

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

R24-058

Status Document-1



Peramunage A N – IT21080562

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

Information Technology

Department of Information Technology

Sri Lanka Institute of Information Technology

Sri Lanka

March 2023

Group Details

Supervisor – Mr.Nelum Chathuranga Amarasena

Co-supervisor – Mr. Deemantha Nayanajith Siriwardana

External Supervisor – Mr. Onray Sahinda

Student Name	Student ID	Contact No	Email Address
Dharmagunawardana W.M.P.I	IT21132346	0772785361	it21132346@my.sliit.lk
Anthick G.N	IT21096266	0779820516	it21096266@my.sliit.lk
Abeywardhana D.N	IT21133718	0714057155	it21133718@my.sliit.lk
Peramunage A.N	IT21080562	0713999266	it21080562@my.sliit.lk

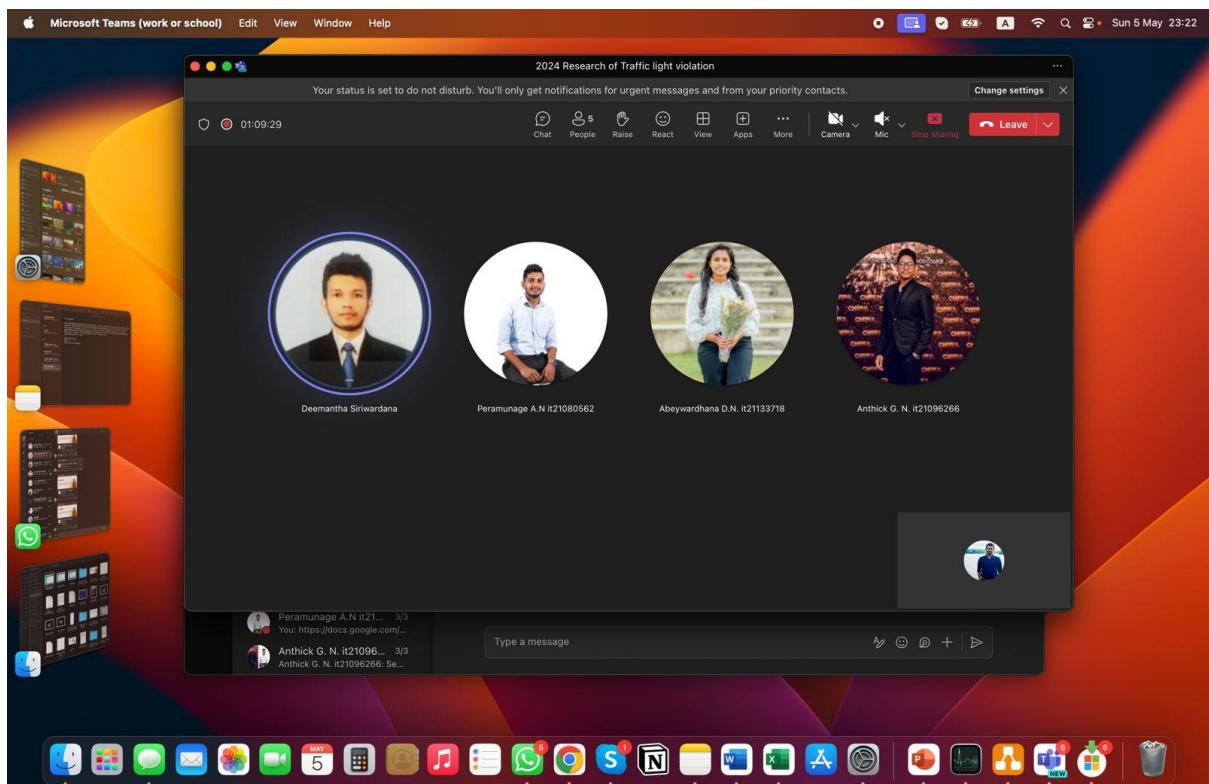
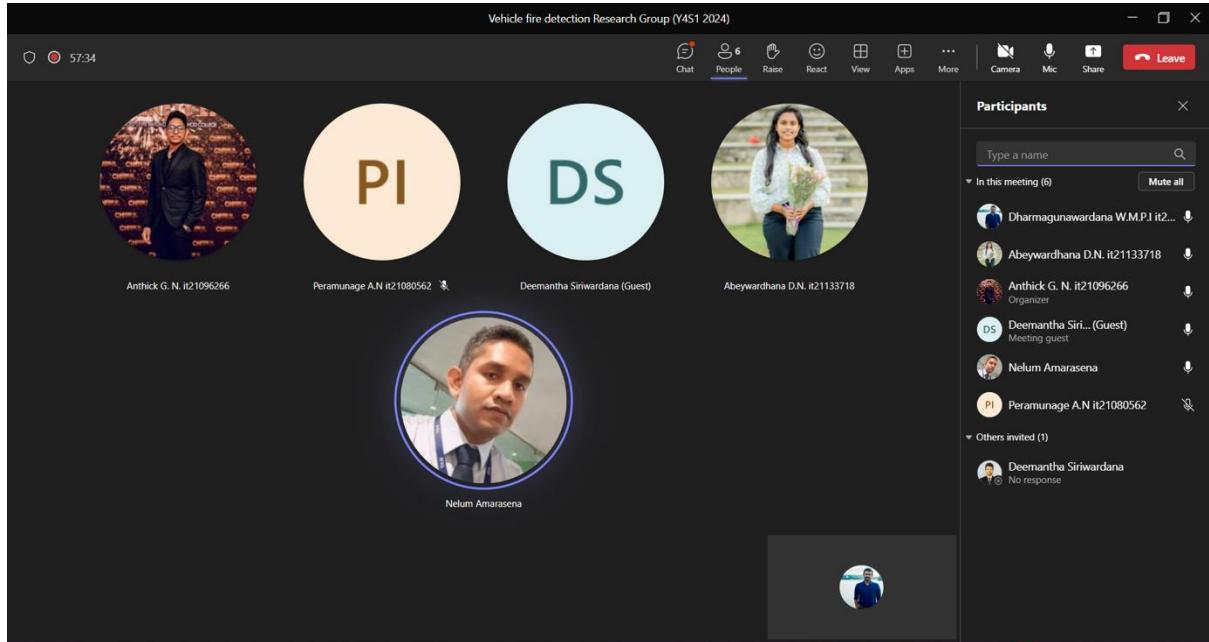
Table of Contents

