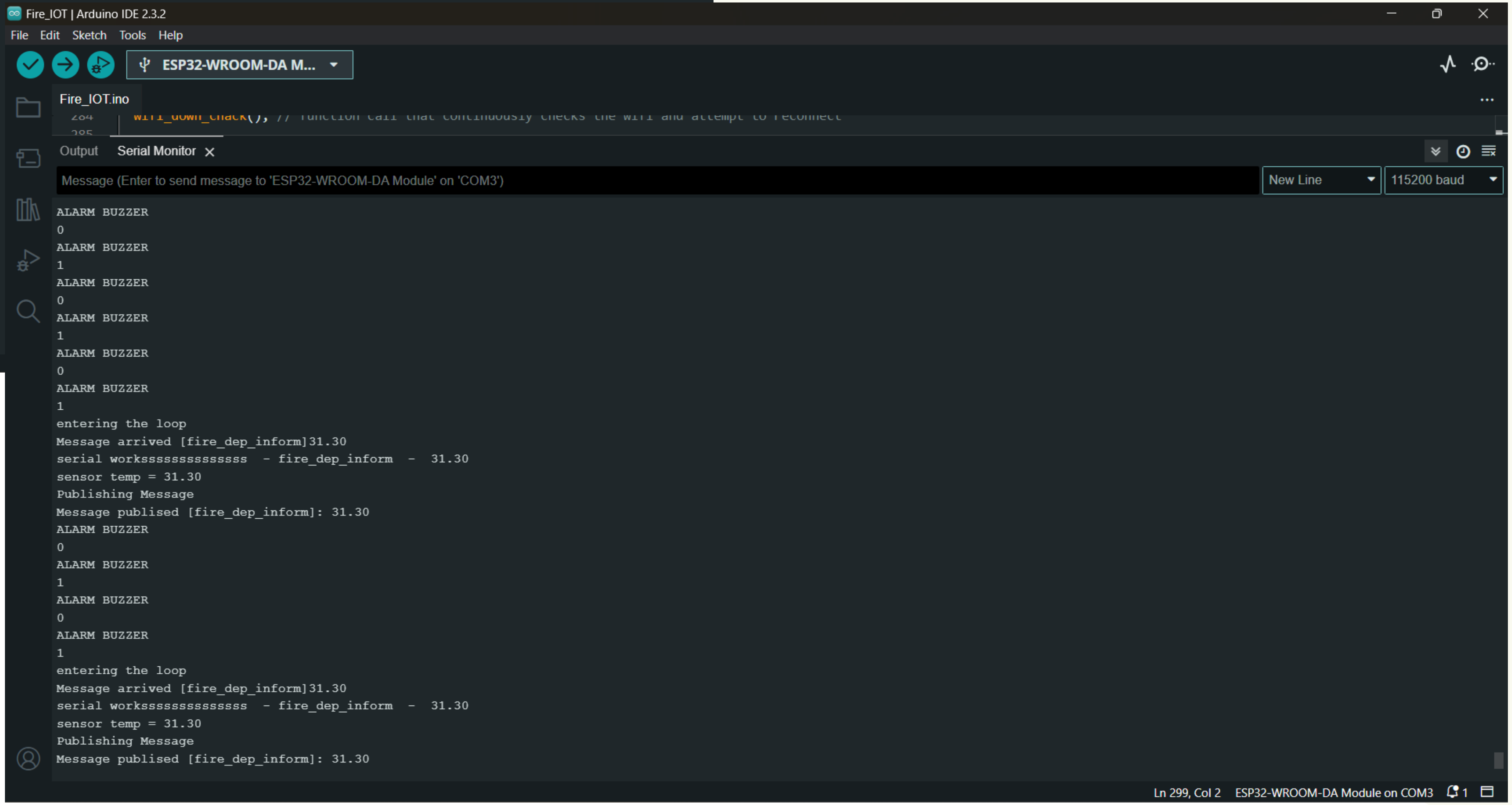
```
280
281  void loop() {
282
283
284    wifi_down_chack(); // function call that continuously checks the wifi and attempt to reconnect
285
286    ////////////////////////////////////////////
287
288    float room_temp = temp_sensor(0);
289
290    Serial.print("sensor temp = ");
291    Serial.println(room_temp);
292
293    dtostrf(room_temp, 4, 2, msg);  // Converts the floating-point number room_temp into a string format to have a width of 4 characters and 2 decimal places.
294                                    // The result is stored in msg, which is an array of characters.
295
296    Serial.println("Publishing Message ");
297    publishMessage(sensor1_topic, msg, true);
298    delay(1000);  //Pauses the execution of the loop() function for 1000 milliseconds (or 1 second), creating a delay between each execution of the loop.
299  }
300
301
302  void alarm(){
303
304    Serial.println("ALARM BUZZER");
305    Serial.println(digitalRead(Alarm_Pin));
306    digitalWrite(Alarm_Pin, !digitalRead(Alarm_Pin));
307
308  }
309
```

Output

Ln 306, Col 52 ESP32-WROOM-DA Module on COM3 [not connected]

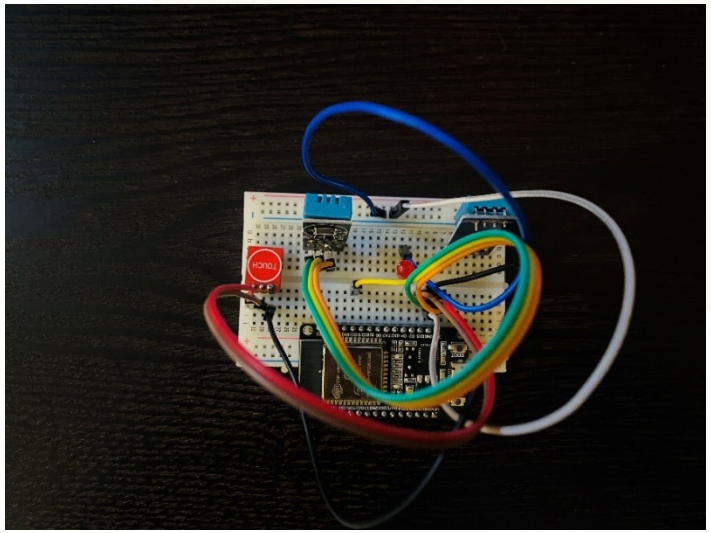
Completion of the component
Implementation