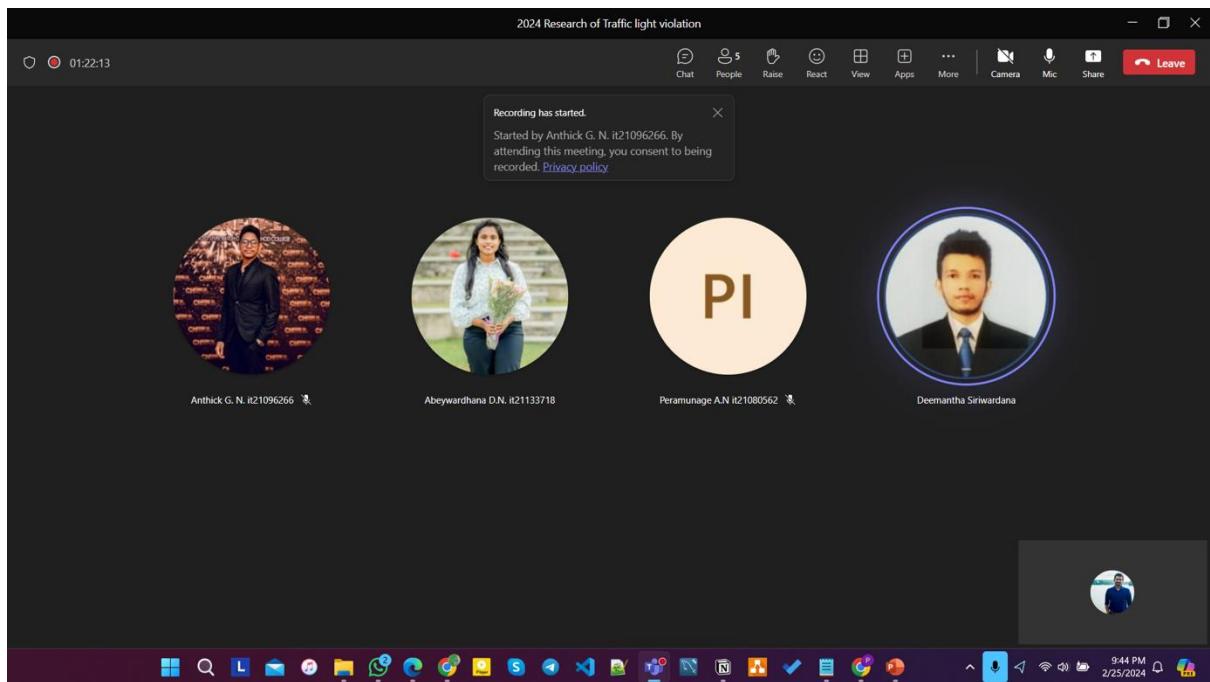
Group Details	2
Meetings & Calls	4
Meetings with supervisor and co-supervisor	4
Meetings with Domain Experts	8
Snapshots from Field Visit	9
Clickup Tasks Allocation.....	11
ClickUp Dashboard	11
In Progress Tasks.....	12
Completed Tasks upto PP1	12
Project Implementation.....	13
Data Collection	13
Machine Learning Model Training.....	Error! Bookmark not defined.
CNN Model	Error! Bookmark not defined.
RNN Model & SVM Model	18
Wireframes.....	24
Mobile Application UI.....	27
Gantt Chart	30
Work Breakdown Structure	30

Meetings & Calls

Meetings with supervisor and co-supervisor

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





2024 Research of Traffic light v... Chat Files Recap Speaker Coach Q&A + Join 0:0 5 : ...

Anthick G. N. it21096266 1/5 11:39 PM

Hi sir, can we arrange a meeting within next two days if you can. We have found some research papers and got some of the components. so we can discuss if there any issue in that, since we have to submit TAF on 11th this month

January 7

You're invited to Co-Supervisor meeting
Sunday, 7 January 2024
9:30 pm - 10:30 pm (GMT+5:30)
https://teams.microsoft.com/l/meetup-join/19%3ameeting_mDc3YmEtMmltNDQxYS00MzEyLTkYjgtMlVmNGl3MzY4NjQy%40thread.v2/0?context=%7b%22Id%22%3a%2244e3cf94-19c9-4e32-96c3-14f5bf01391a%22%2c%22Oid%22%3a%22b3aeafc1cb656-4424-ad90-3b6463f1a17e%22%7d
Tap on the link or paste it in a browser to join.

Join conversation teams.microsoft.com

1/7 9:17 PM Meeting ended: 14s

1/7 9:17 PM Meeting started

1/7 10:02 PM Recording has started

1/7 10:13 PM Meeting ended: 56m 15s

1/7 10:13 PM Recording has stopped. Saving recording...

Co-Supervisor meeting Recording Abeywardhana D.N. it2... 11m 53s

This recording is set to expire. View or change the expiration date [here](#). [Learn more](#)

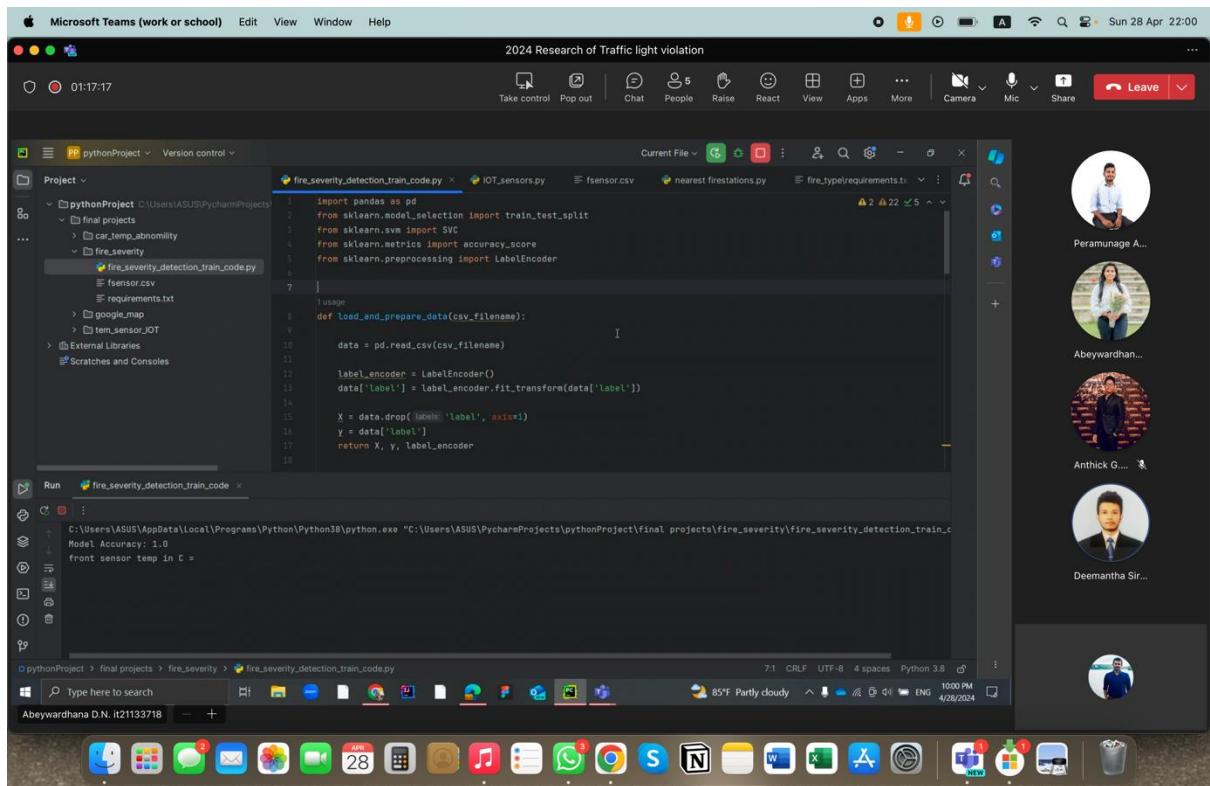
January 16

1/16 9:48 PM Meeting ended: 14s

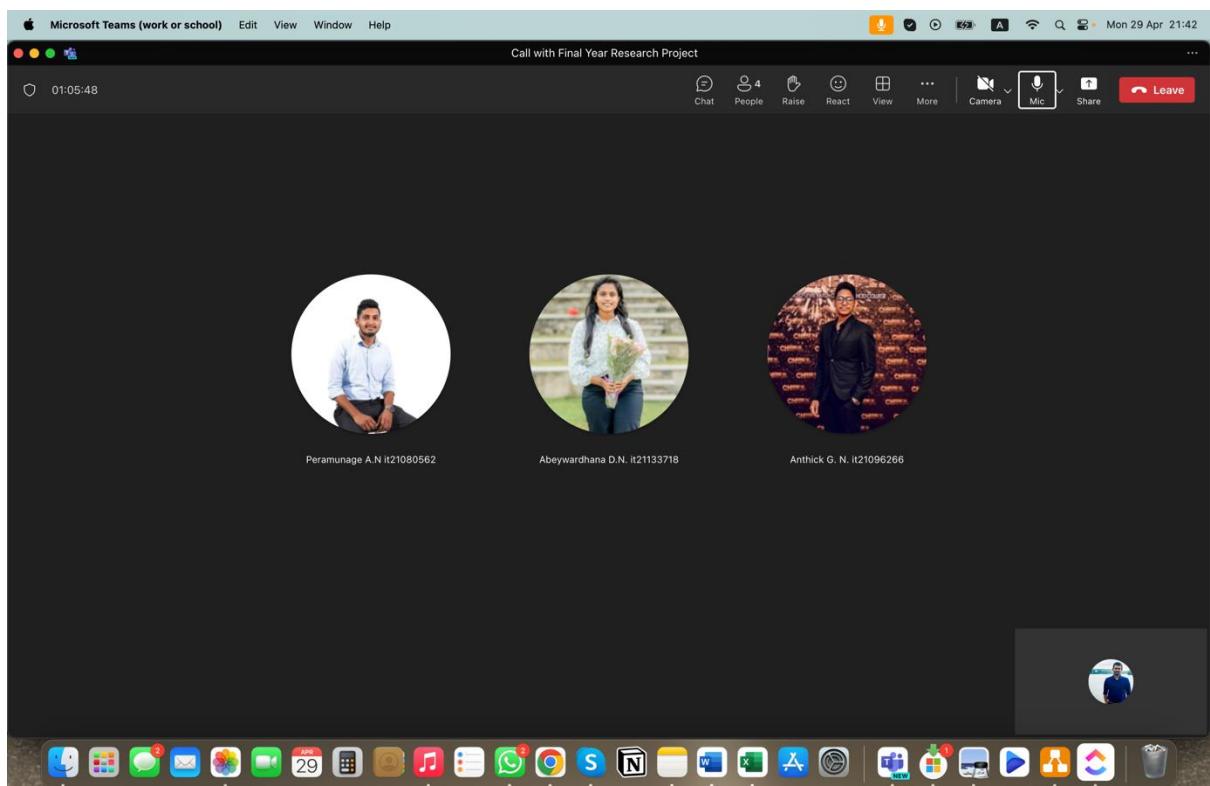
January 17

Type a message

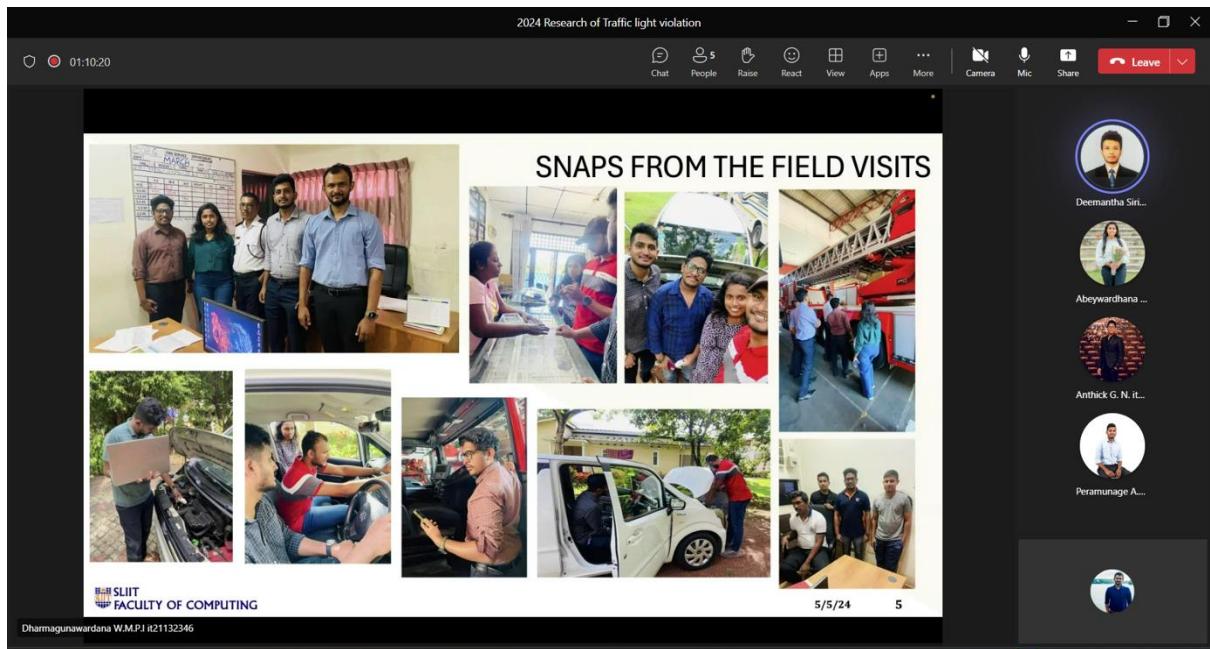
Code review with co-supervisor



Group meeting with group members



Presentation review with supervisor



Meetings with Domain Experts

Meeting With fire department officers and staff.

Mr.Nanayakkara the chief office of Fire department and we discussed about the domain knowledge and requirements.