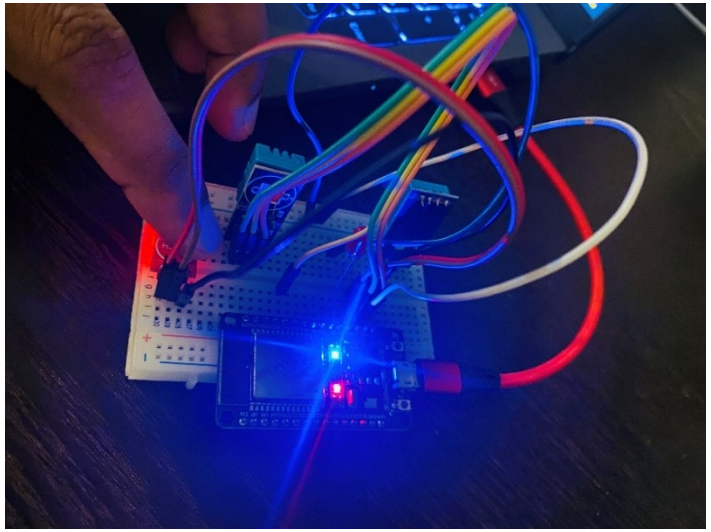


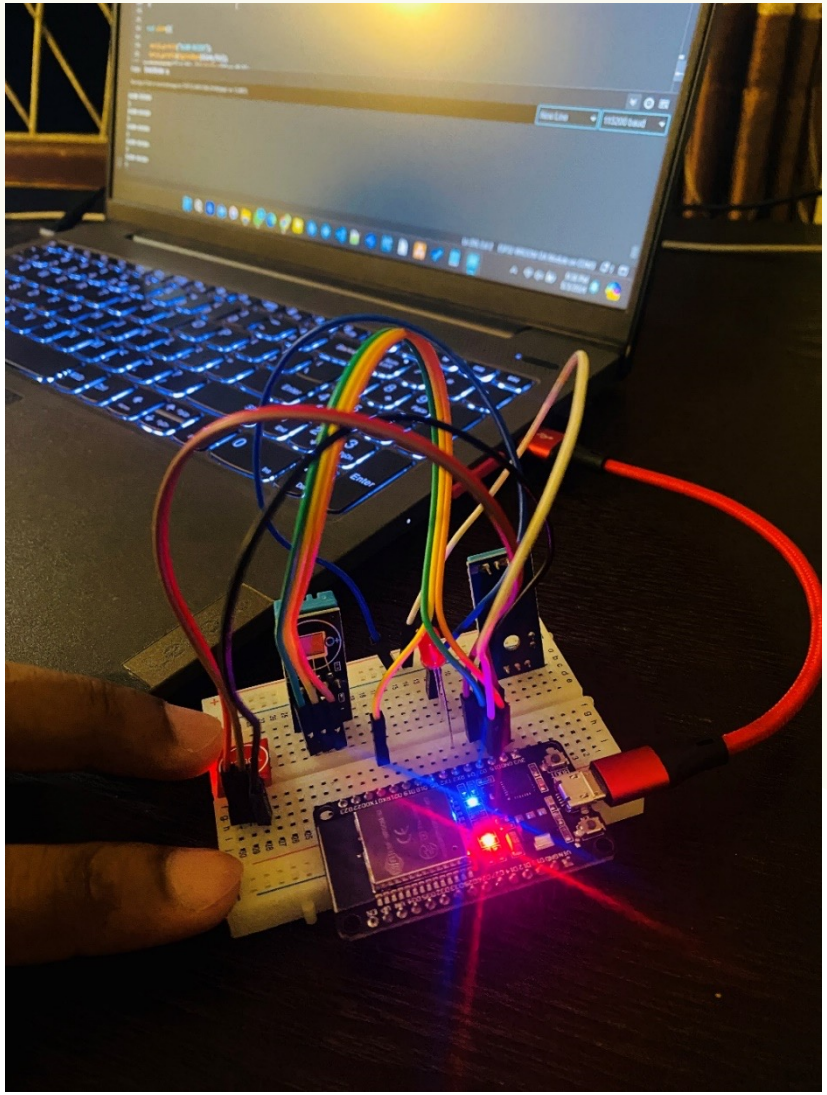


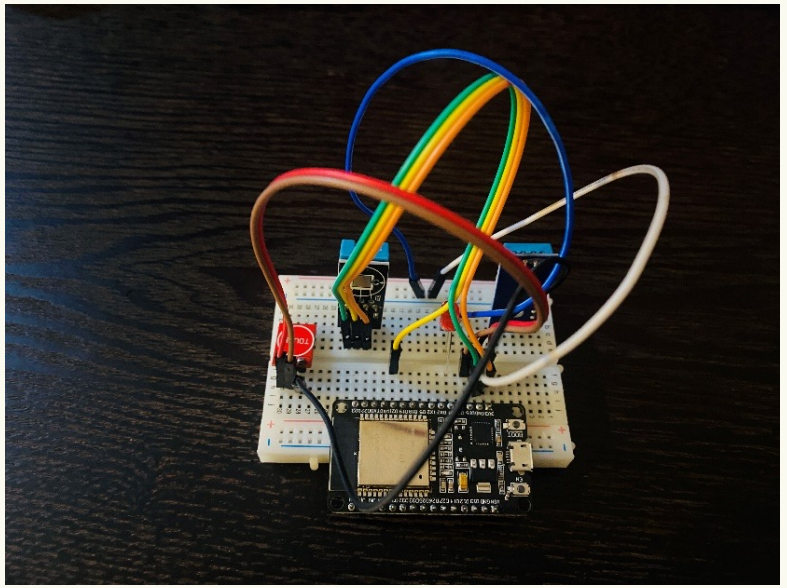
Completion of the component

Implementation

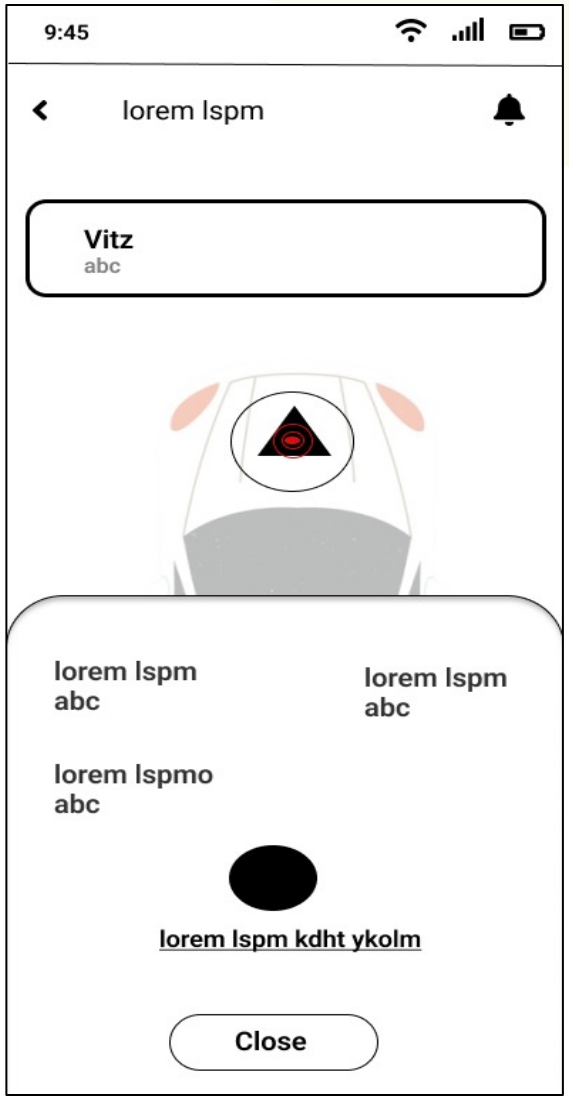


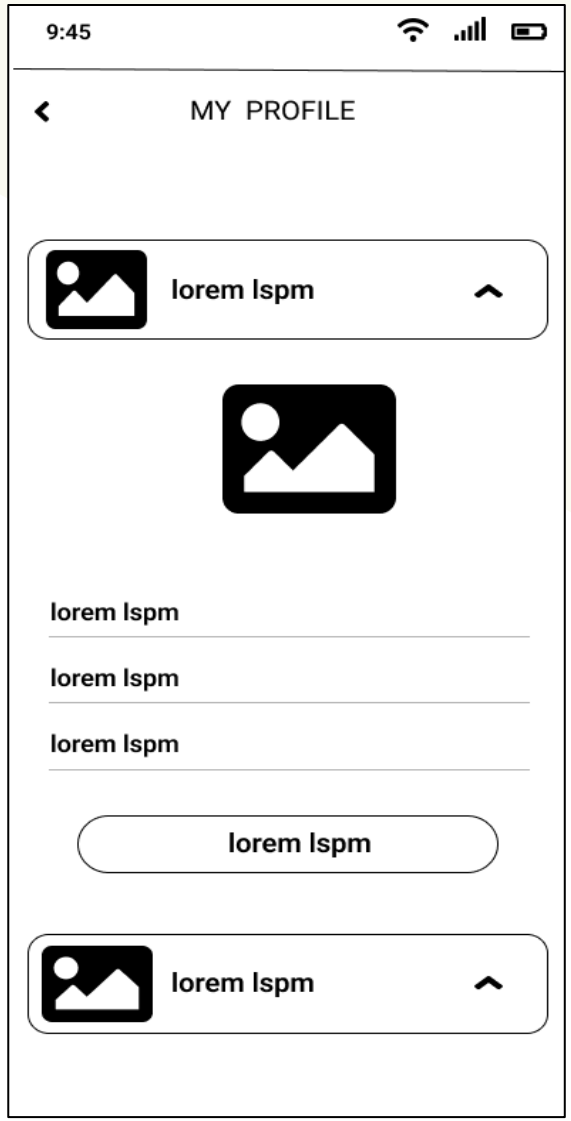


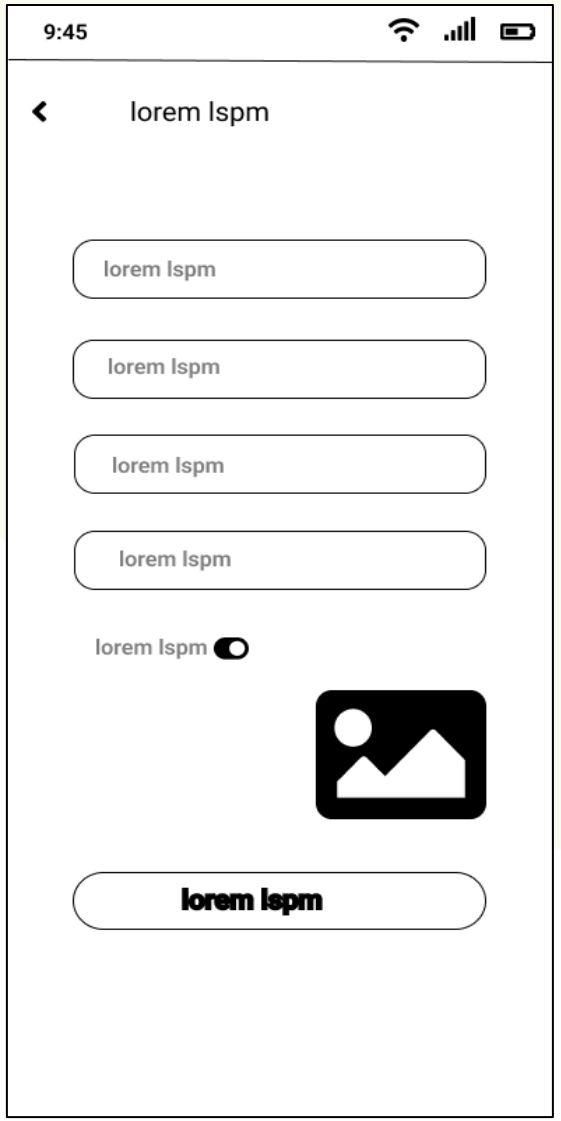


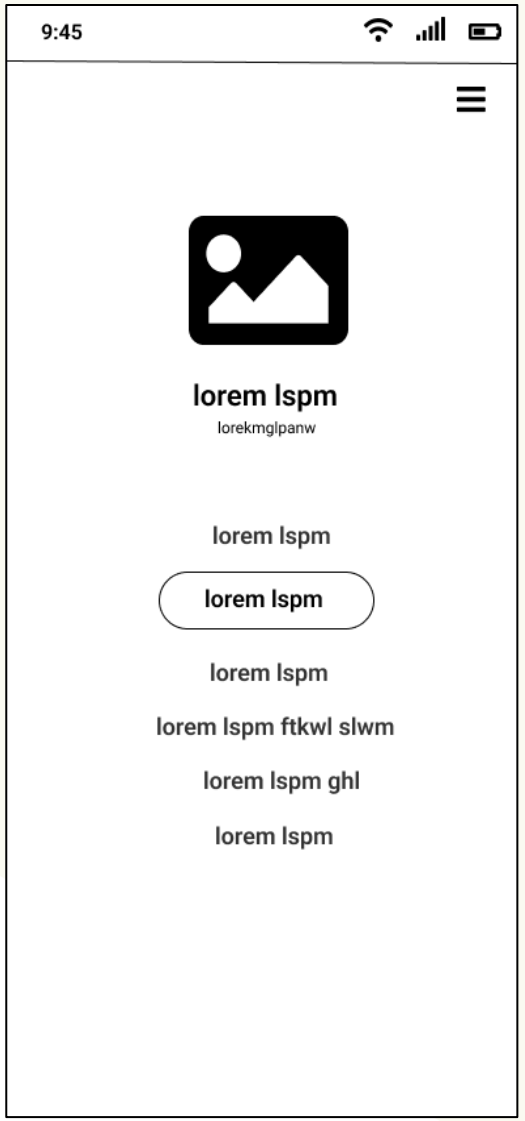


Wireframes


9:45
lorem lspm
Vitz
abc
lorem lspm
abc
lorem lspm
abc
lorem lspmo
abc
lorem lspm kdht ykolm
Close

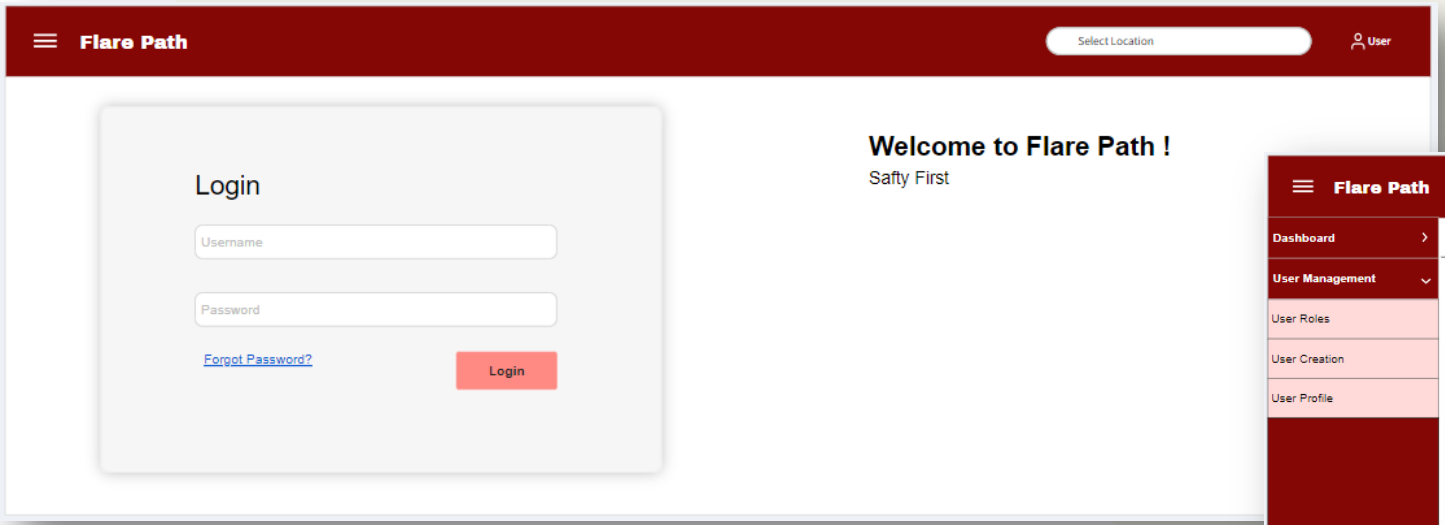

9:45
MY PROFILE
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm

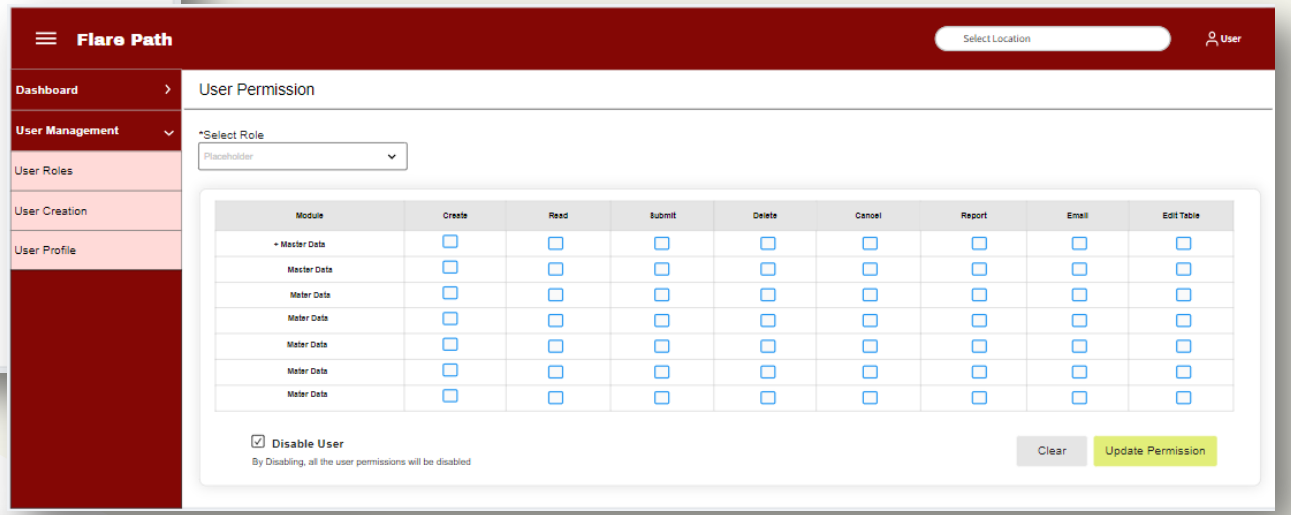

9:45
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm
lorem lspm

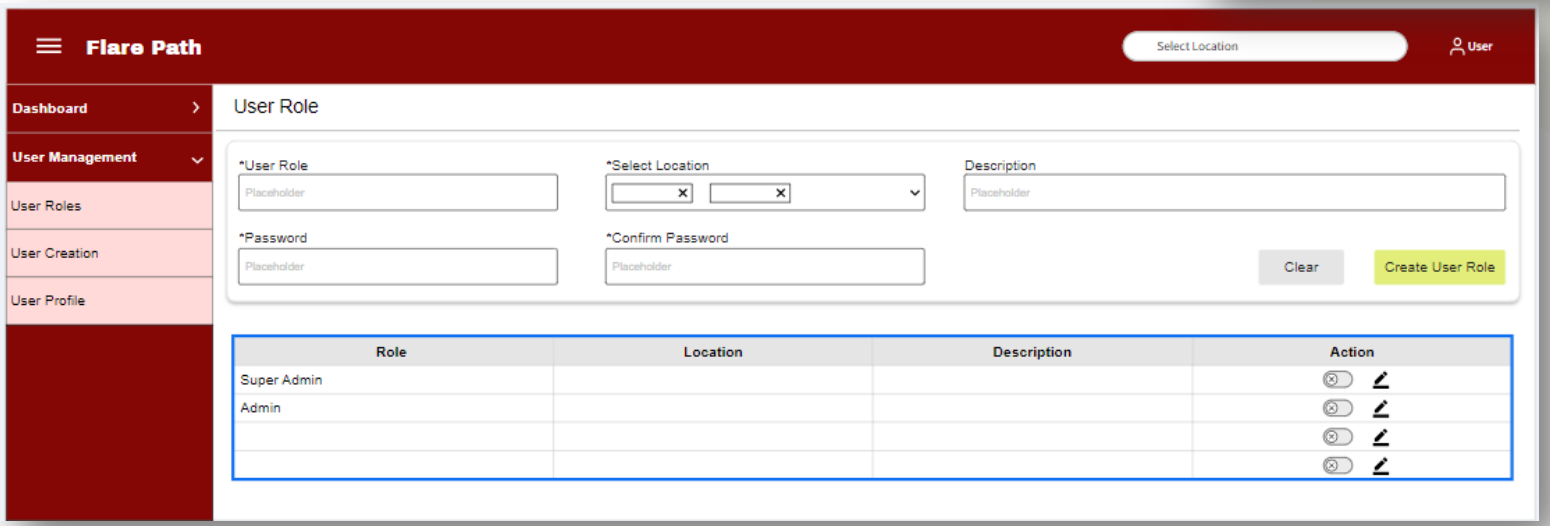

9:45
lorem lspm
lorekmglpanw
lorem lspm
lorem lspm
lorem lspm ftkwl slwm
lorem lspm ghl
lorem lspm