	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
(Communication)

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

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

Snapshots from Field Visit





Clickup Tasks Allocation

The ClickUp Board view displays tasks for the project "Research - 058". The tasks are organized into five main status categories:

- To Do:** 0 tasks
- Planning:** 4 tasks
 - Implement the Fire sensor
 - Create a circuit Diagram
 - Add few more sensors and increase the accuracy
 - Fire severity model creation
- In Progress:** 11 tasks
 - create a mobile app
 - Decide features
 - Create Figma design
 - Fire severity data set creation
 - mobile app Design
 - Train the model
- Complete:** 13 tasks
 - Uploaded a Blinking program to the ESP32
 - Arduino IDE download and setup
 - Fire sensor Integration with ESP32
 - optimal path suggestion
 - nearest fire department identification
 - Fire department location data collection and data set creation
- Cancelled:** 0 tasks

ClickUp Dashboard

The ClickUp Dashboard provides a high-level overview of the project's status:

- Unassigned:** 10 tasks
- In Progress:** 15 tasks
- Completed:** 13 tasks
- Tasks Completed This Week:** No results

Total Tasks by Assignee:

Assignee	Percentage
it21132346 Dharm...	20%
ii21096266 Anithick G. N.	17.64%
ii21090562 Paramunage A.N	8.82%
it21133718 Abeywardh...	6.82%
Unassigned	44.11%

Open Tasks by Assignee:

Assignee	Count
it21132346 Dharm...	2
ii21096266 Anithick G. N.	1
ii21090562 Paramunage A.N	1
it21133718 Abeywardh...	1
Unassigned	10

Latest Activity:

- Fire severity model creation

In Progress Tasks

The screenshot shows a ClickUp dashboard titled "In Progress" with 15 tasks listed. The tasks are:

- Implement the Fire sensor
- create a circuit Diagram
- create a mobile app [t_0 2]
- Fire severity data set creation
- mobile app Design [t_0 2]
- Train the model
- Fire prediction data set creation [t_0 1]
- fire Prediction model
- fire prediction data collection
- fire department resource allocation prediction with the severity of the fire.
- Web Application Wireframes Design
- Add few more sensors and increase the accuracy
- fire prediction monitoring IOT device [t_0 1]
- Mobile Application Design
- Fire severity model creation

The tasks are displayed in a table with columns for ASSIGNEE, DUE DATE, and STATUS. Most tasks are labeled "IN PROGRESS".

Completed Tasks upto PP1

The screenshot shows a ClickUp dashboard titled "Completed" with 13 tasks listed. The tasks are:

- Uploaded a Blinking program to the ESP32
- Arduino IDE download and setup [t_0 1]
- Fire sensor Integration with ESP32
- optimal path suggestion
- nearest fire department identification
- Fire department location data collection and data set creation
- fire department resource data collection
- Draw Swimlane chart
- Create a swimlane diagram
- Web application Wireframes Design [t_0 3]
- Swimlane chart design
- Web Application Wireframes Design
- Fire department data collection

The tasks are displayed in a table with columns for ASSIGNEE, DUE DATE, and STATUS. All tasks are labeled "COMPLETE".

Project Implementation

Data Collection

Collectiong the temprature and RPM data using the themometer and a RPM guge.



Vehicle Safety System for Fire Detection and Prevention

Deep Learning CNN Model

```
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
```

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

# Prepare the input features and labels
X = df[['1000_rpm_tempreature', '2000_rpm_tempreature', '3000_rpm_tempreature', '4000_rpm_tempreature',
         '5000_rpm_tempreature', '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)
```

```
# 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)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=256, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=512, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1024, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2048, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4096, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=8192, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=16384, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=32768, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=65536, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=131072, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=262144, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=524288, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1048576, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2097152, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4194304, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=8388608, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=16777216, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=33554432, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=67108864, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=134217728, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=268435456, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=536870912, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1073741824, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2147483648, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4294967296, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=8589934592, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=17179869184, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=34359738368, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=68719476736, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=137438953472, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=274877906944, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=549755813888, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1099511627776, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2199023255552, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4398046511104, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=8796093022208, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=17592186044416, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=35184372088832, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=70368744177664, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=140737488355328, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=281474976710656, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=562949953421312, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1125899906842624, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2251799813685248, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4503599627370496, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=9007199254740992, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=18014398509481984, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=36028797018963968, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=72057594037927936, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=144115188075855872, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=288230376151711744, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=576460752303423488, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1152921504606846976, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2305843009213693952, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4611686018427387904, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=9223372036854775808, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=18446744073709551616, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=36893488147419103232, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=73786976294838206464, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=147573952589676412928, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=295147905179352825856, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=590295810358705651712, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1180591620717411303424, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2361183241434822606848, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4722366482869645213696, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=9444732965739290427392, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=18889465931478580854784, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=37778931862957161709568, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=75557863725914323419136, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=15111572745182864683832, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=30223145490365729367664, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=60446290980731458735328, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=120892581961462917470656, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=241785163922925834941312, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=483570327845851669882624, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=967140655691703339765248, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1934281311383406679530496, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=3868562622766813359060992, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=7737125245533626718121984, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=15474250491067253436243968, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=30948500982134506872487936, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=61897001964269013744975872, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=12379400392853802748995176, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=24758800785707605497990352, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=49517601571415210995980704, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=99035203142830421991961408, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=198070406285660843983922816, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=396140812571321687967845632, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=792281625142643375935691264, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1584563252285286751871382528, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=3169126504570573503742765056, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=6338253009141147007485530112, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=12676506018282294014971060224, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=25353012036564588029942120448, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=50706024073129176059884240896, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=101412048146258352119768811792, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=202824096292516704239537623584, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=405648192585033408479075247168, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=811296385170066816958150494336, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1622592770340133633916300988672, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=3245185540680267267832601977344, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=6490371081360534535665203954688, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=12980742162721069071330407909376, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=25961484325442138142660815818752, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=51922968650884276285321631637504, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=103845937301768552570643263275008, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=207691874603537105141286526550016, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=415383749207074210282573053100032, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=830767498414148420565146106200064, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1661534996828296841130292212400128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=3323069993656593682260584424800256, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=6646139987313187364521168849600512, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=13292279974626374729042337699201024, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=26584559949252749458084675398402048, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=53169119898505498916169350796804096, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=106338239797010997832338701593608192, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=212676479594021995664677403187216384, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=425352959188043991329354806374432768, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=850705918376087982658709612748865536, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1701411836752175965317419225497731072, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=3402823673504351930634838450995462144, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=6805647347008703861269676901989324288, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=13611294694017407722539353803978688576, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=27222589388034815445078707607957377152, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=54445178776069630890157415215914754304, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=10889035755213926178031483043182908608, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=21778071510427852356062966086365817216, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=43556143020855704712125932172731634432, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=87112286041711409424251864345463268864, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=174224572083422818848503728690926537728, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=348449144166845637697007457381853075456, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=696898288333691275394014914763706150912, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1393796576667382550788029829527412301824, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2787593153334765101576059659054824603648, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=5575186306669530203152119318109649207296, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=1115037261333906040630423863621929841492, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2230074522667812081260847727243859682984, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=4460149045335624162521695454487719365968, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=8920298090671248325043390908975438739376, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=17840596181342496650086781817950877478752, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=35681192362684993300173563635901555457504, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=71362384725369986600347127271803110155008, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=14272476945073997320069425454360622030016, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=28544953890147994640018850888721244060032, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=57089867780295989280037701777442488120064, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=11417973556059197856007540355488497624128, kernel_size=3, activation='relu', kernel_regularizer=l2(0.01)),
    BatchNormalization(),
    MaxPooling1D(pool_size=2),
    Dropout(0.35),
    Conv1D(filters=2283594711211839571201508071097
```