Best Practices of the development

Wireframes

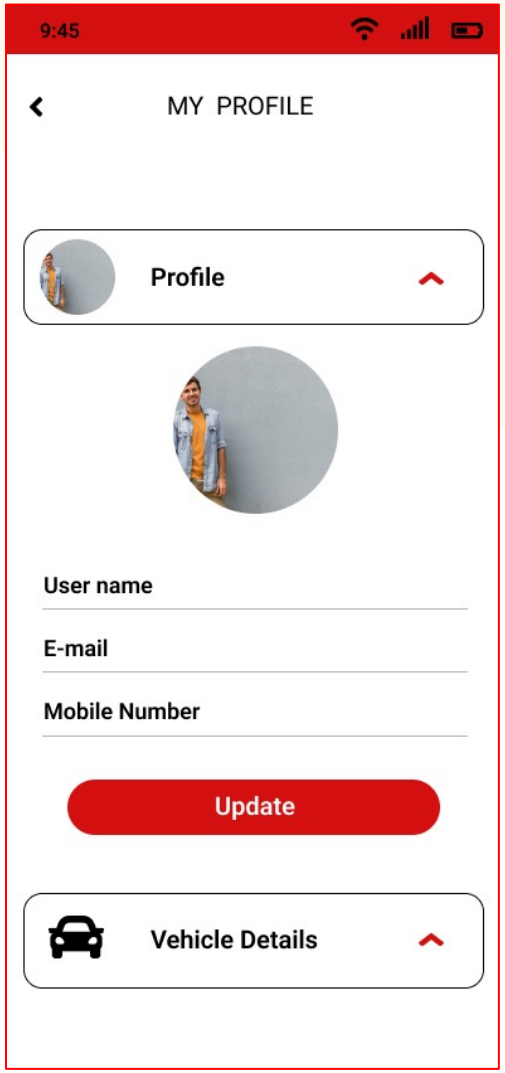


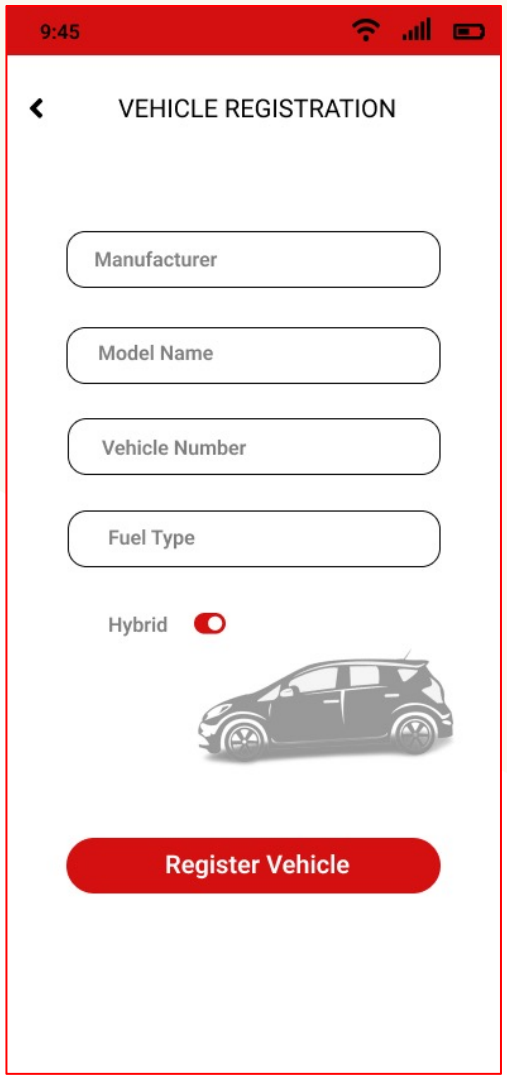


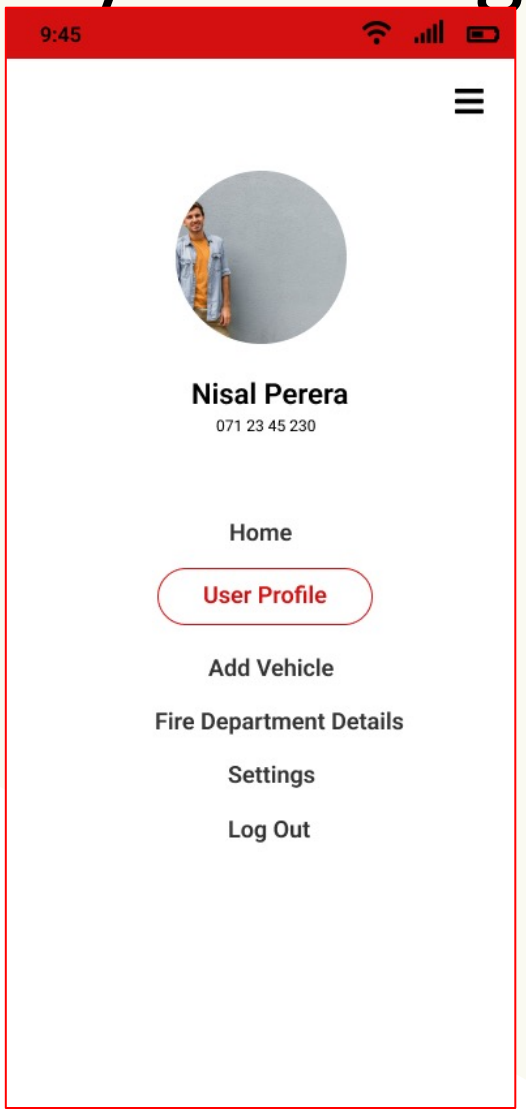


Best Practices of the development

UI/UX Designs







Tools and Technologies

Hardware Tools

- ESP32
- Temperature Sensor
- Flame Detector
- Speaker
- Fire Extinguisher

Software Technologies

- Micro Python
- c/c++
- Arduino IDE
- Pycharm
- Flutter
- AWS/Azure

Requirement Analysis

Functional Requirements

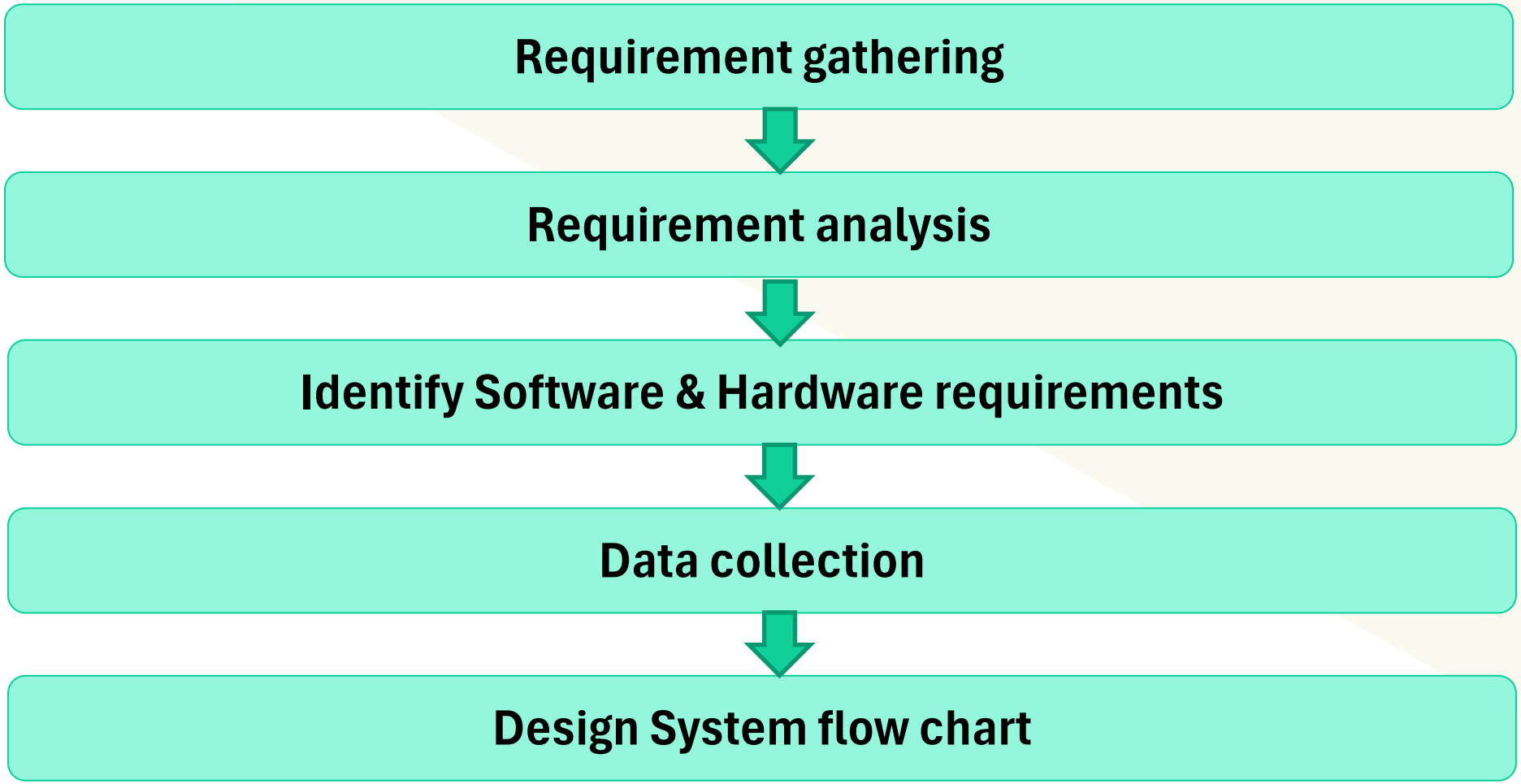
- The system must transmit the collected data securely to a cloud.
- The system must provide real-time feedback to drivers based on the situation.
- The system must include a user-friendly interface.
- The system must be capable of integrating with existing vehicle systems.
- The Fire extinguishers must 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.

Overall Progress

Completed | Ongoing | To be started



Overall Progress

Design mobile app Wireframe

Design mobile app frontend

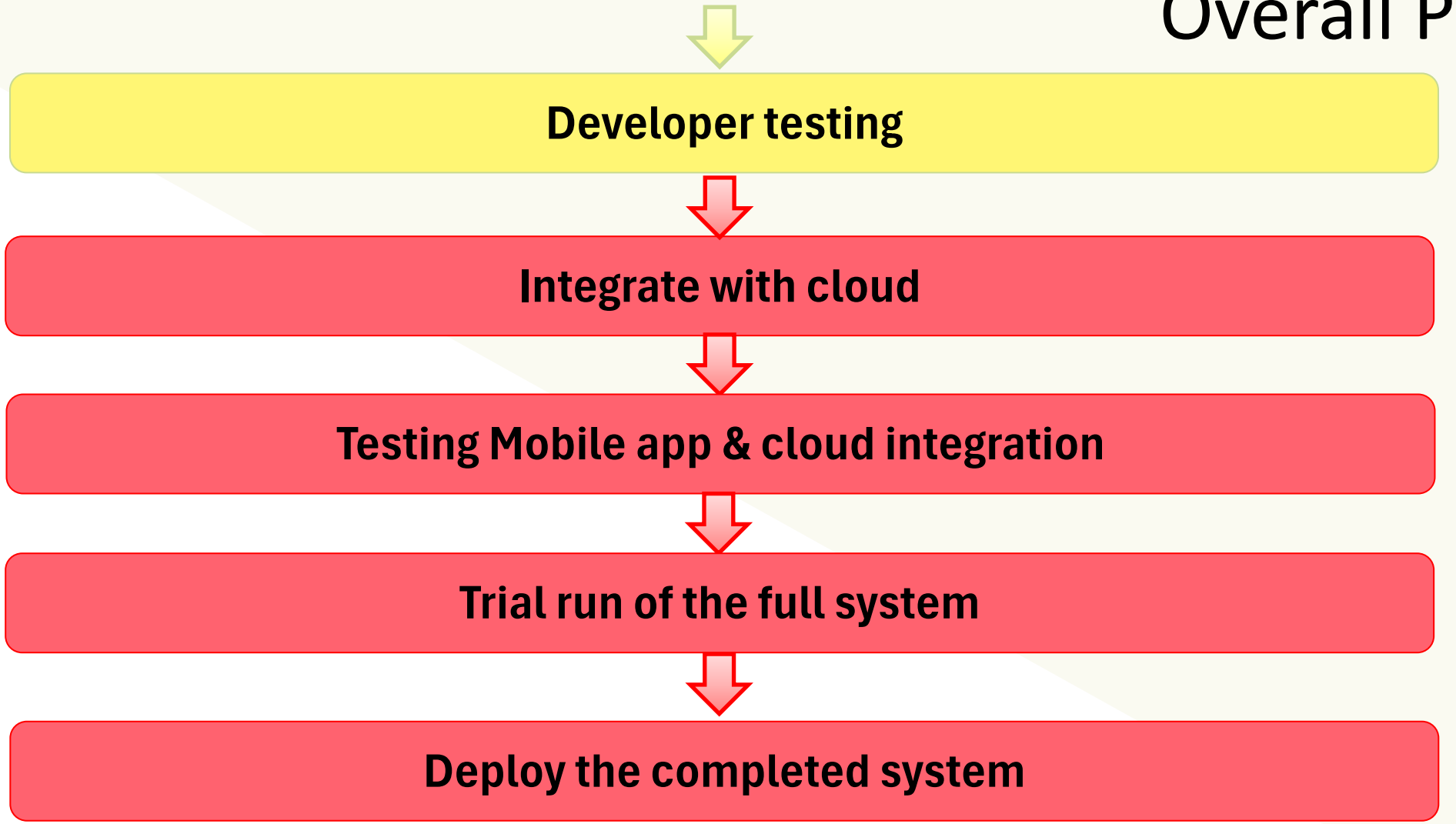
IOT device design

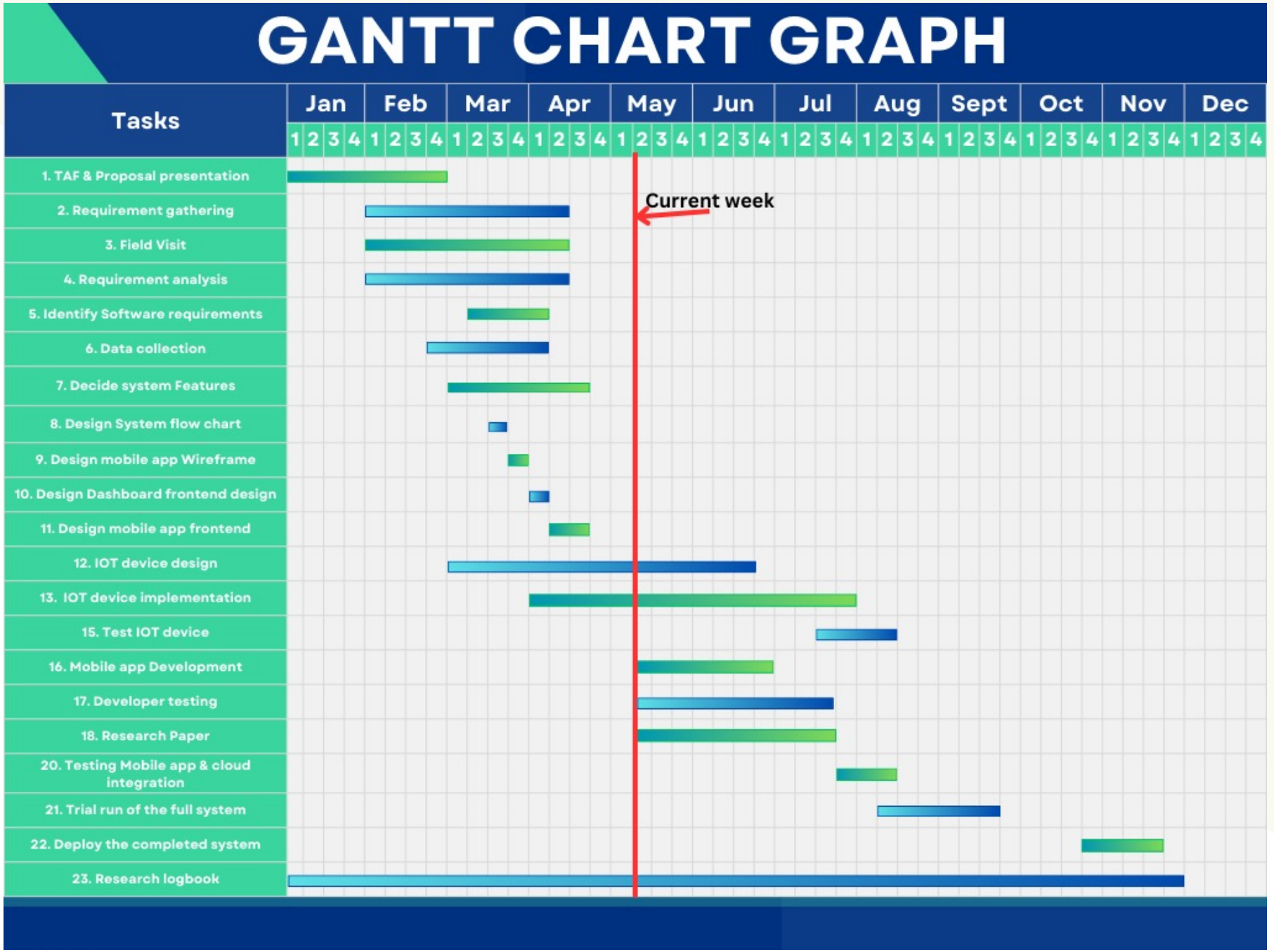
IOT device implementation

Test IOT device

Mobile app Development

Overall Progress




GANTT CHART GRAPH
Tasks
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Dec
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
1. TAF & Proposal presentation
2. Requirement gathering
3. Field Visit
4. Requirement analysis
5. Identify Software requirements
6. Data collection
7. Decide system Features
8. Design System flow chart
9. Design mobile app Wireframe
10. Design Dashboard frontend design
11. Design mobile app frontend
12. IOT device design
13. IOT device implementation
15. Test IOT device
16. Mobile app Development
17. Developer testing
18. Research Paper
20. Testing Mobile app & cloud integration
21. Trial run of the full system
22. Deploy the completed system
23. Research logbook
Current week

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

Background

- Fire departments often use manual processes like phone call verification, causing delays during emergencies.
- Delays in emergency responses waste critical time, reducing service efficiency and potentially increasing harm and damage.
- Callers receive no real-time updates on help arrival, increasing their anxiety during crises.
- There's an urgent need for a modernized system with advanced routing and real-time tracking for fire trucks in urban areas.

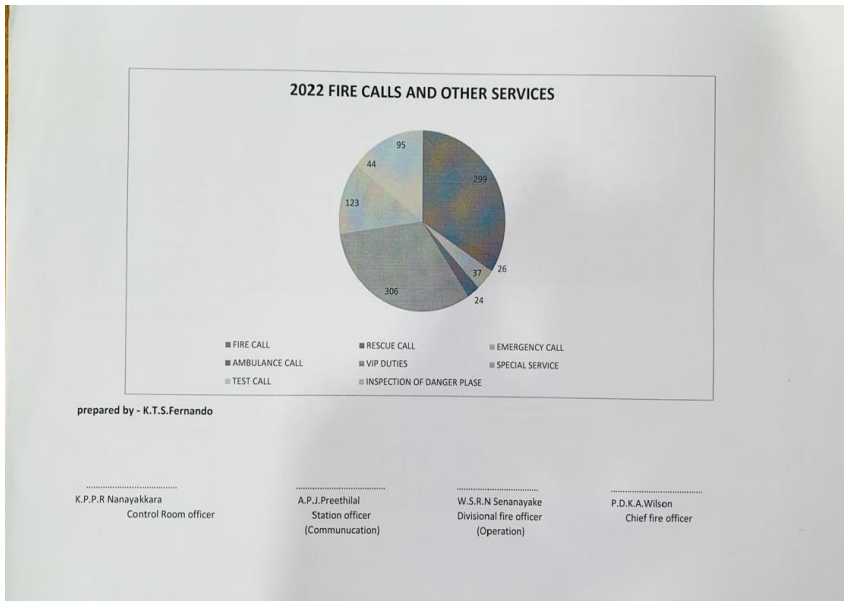
Research Problem

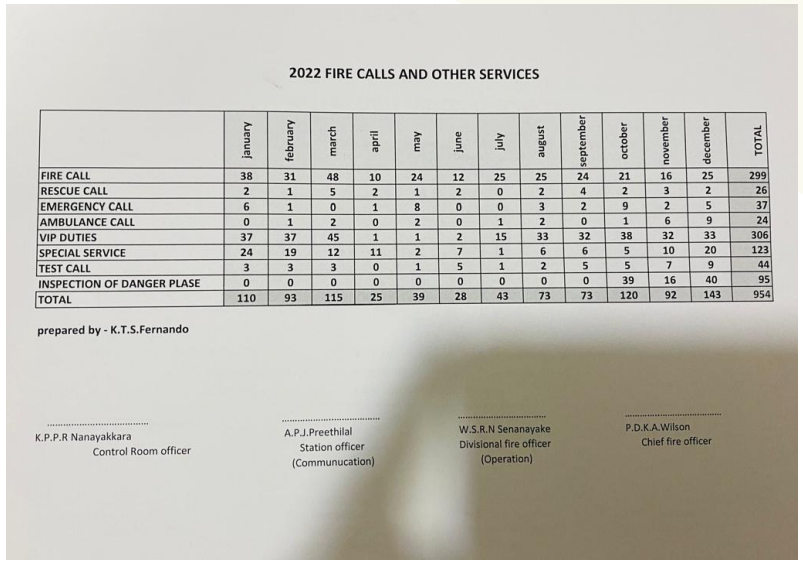
Inefficiency of Manual Call Verification

Lack of Real-Time Response Information

Caller Helplessness During Emergencies

Research Problem





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)

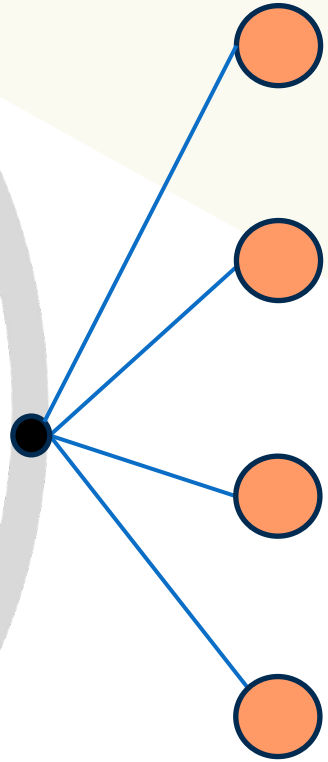
P.D.K.A.Wilson
Chief fire officer

Research Gap

- Essential for reducing response times in vehicle fire emergencies.
- Absence of real-time AI 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 with an AI 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.**
- **Use AI to determine and communicate available firefighting resources to fire departments.**
- **Create an model to inform vehicle owners about nearby fire extinguishers using AI and local data.**

Completion of the Component

- 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 wireframe
- Design Active Incidents wireframe
- Design Active Incidents view wireframe
- Design Logs wireframe
- Design Dashboard frontend design
- Design Active Incidents frontend design
- Design Active Incidents view frontend design
- Design Logs frontend design
- Nearest fire station calculator

Gathered Data (Fire Department)

vehicle_registered_	department_regis	station	vehicle_type	fuel_capacity_lite	fuel_consumption_Km/	approved_monthly_fuel_quantity_lite	water_capacity_heigh	foam_capacity_heigh	max_height_lif	none_user_vehicle
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 Control	100	5	unlimited	N/A	N/A	N/A	Y
LF-1114	144	H	Mobile Workshop	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 Rescue	180	6	150	N/A	N/A	N/A	Y

Fire_departments_Colombo | Total_vehicle | Vehicle_details

Gathered Data (RDA)

ROUTE NO	ROAD NAME	ALL LENGTHS ARE IN KILOMETRES							
		FROM	TO	PARTIAL ROAD LENGTH	TOTAL ROAD LENGTH	EE_DIVISION	CER_NAME	PROVINCE	SUB TOTAL OF PARTIAL ROAD LENGTHS
	(1.0) WESTERN PROVINCE								
	(1.1) COLOMBO								
	(1.1.1) AVISSAWELLA								
B030.10	Approach Roads to Public Buildings (Avissawella)	0.00	0.80	0.80	0.80	Avissawella	COLOMBO	WESTERN	
B123	Galagedera - Horana	0.00	19.05	19.05	19.05	Avissawella	COLOMBO	WESTERN	
B145	Hanwella Junction Road	0.00	0.80	0.80	0.80	Avissawella	COLOMBO	WESTERN	
B188	Kaluaggala - Labugama	0.00	14.00	14.00	14.00	Avissawella	COLOMBO	WESTERN	
B239	Kottawa - Talagala	0.00	15.48	15.48	15.48	Avissawella	COLOMBO	WESTERN	
B267	Mampe - Kottawa	0.00	6.23	6.23	6.23	Avissawella	COLOMBO	WESTERN	
B285	Meepe - Ingiriya	0.00	18.09	18.09	18.09	Avissawella	COLOMBO	WESTERN	
B291	Mirihana - Udahamulla - Nawinna	0.00	3.38	3.38	3.38	Avissawella	COLOMBO	WESTERN	
B335	Old Bazaar Road, Maharagama	0.00	0.80	0.80	0.80	Avissawella	COLOMBO	WESTERN	
B354	Panagoda - Henpita	0.00	6.44	6.44	6.44	Avissawella	COLOMBO	WESTERN	
B367	Piliyandala - Maharagama	0.00	7.37	7.37	7.37	Avissawella	COLOMBO	WESTERN	
B389	Ratmalana - Mirihana	4.83	8.05	3.22	8.05	Avissawella	COLOMBO	WESTERN	
B426	Tummodera - Puwakpitiya	0.00	9.65	9.65	9.65	Avissawella	COLOMBO	WESTERN	**105.31**
	(1.1.2) COLOMBO MUNICIPAL COUNCIL (C.M.C)								
B084	Colombo - Horana	0.00	28.01	28.01	28.01	C.M.C.	COLOMBO	WESTERN	**28.01**
	(1.1.3) COLOMBO								
B011	Allan Avenue, Dehiwela	0.00	1.96	1.96	1.96	Colombo	COLOMBO	WESTERN	

Completion of the Component

Nearest fire station calculator

```
[1]: #pip install googlemaps==4.10.0
```

```
[2]: import googlemaps
     from datetime import datetime
```

```
[3]: def get_distance_and_time(origin, destination, api_key):

         api_key = 'AIzaSyA_ZSQQ7qESG6TPvKviBf0XUIlIcGd84I4'

         # Create a Google Maps client instance
         gmaps = googlemaps.Client(key=api_key)

         # Get the current date and time
         now = datetime.now()
         # Request distance matrix data for specified origin and destination
         directions_result = gmaps.distance_matrix(origins=origin,
                                                   destinations=destination,
                                                   mode="driving",
                                                   departure_time=now)

         if directions_result['rows'][0]['elements'][0]['status'] == 'OK':
             distance = directions_result['rows'][0]['elements'][0]['distance']['text']
             duration = directions_result['rows'][0]['elements'][0]['duration']['text']
             return distance, duration
         else:
             return None, None
```