```

# 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'])

```

```

# 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')

```

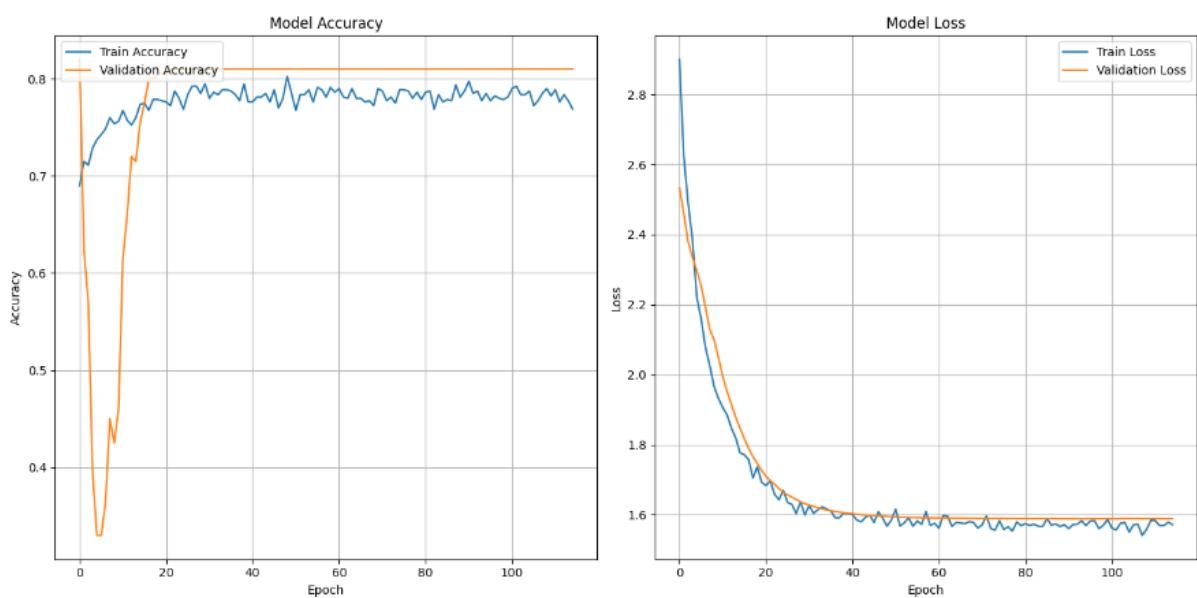
```

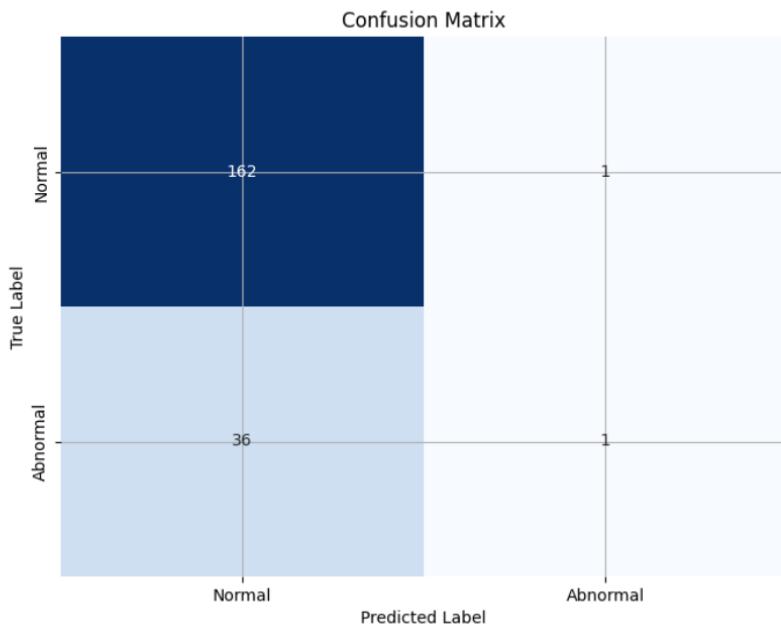
# Visualization of the training process
plt.figure(figsize=(14, 7))
plt.subplot( *args: 1, 2, 1)
plt.plot( *args: history.history['accuracy'], label='Train Accuracy')
plt.plot( *args: history.history['val_accuracy'], label='Validation Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
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}")

```





```
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
```

Deep Learning RNN Model(LSTM)

```
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_tempreature', '1000_rpm', '2000_rpm_tempreature', '3000_rpm_tempreature',
            '4000_rpm_tempreature', '5000_rpm_tempreature', '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)
```

```

# 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))

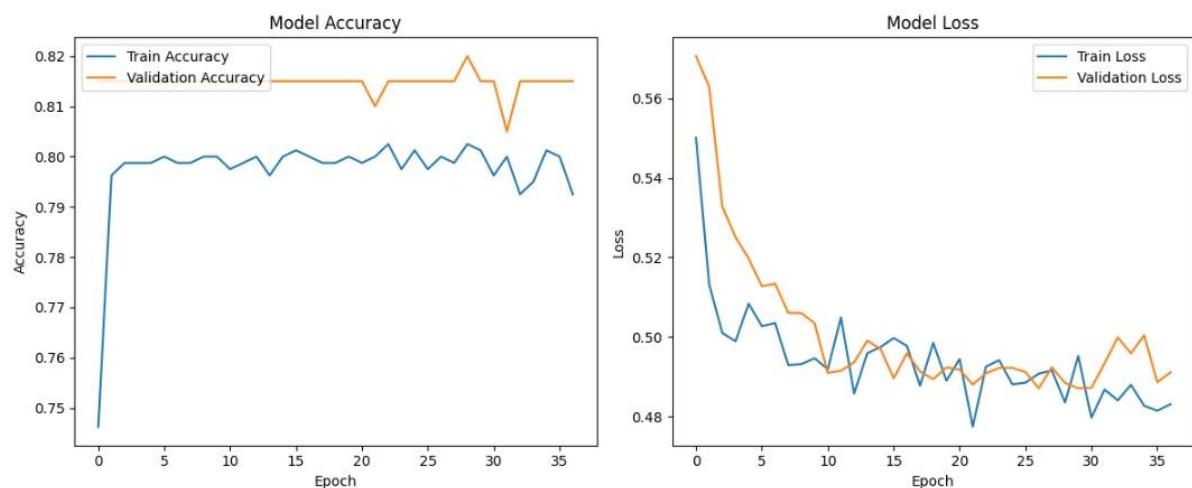
```

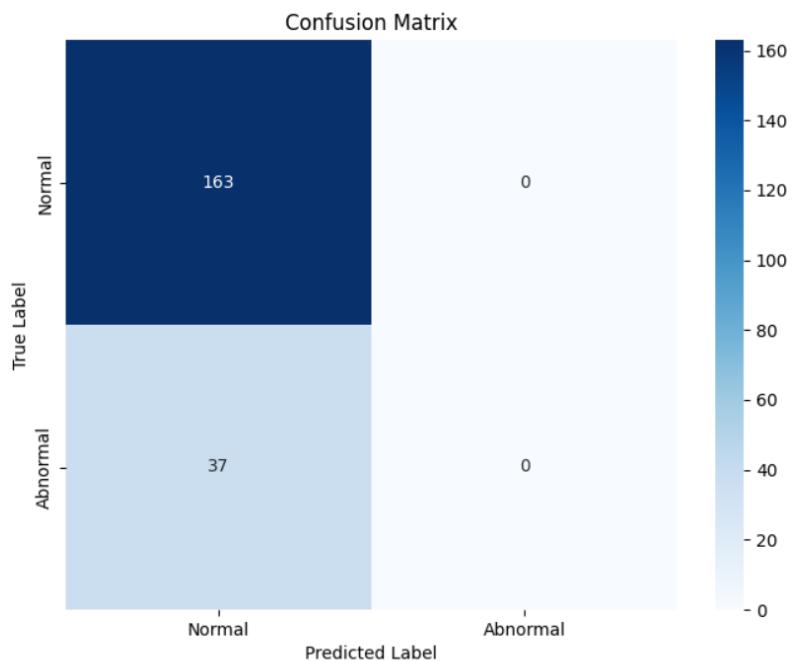
```