Completion of the Component

Nearest fire station calculator

```
[4]: def point_to_point_shorter_time_and_distance_calculator(origin_la , origin_ln , dies_la , dies_ln):

    api_key = 'AIzaSyA_ZSQQ7qESG6TPvKviBf0XUIlIcGd84I4'

    # origin_la , origin_ln = 6.784841,79.932660
    # dies_la , dies_ln = 6.795540,79.940790

    # Assemble the origin latitude and longitude into a dictionary
    origin = {"lat": origin_la, "lng": origin_ln}
    # Assemble the destination latitude and longitude into a dictionary
    destination = {"lat": dies_la, "lng": dies_ln}

    # origin = {"lat": 40.6655101, "lng": -73.89188969999998}
    # destination = {"lat": 40.6905615, "lng": -73.9976592}

    # Call the function to calculate distance and duration between origin and destination
    distance, duration = get_distance_and_time(origin, destination, api_key)

    # Check if distance and duration were successfully retrieved
    if distance and duration:
        print(f"Distance: {distance}, Duration: {duration}")
    else:
        print("Could not retrieve the distance and time.")

    # Return the distance and duration as a tuple
    return (distance , duration)
```

```
[5]: def find_nearest_fire_stations(api_key, latitude, longitude, radius=5000):

    # Initialize the Google Maps client
    gmaps = googlemaps.Client(key=api_key)

    # Define the location using provided latitude and longitude
    location = (latitude, longitude)

    # Perform a nearby search for fire stations within the specified radius
    places_result = gmaps.places_nearby(location=location, radius=radius, keyword='fire station')

    # Check if results are found
    if places_result.get('results'):
        # Loop through each place found in the results
        for place in places_result['results']:

            #print(place)
            print(f"Name: {place.get('name')}")
            print(f"Address: {place.get('vicinity')}")
            print(f"Location: {place.get('geometry', {}).get('location')}")
            print("------------\n\n\n\n")

            # Extract the latitude and longitude from the location data
            location = place.get('geometry', {}).get('location')
            lat = float(location['lat'])
            lng = float(location['lng'])

            # Return the latitude and longitude of the first fire station found
            return lat , lng

    else:
        print("No fire stations found within the specified radius.")
```

Completion of the Component

Nearest fire station calculator

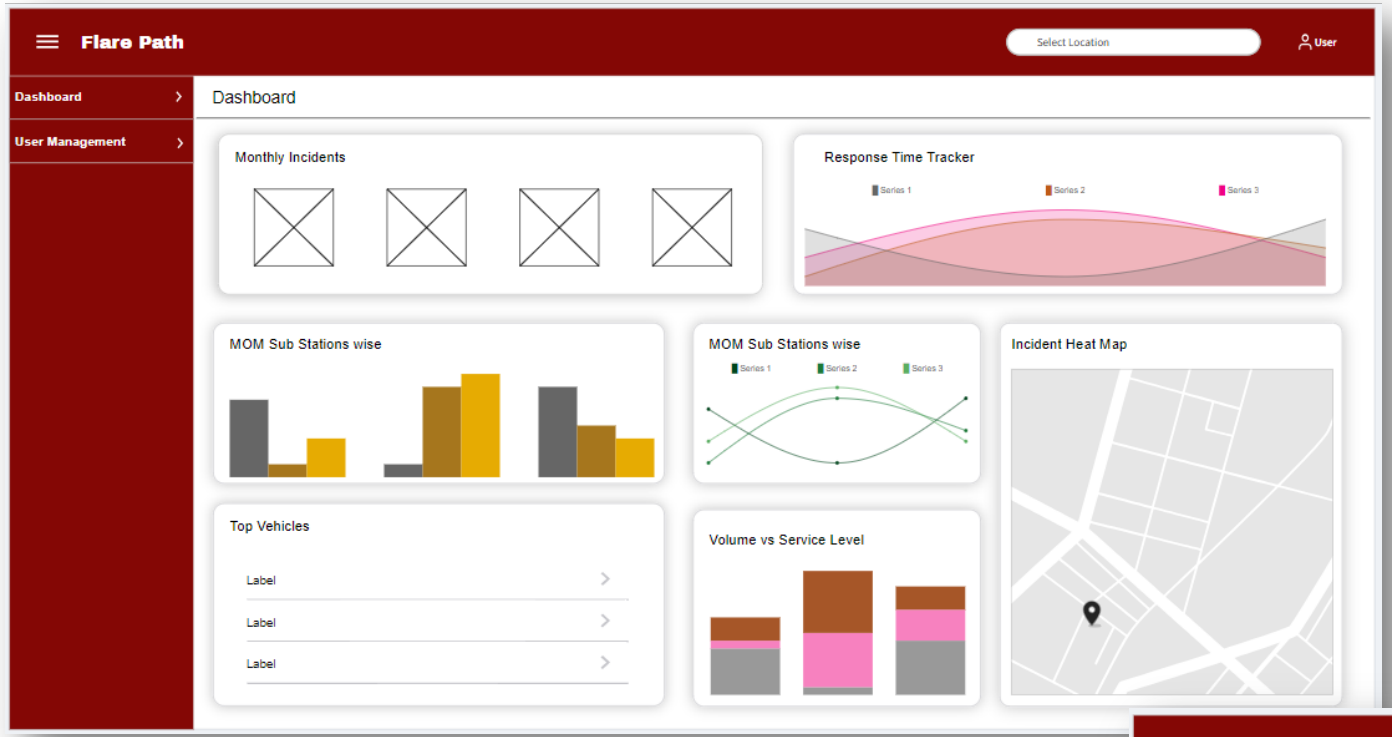
```
[6]: def nearest_fire_station_acording_to_given_location(lat , long):

        API_KEY = 'AIzaSyA_ZSQQ7qESG6TPvKviBf0XUIlIcGd84I4'

        # lat , long = 6.784841,79.932660
        # Set the latitude and longitude for the location
        LATITUDE = lat
        LONGITUDE = long

        # Call the function to find the nearest fire station to the given latitude and longitude
        lat,lng = find_nearest_fire_stations(API_KEY, LATITUDE, LONGITUDE)

        return lat , lng
```

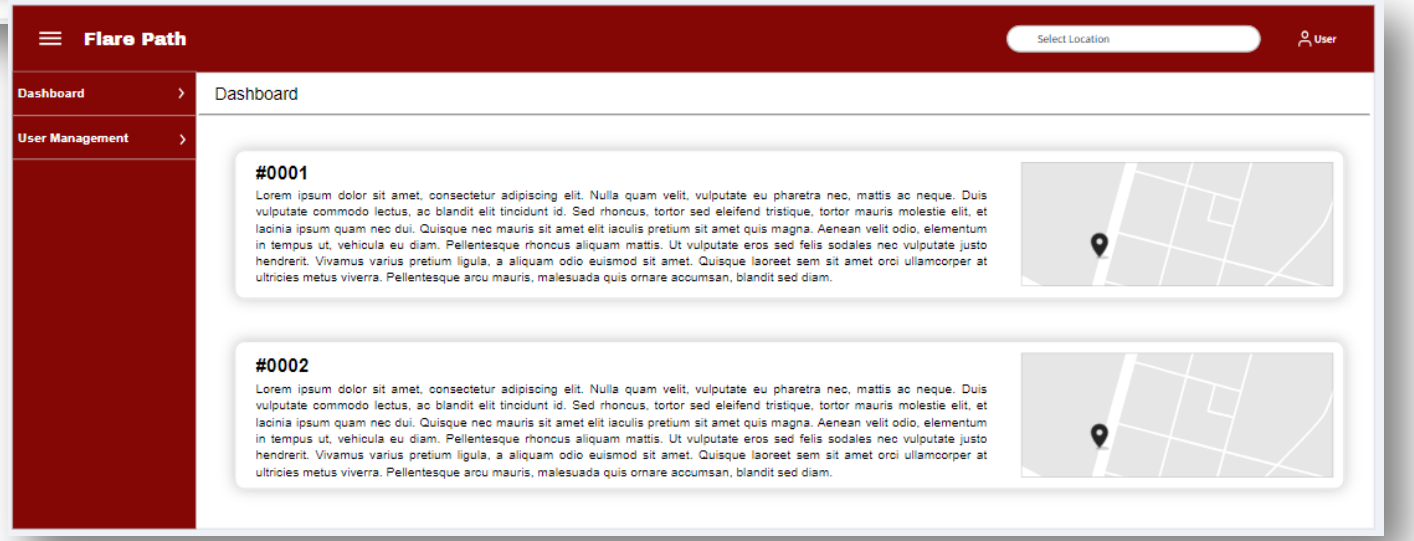
```
[7]: def create_google_maps_link(lat, long):

        # Base URL for Google Maps
        base_url = "https://www.google.com/maps?q="
        # Construct the full URL by appending the latitude and longitude to the base URL
        return f"{base_url}{lat},{long}"

        # lat , long = 6.782137,79.966980

        latitude = lat
        longitude = long

        link = create_google_maps_link(latitude, longitude)

        print(link)

        return link
```

```
[8]: def get_relatime_data(lat , long):

        # Find the nearest fire station to the given latitude and longitude
        fire_station_detels = nearest_fire_station_acording_to_given_location(lat , long)
        print("nearest fire station GPS coordinate - ", fire_station_detels)

        # Prepare the origin and destination coordinates for distance and time calculation
        origin_la , origin_ln , dies_la , dies_ln =  lat , long ,fire_station_detels[0] ,fire_station_detels[1]

        # Calculate the shortest distance and time from the vehicle to the fire station
        distance , duration = point_to_point_shorter_time_and_distance_calculator(origin_la , origin_ln , dies_la , dies_ln)
        print("shortest distance and time to vehicle - ", distance , duration)

        # Generate a Google Maps link for the vehicle's location
        gmap_link = create_google_maps_link(lat, long)
        print('vehicle google map related location link - ',gmap_link)
```

```
[9]: get_relatime_data(6.987395764204013,79.89860961287808)

Name: Grandpass Fire Station
Address: WVXC+7FQ, Colombo
Location: {'lat': 6.9482076, 'lng': 79.8712386}
------------



nearest fire station GPS coordinate -  (6.9482076, 79.8712386)
Distance: 6.0 km, Duration: 14 mins
shortest distance and time to vehicle -  6.0 km 14 mins
vehicle google map related location link -  https://www.google.com/maps?q=6.987395764204013,79.89860961287808
```