# 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()

```





```
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

Deep learning SVM Model

```
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_tempntrate', '1000_rpm', '2000_rpm_tempntrate', '3000_rpm_tempntrate',
            '4000_rpm_tempntrate', '5000_rpm_tempntrate', '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)

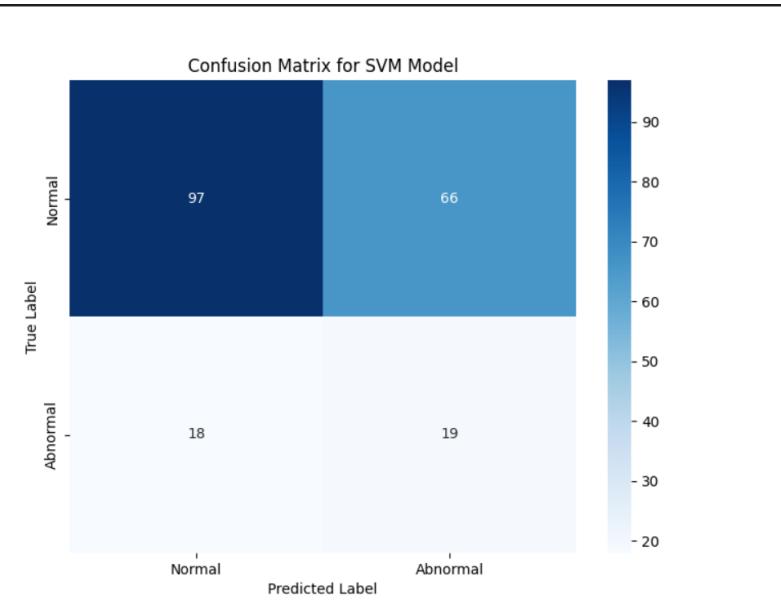
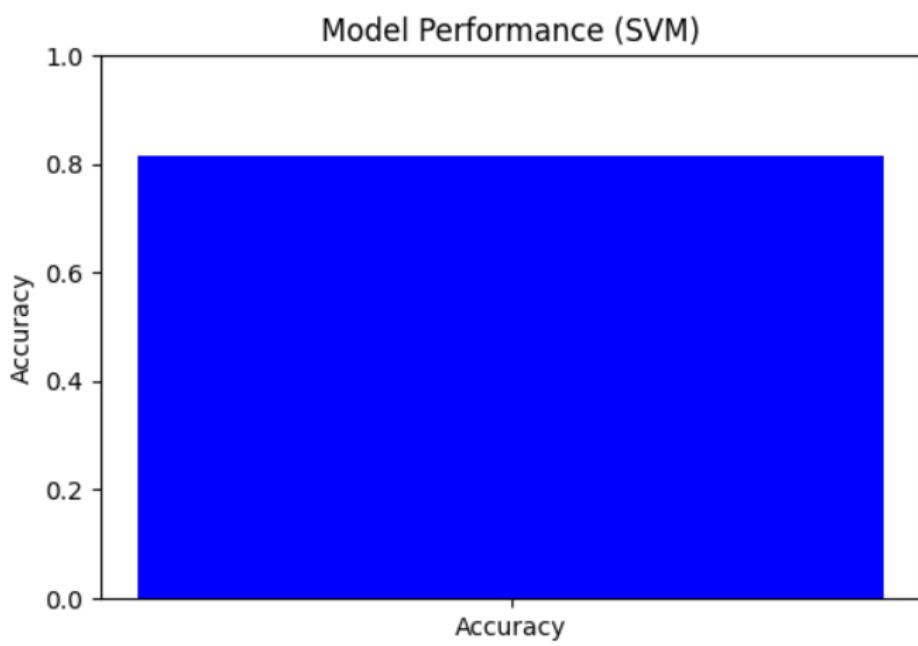
# Build SVM model
svm_model = SVC(class_weight='balanced') # Adjusting class weights for handling imbalanced data
svm_model.fit(X_train, y_train)

# 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))
```