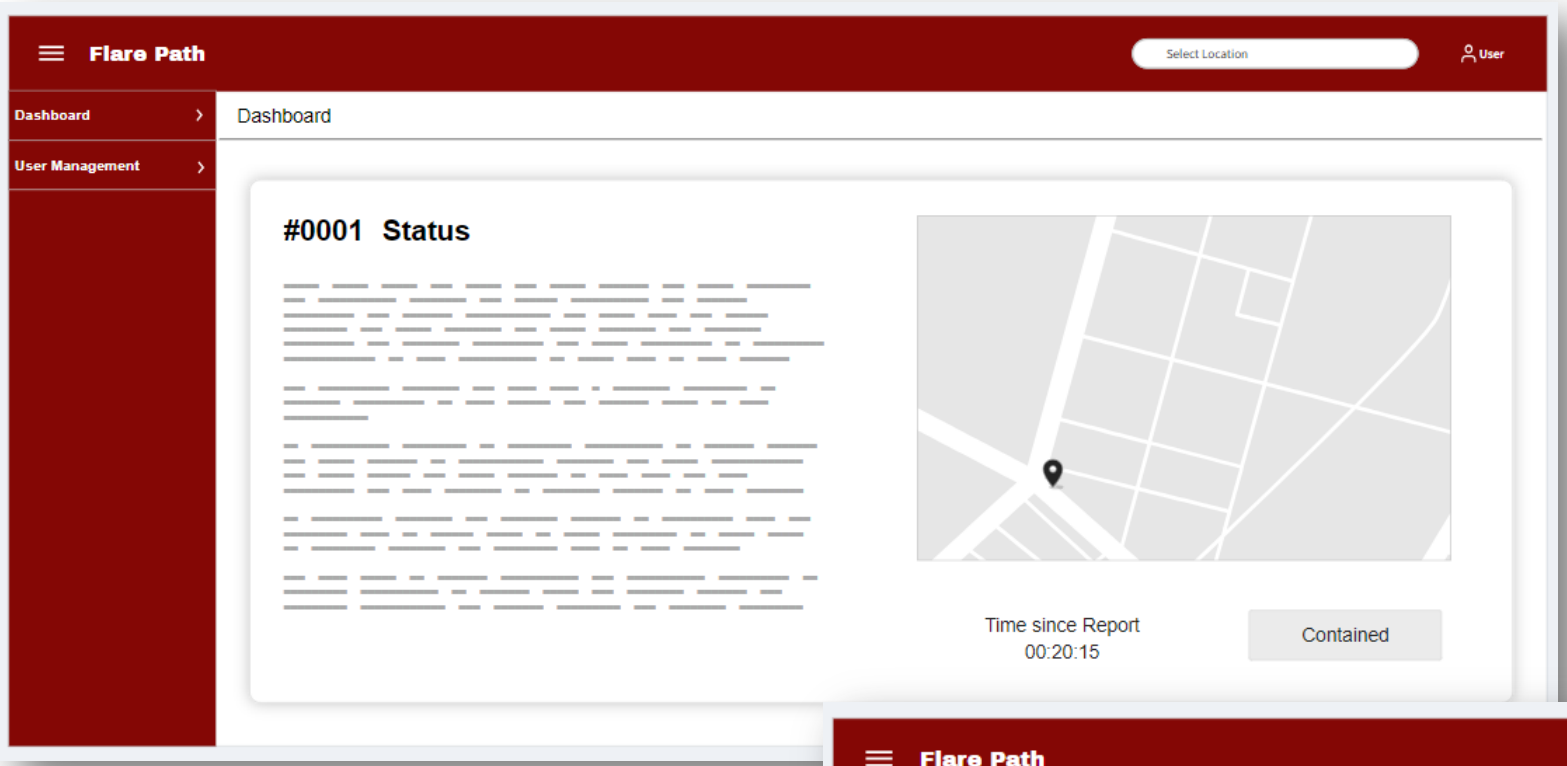
Wireframe

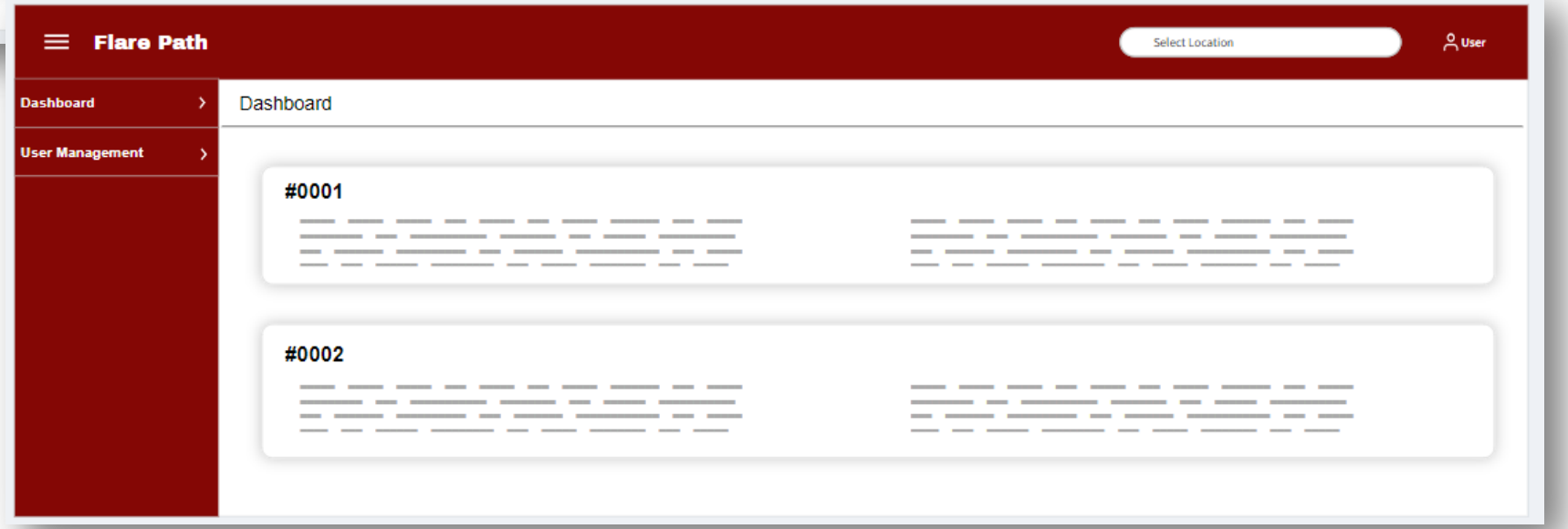

Flare Path
Select Location
User
Dashboard
User Management
Dashboard
Monthly Incidents
Response Time Tracker
Series 1
Series 2
Series 3
MOM Sub Stations wise
MOM Sub Stations wise
Series 1
Series 2
Series 3
Incident Heat Map
Top Vehicles
Label
Label
Label
Volume vs Service Level


Flare Path
Select Location
User
Dashboard
User Management
Dashboard
#0001
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam. Pellentesque rhoncus aliquam mattis. Ut vulputate eros sed felis sodales nec vulputate justo hendrerit. Vivamus varius pretium ligula, a aliquam odio euismod sit amet. Quisque laoreet sem sit amet orci ullamcorper at ultricies metus viverra. Pellentesque arcu mauris, malesuada quis ornare accumsan, blandit sed diam.
#0002
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nulla quam velit, vulputate eu pharetra nec, mattis ac neque. Duis vulputate commodo lectus, ac blandit elit tincidunt id. Sed rhoncus, tortor sed eleifend tristique, tortor mauris molestie elit, et lacinia ipsum quam nec dui. Quisque nec mauris sit amet elit iaculis pretium sit amet quis magna. Aenean velit odio, elementum in tempus ut, vehicula eu diam. Pellentesque rhoncus aliquam mattis. Ut vulputate eros sed felis sodales nec vulputate justo hendrerit. Vivamus varius pretium ligula, a aliquam odio euismod sit amet. Quisque laoreet sem sit amet orci ullamcorper at ultricies metus viverra. Pellentesque arcu mauris, malesuada quis ornare accumsan, blandit sed diam.