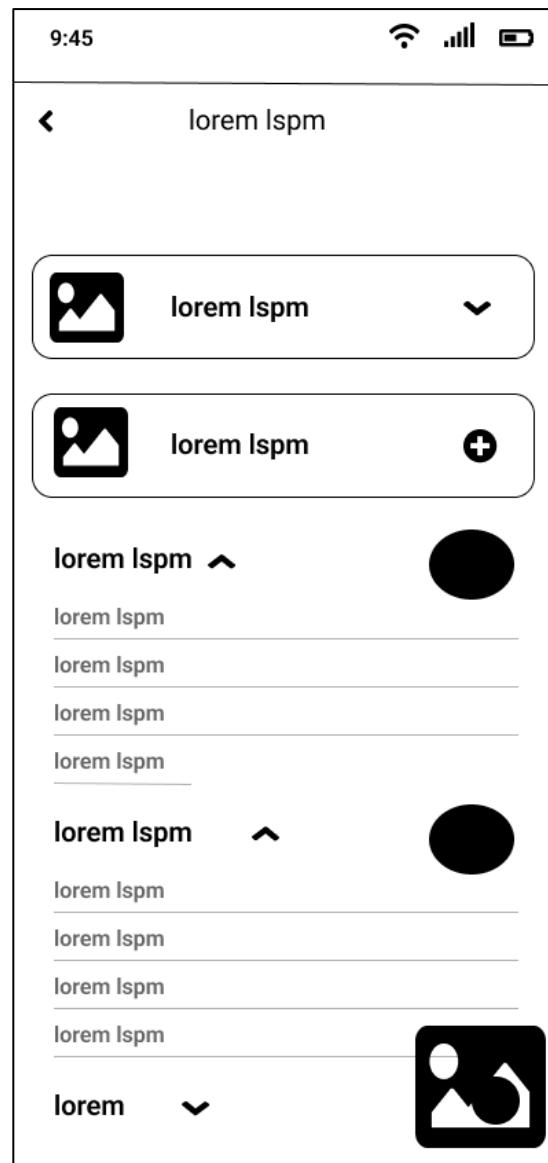
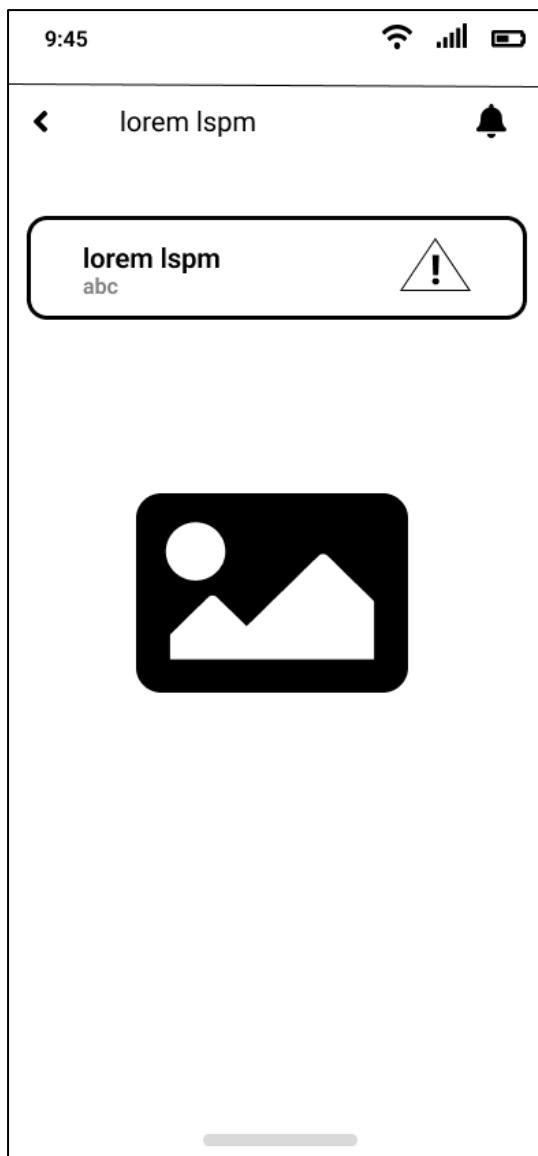
SVM Accuracy: 0.58

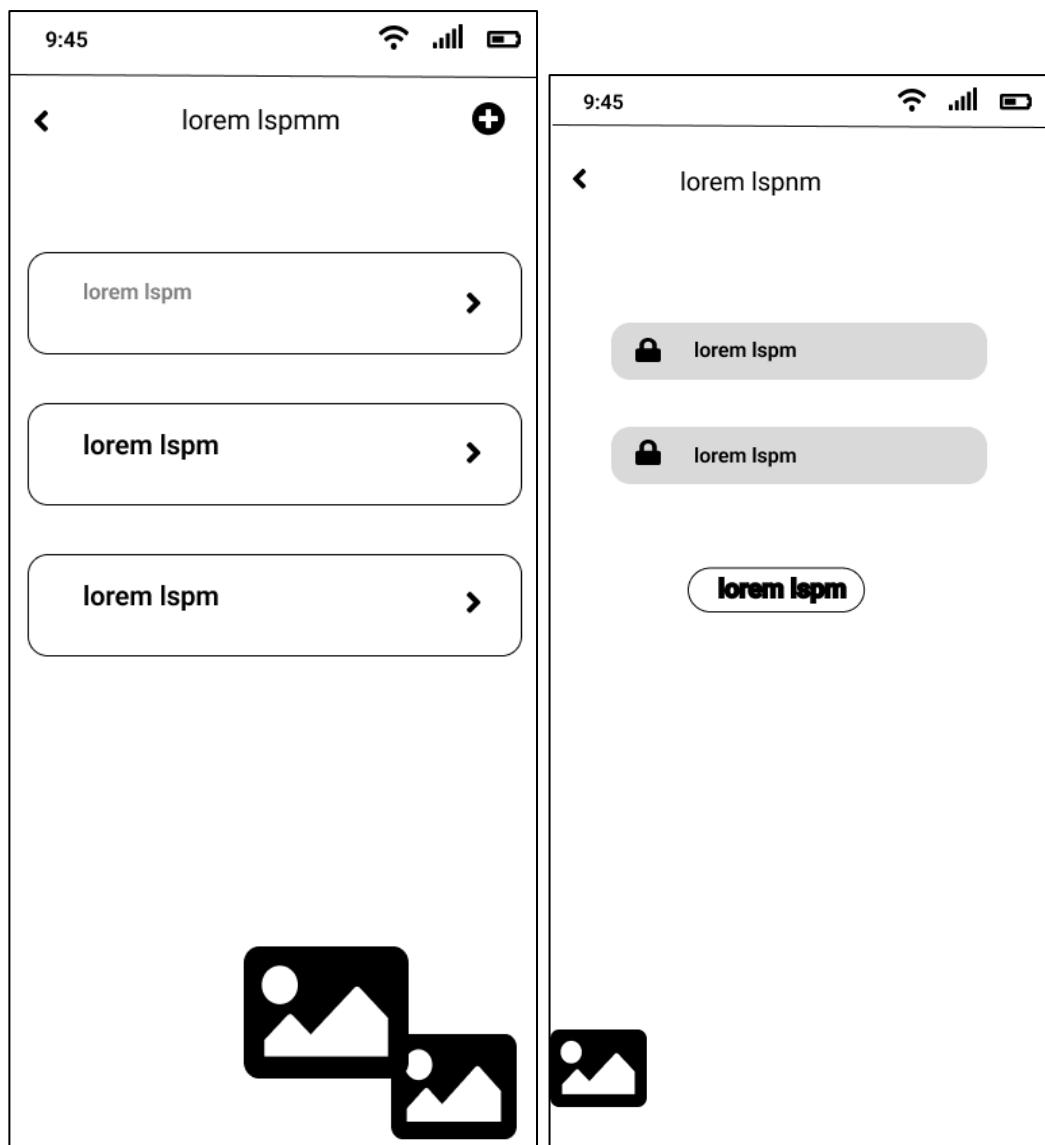
Classification Report:

	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

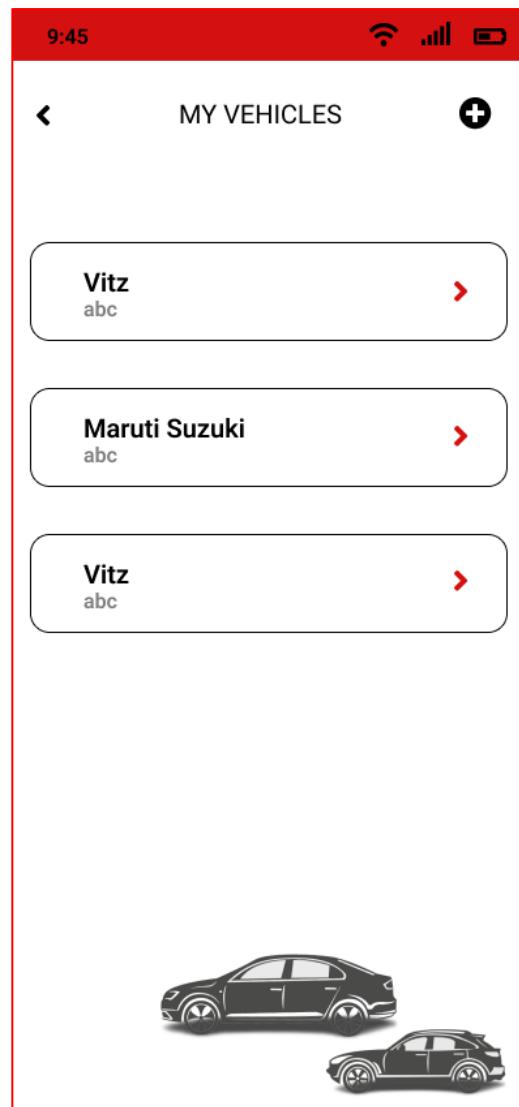
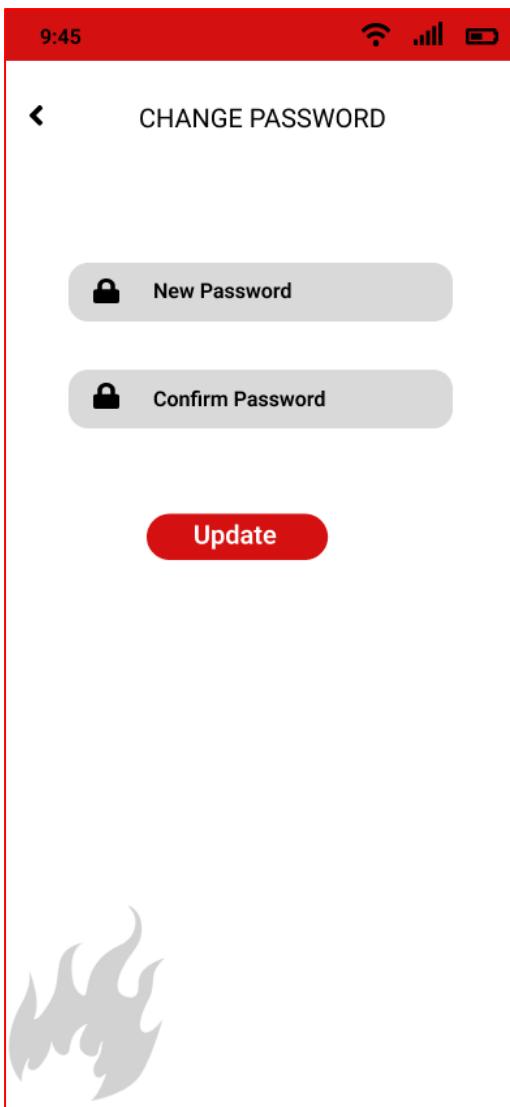
Used Algorithm	Overall Accuracy
CNN Algoritham	82.0%
Support Vector Machine(SVM)	81.50%
RNN Model(LSTM)	81.50%

Wireframes





Mobile Application UI



9:45

MY PROFILE

Profile 

Vehicle Details 

Vitz  

Manufacturer

Model Name

Vehicle Number

Fuel Type

Maruti Suzuki  

Manufacturer

Model Name

Vehicle Number

Fuel Type

Vitz  

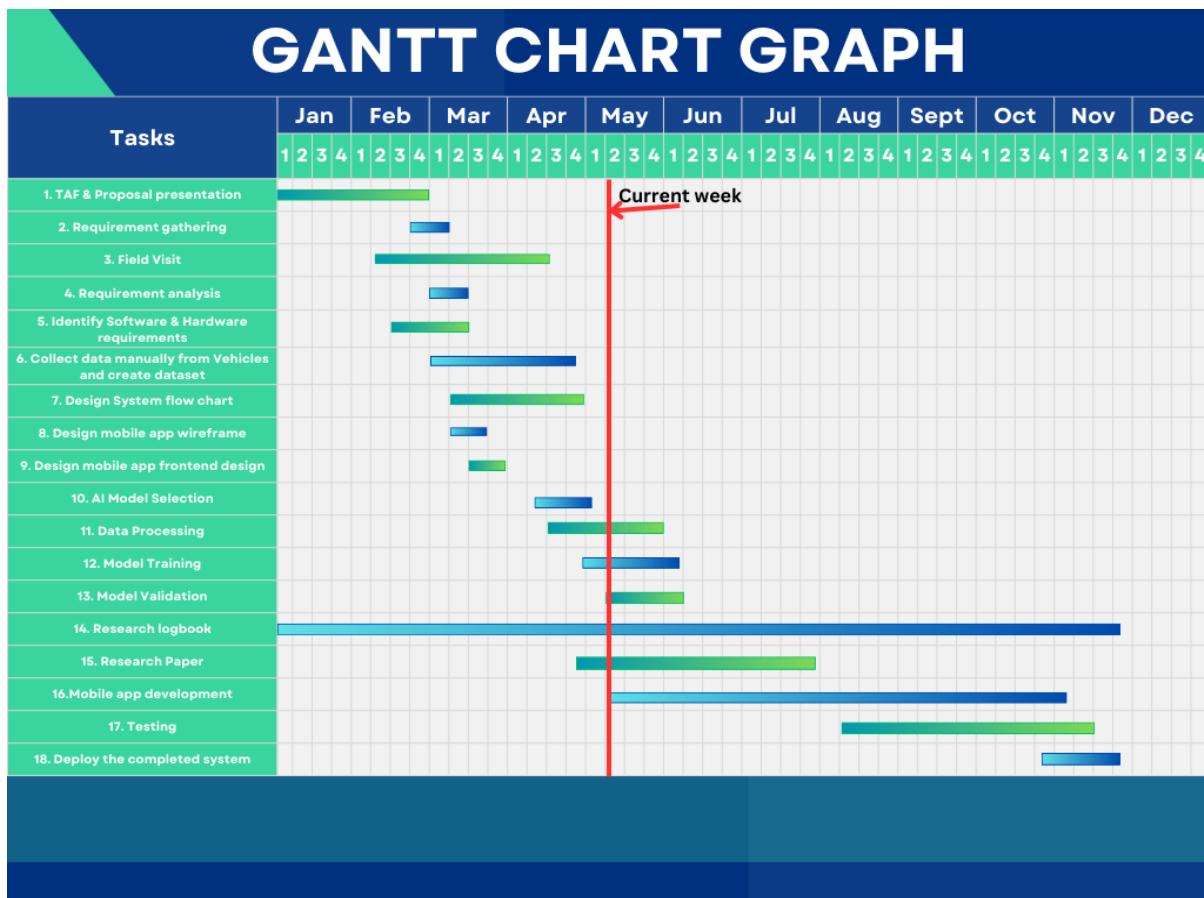
9:45

FIRE ALERT RECEIVED 

Vitz abc 



Gantt Chart



Work Breakdown Structure