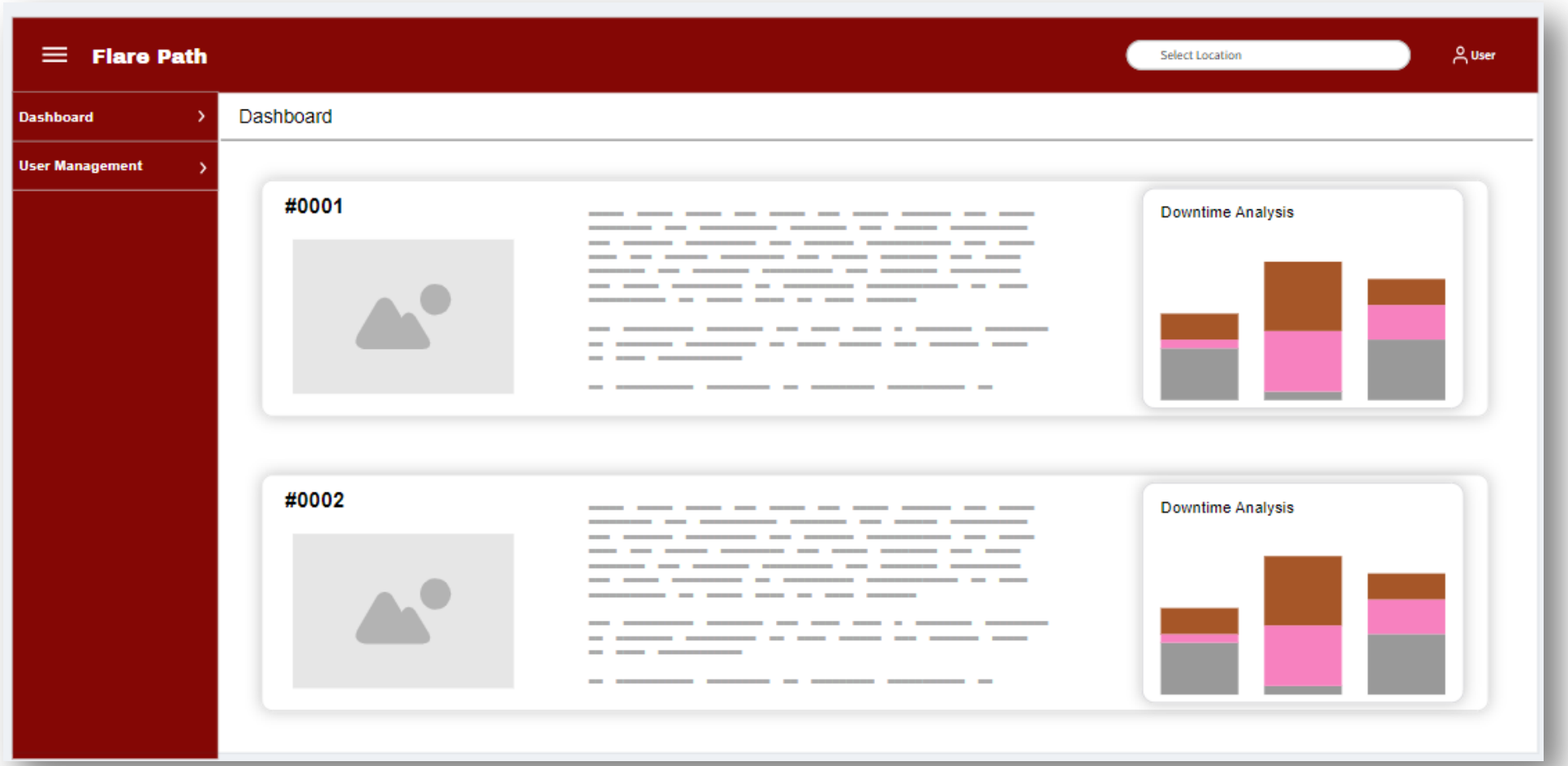
Best Practices of the development

Wireframe

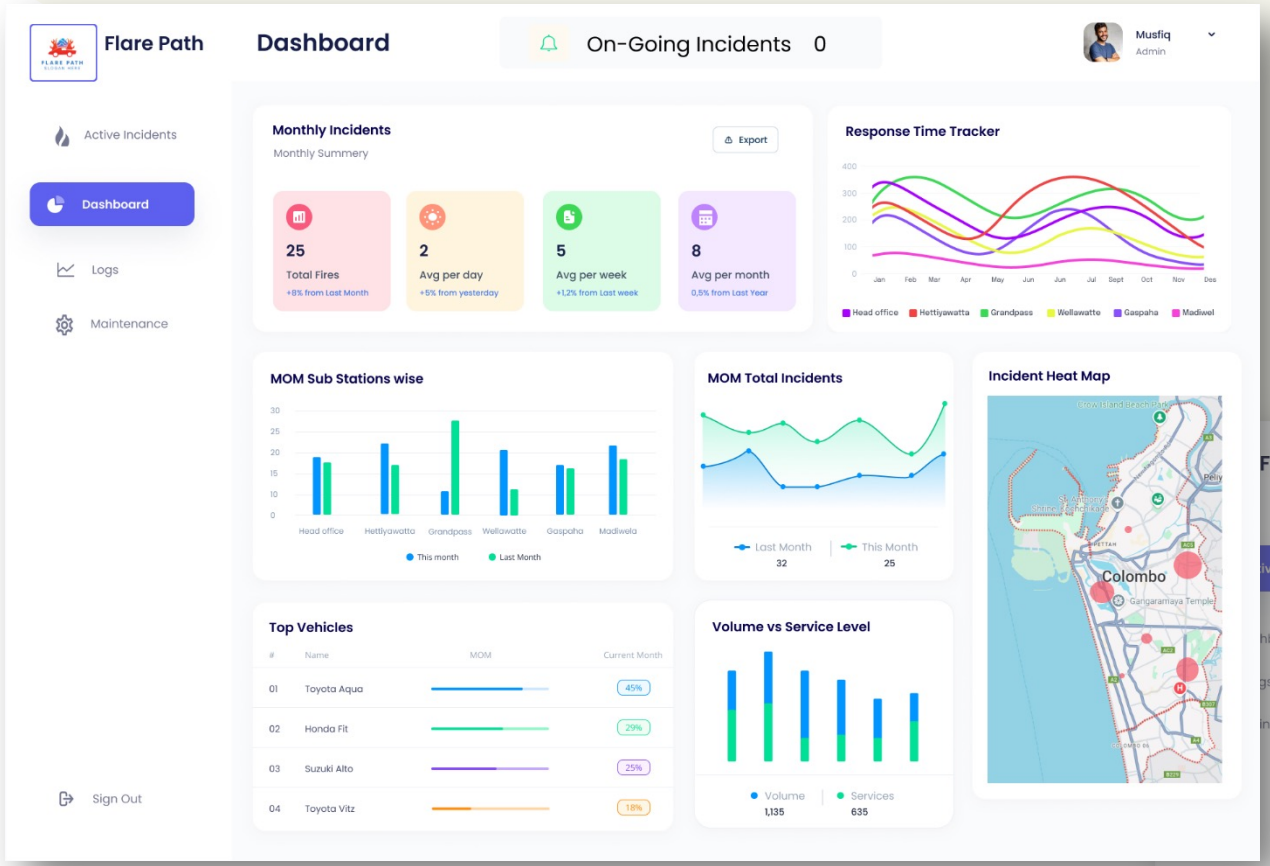


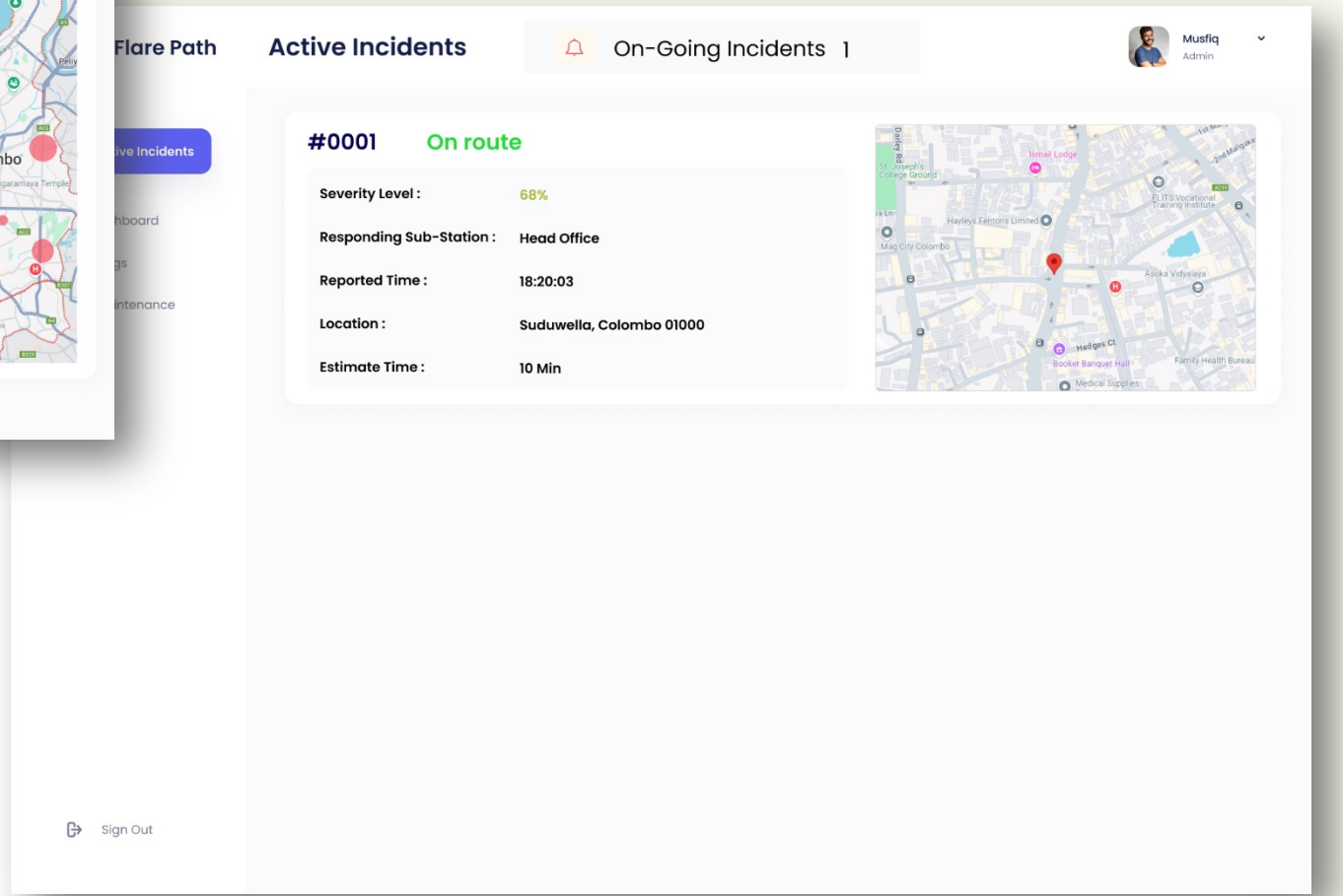


Wireframe

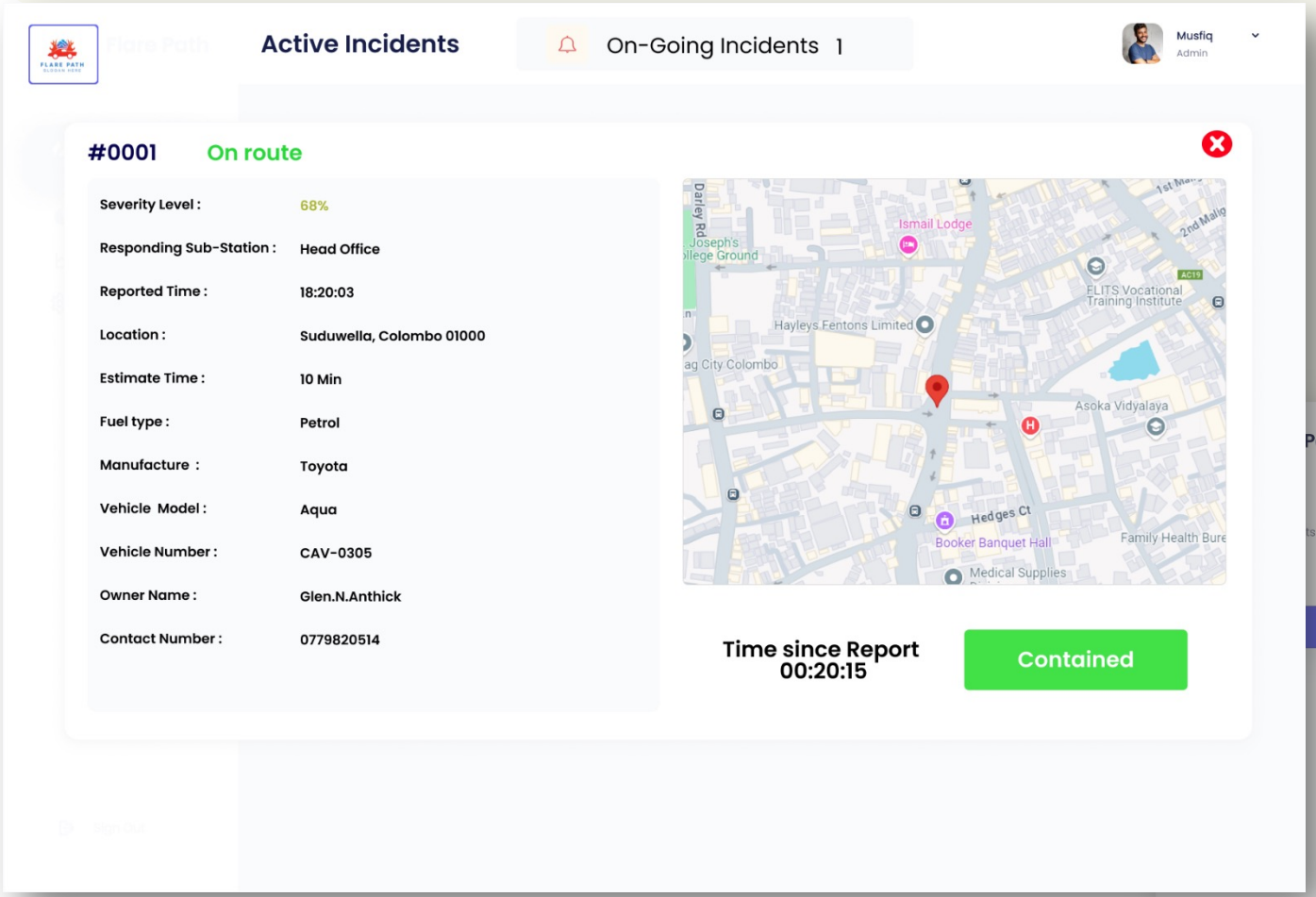

Flare Path
Select Location
User
Dashboard
User Management
Dashboard
#0001
Downtime Analysis
#0002
Downtime Analysis

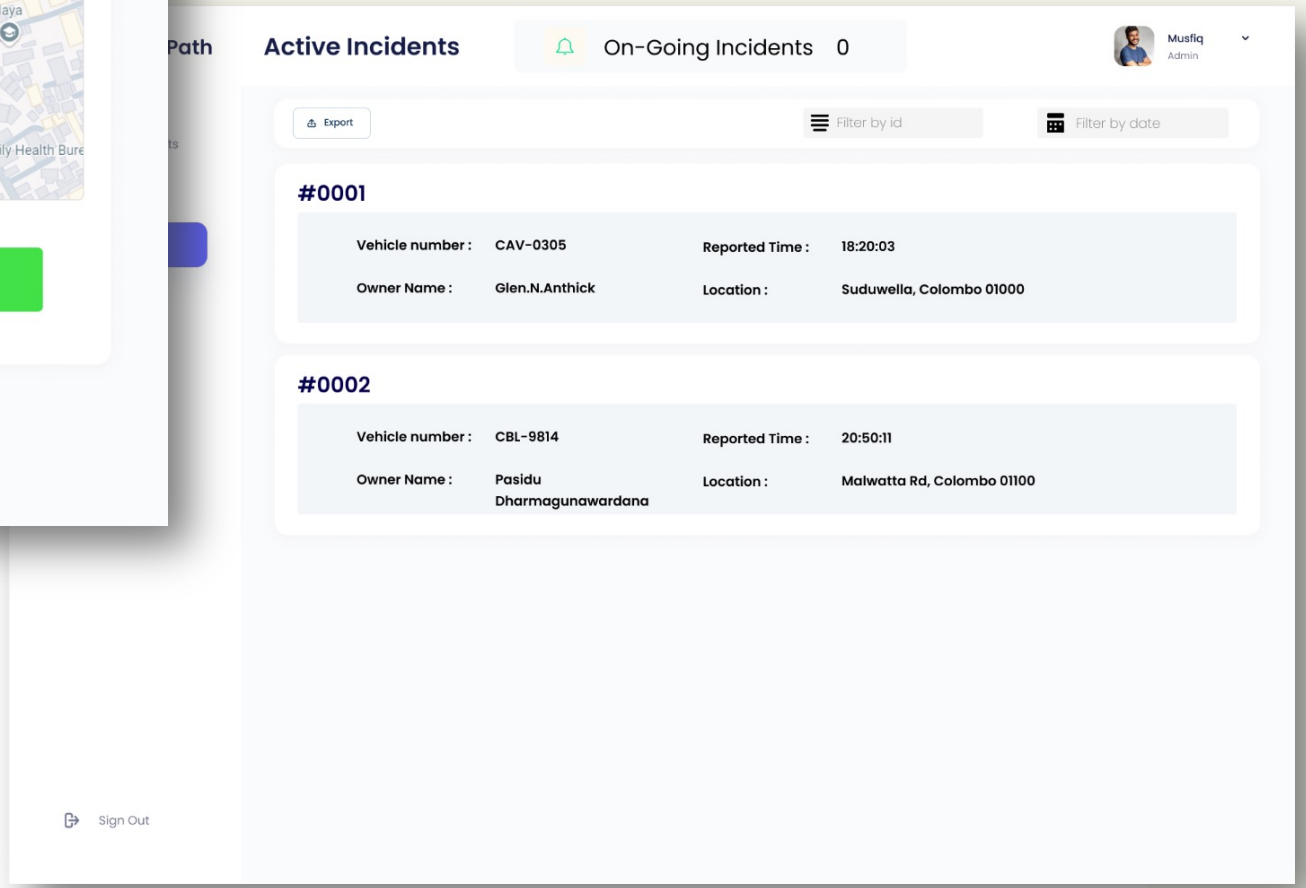
Frontend


Flare Path
Dashboard
On-Going Incidents 0
Musfiq
Admin
Active Incidents
Dashboard
Logs
Maintenance
Sign Out
Monthly Incidents
Monthly Summery
Export
25
Total Fires
+8% from Last Month
2
Avg per day
+5% from yesterday
5
Avg per week
+1.2% from Last week
8
Avg per month
0.5% from Last Year
Response Time Tracker
Head office
Hettiyawatta
Grandpass
Wellawatte
Gaspaha
Madiwel
MOM Sub Stations wise
Head office
Hettiyawatta
Grandpass
Wellawatte
Gaspaha
Madiwela
This month
Last Month
MOM Total Incidents
Last Month
32
This Month
25
Incident Heat Map
Colombo
Top Vehicles
Name MOM Current Month
01 Toyota Aqua 45%
02 Honda Fit 29%
03 Suzuki Alto 25%
04 Toyota Vitz 18%
Volume vs Service Level
Volume
1,135
Services
635

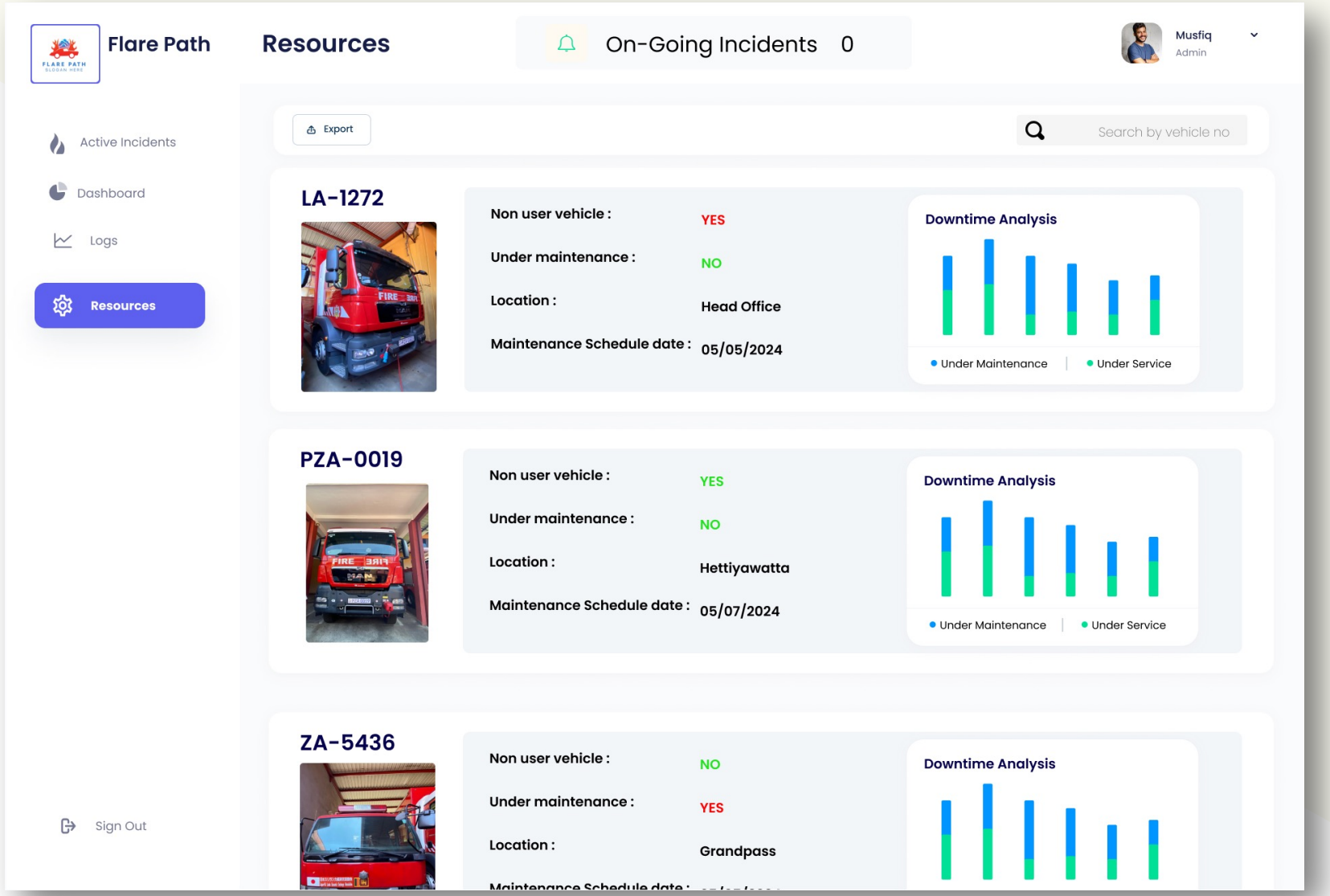

Flare Path
Active Incidents
On-Going Incidents 1
Musfiq
Admin
Sign Out
#0001
On route
Severity Level : 68%
Responding Sub-Station : Head Office
Reported Time : 18:20:03
Location : Suduwella, Colombo 01000
Estimate Time : 10 Min

Frontend


Active Incidents
On-Going Incidents 1
Musfiq
Admin
#0001
On route
Severity Level : 68%
Responding Sub-Station : Head Office
Reported Time : 18:20:03
Location : Suduwella, Colombo 01000
Estimate Time : 10 Min
Fuel type : Petrol
Manufacture : Toyota
Vehicle Model : Aqua
Vehicle Number : CAV-0305
Owner Name : Glen.N.Anthick
Contact Number : 0779820514
Ismail Lodge
Hayleys Fentons Limited
FLITS Vocational Training Institute
Asoka Vidyalaya
Hedges Ct
Booker Banquet Hall
Medical Supplies
Time since Report
00:20:15
Contained


Active Incidents
On-Going Incidents 0
Musfiq
Admin
Export
Filter by id
Filter by date
#0001
Vehicle number : CAV-0305
Reported Time : 18:20:03
Owner Name : Glen.N.Anthick
Location : Suduwella, Colombo 01000
#0002
Vehicle number : CBL-9814
Reported Time : 20:50:11
Owner Name : Pasidu Dharmagunawardana
Location : Malwatta Rd, Colombo 01100
Sign Out

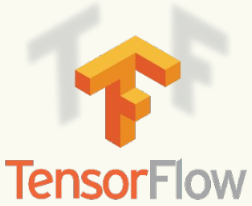
Frontend


Flare Path
Resources
On-Going Incidents 0
Musfiq
Admin
Active Incidents
Dashboard
Logs
Resources
Sign Out
Export
Search by vehicle no
LA-1272
Non user vehicle : YES
Under maintenance : NO
Location : Head Office
Maintenance Schedule date : 05/05/2024
Downtime Analysis
Under Maintenance
Under Service
PZA-0019
Non user vehicle : YES
Under maintenance : NO
Location : Hettiyawatta
Maintenance Schedule date : 05/07/2024
Downtime Analysis
Under Maintenance
Under Service
ZA-5436
Non user vehicle : NO
Under maintenance : YES
Location : Grandpass
Downtime Analysis

Tools and Technologies

Software Technologies

- Python
- TensorFlow
- AWS / Azure
- Jupyter Lab



Requirement Analysis

Functional Requirements

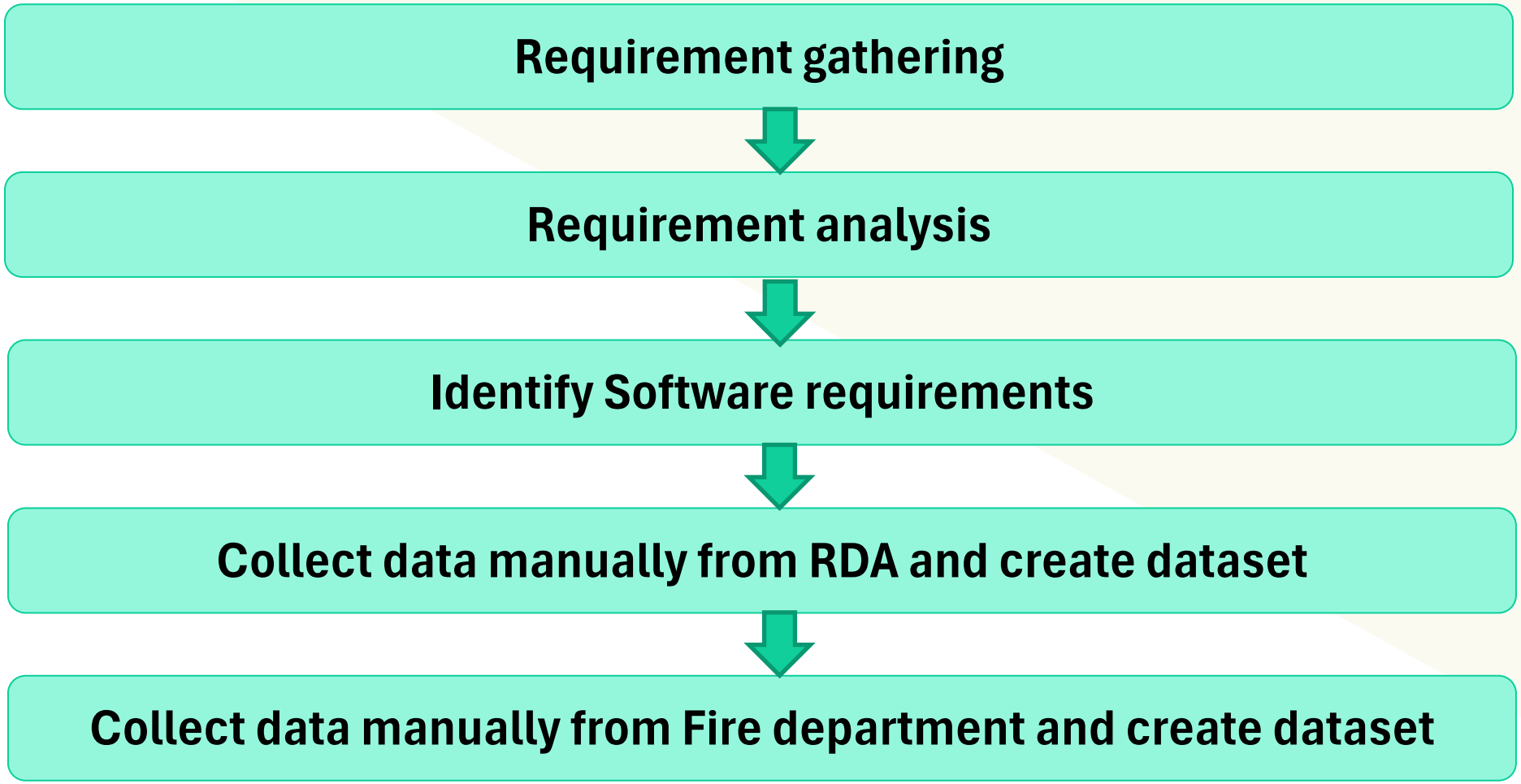
- Detect the location of a vehicle on fire and transmit this data to the cloud.
- Identify and notify the nearest fire department about the incident, including sending detailed information about the location and severity of the fire.
- Display the available emergency resources, including fire trucks and their operational status, using AI algorithms.
- 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

Overall Progress

Completed | Ongoing | To be started



Overall Progress

Design System flow chart

Design Dashboard wireframe

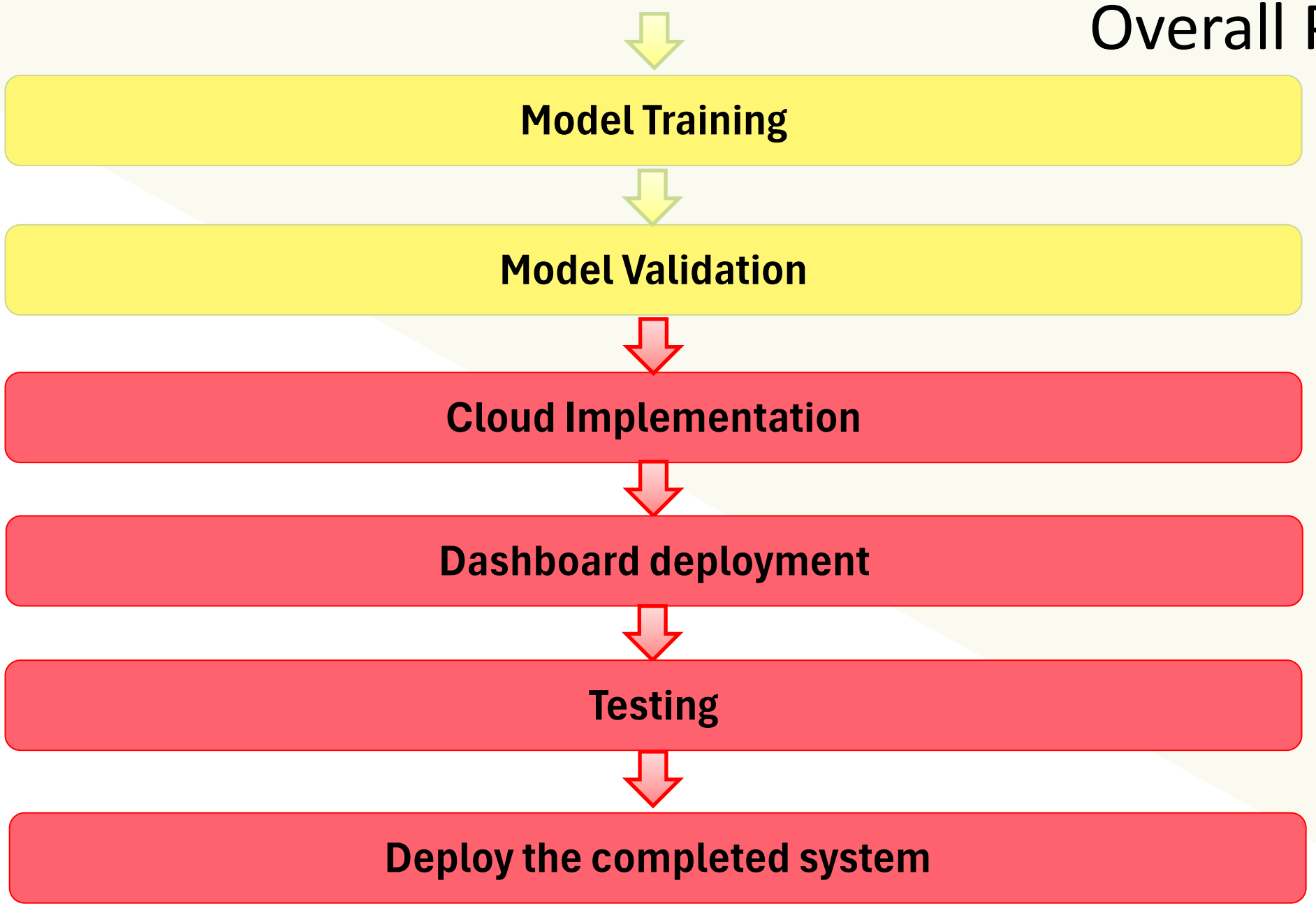
Design Dashboard frontend design

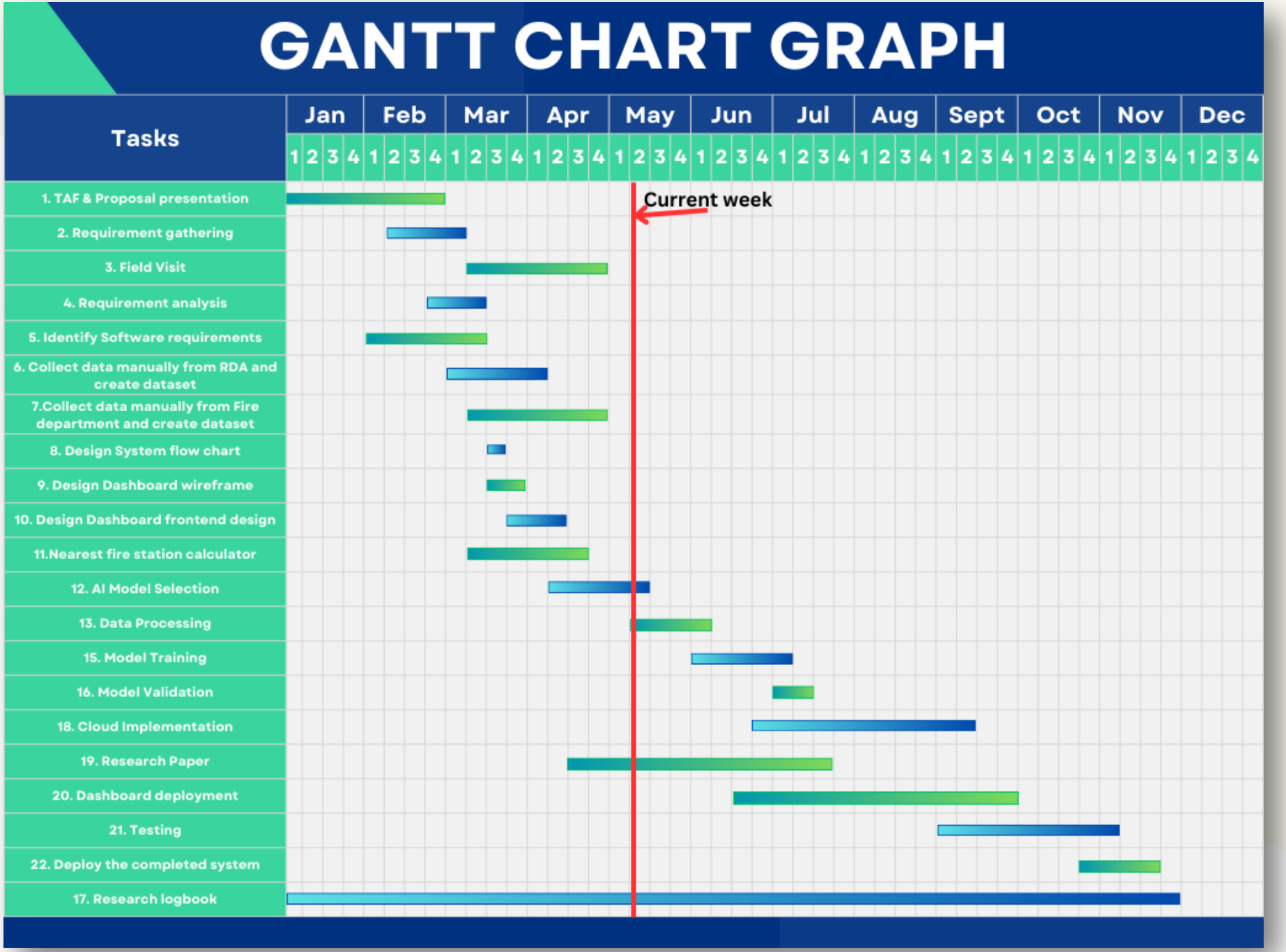
Nearest fire station calculator

AI Model Selection

Data Processing

Overall Progress




GANTT CHART GRAPH
Tasks
Jan
Feb
Mar
Apr
May
Jun
Jul
Aug
Sept
Oct
Nov
Dec
1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4 1 2 3 4
Current week
1. TAF & Proposal presentation
2. Requirement gathering
3. Field Visit
4. Requirement analysis
5. Identify Software requirements
6. Collect data manually from RDA and create dataset
7.Collect data manually from Fire department and create dataset
8. Design System flow chart
9. Design Dashboard wireframe
10. Design Dashboard frontend design
11.Nearest fire station calculator
12. AI Model Selection
13. Data Processing
15. Model Training
16. Model Validation
18. Cloud Implementation
19. Research Paper
20. Dashboard deployment
21. Testing
22. Deploy the completed system
17. Research logbook

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.

Thank You !