**Department of Aerospace Engineering**
**Indian Institute of Technology, Kharagpur**

# FLIGHT TESTING LAB REPORT

## Collect Atmospheric Data and Repository Engine through Drone" (CADRE-D)

## Group 1

SARTHAK BEHERA
BHANUPRIYA GUPTA
C RAMAKRISHNAN
DHURJATI V S SAI SAMPREET
INDRA KUMAR GUPTA
MANJIRI RANE
SHASHANK SHEKHAR
DEVJEET SAHU
RAGHAV AGARWAL
NIKHIL ATRAM
RAHUL RAJ

## PROJECT OVERVIEW

The project, titled **"Collect Atmospheric Data and Repository Engine through Drone" (CADRE-D)**, leverages advanced drone technology to gather critical atmospheric data. The core of the CADRE-D project is its sophisticated data collection system. This initiative aims to provide comprehensive insights into atmospheric conditions by employing various sensors mounted on drones, which are capable of reaching and assessing areas that are otherwise challenging for traditional ground-based stations. The data collected is essential for predicting weather patterns, monitoring air quality, and conducting atmospheric research, thus contributing significantly to environmental science and meteorology.

### Data Collection

Drones are equipped with several types of sensors:

**Temperature, Humidity, and Pressure Sensors**- These sensors collect basic but vital atmospheric parameters that contribute to the understanding of weather conditions and climate patterns.

**Gas Sensors**- These are utilized for detecting specific gases in the atmosphere, crucial for assessing air quality index (AQI) and monitoring pollutants.

**Wind Sensors**- To measure both the speed and direction of the wind, which are critical for weather forecasting and studying atmospheric dynamics.

Using these sensors, the drones can autonomously gather data across diverse and remote locations, providing a dense and rich dataset that reflects real-time atmospheric conditions.

## Data Analysis and Interpretation

Once data is collected, the CADRE-D project processes and analyzes this information through several steps:

**Data Processing**: Initial raw data from the drones is cleaned and formatted for further analysis.

**Data Interpretation**: Advanced algorithms and analytical techniques are applied to interpret the data, providing insights into atmospheric conditions and trends.

**Visualization**: The interpreted data is visualized through graphs and charts, making it easier to understand and communicate the findings.

**Model Integration**: The processed data is also integrated with existing atmospheric models to enhance the accuracy and reliability of weather predictions and climatic studies.

## Impact and Applications

The integration of drone technology for atmospheric data collection opens new avenues for real-time environmental monitoring and enhances the capabilities of meteorological studies. This project not only supports scientific research but also aids in practical applications such as agriculture, urban planning, and disaster management by providing timely and accurate atmospheric data. By advancing our understanding of the atmosphere, CADRE-D plays a pivotal role in addressing environmental challenges and promoting sustainable practices.

**Sensors required**:

## 1. UV Sensor ML8511 Compatible with Arduino

The ML8511 UV sensor serves as a pivotal component in our drone-based atmospheric data collection system, enabling precise measurement of ultraviolet (UV) radiation levels in the atmosphere. Its seamless compatibility with Arduino microcontrollers facilitates effortless integration into our drone platform, allowing for real-time monitoring of UV exposure levels across diverse geographical locations. By accurately detecting both UV-A and UV-B light wavelengths, the ML8511 sensor provides critical insights into environmental factors such as solar radiation intensity, ozone depletion, and atmospheric composition. This data is invaluable for assessing potential risks to human health, including sunburn, skin cancer, and eye damage, as well as for studying broader environmental phenomena such as climate change and ecosystem dynamics.

## 2. SmartElex Pressure Humidity Temp (PHT) Sensor MS8607

The MS8607 sensor represents a cornerstone of our drone-based atmospheric data collection efforts, offering comprehensive capabilities for measuring key atmospheric parameters, including pressure, humidity, and temperature. Its compact form factor and digital output make it an ideal choice for integration into our drone platform, enabling precise and real-time monitoring of atmospheric conditions during flight missions. By providing accurate data on pressure, humidity, and temperature, the MS8607 sensor empowers us to study weather patterns, climate trends, and environmental changes with unprecedented detail and accuracy. This data is essential for enhancing our understanding of atmospheric dynamics, supporting

meteorological research, and informing decision-making in areas such as agriculture, urban planning, and disaster management.

### 3. MQ-7 CO Carbon Monoxide Coal

The MQ-7 sensor plays a critical role in our drone-based environmental monitoring system, enabling the detection of carbon monoxide (CO) gas concentrations in the atmosphere. Its robust design and chemical reaction-based detection mechanism make it well-suited for deployment in diverse environments, including indoor and outdoor settings, where CO pollution poses significant health and safety risks. By providing real-time measurements of CO concentrations, the MQ-7 sensor allows us to assess air quality, identify sources of pollution, and mitigate potential health hazards associated with CO exposure. This data is vital for protecting public health, ensuring occupational safety, and informing regulatory policies aimed at reducing emissions and improving air quality standards.

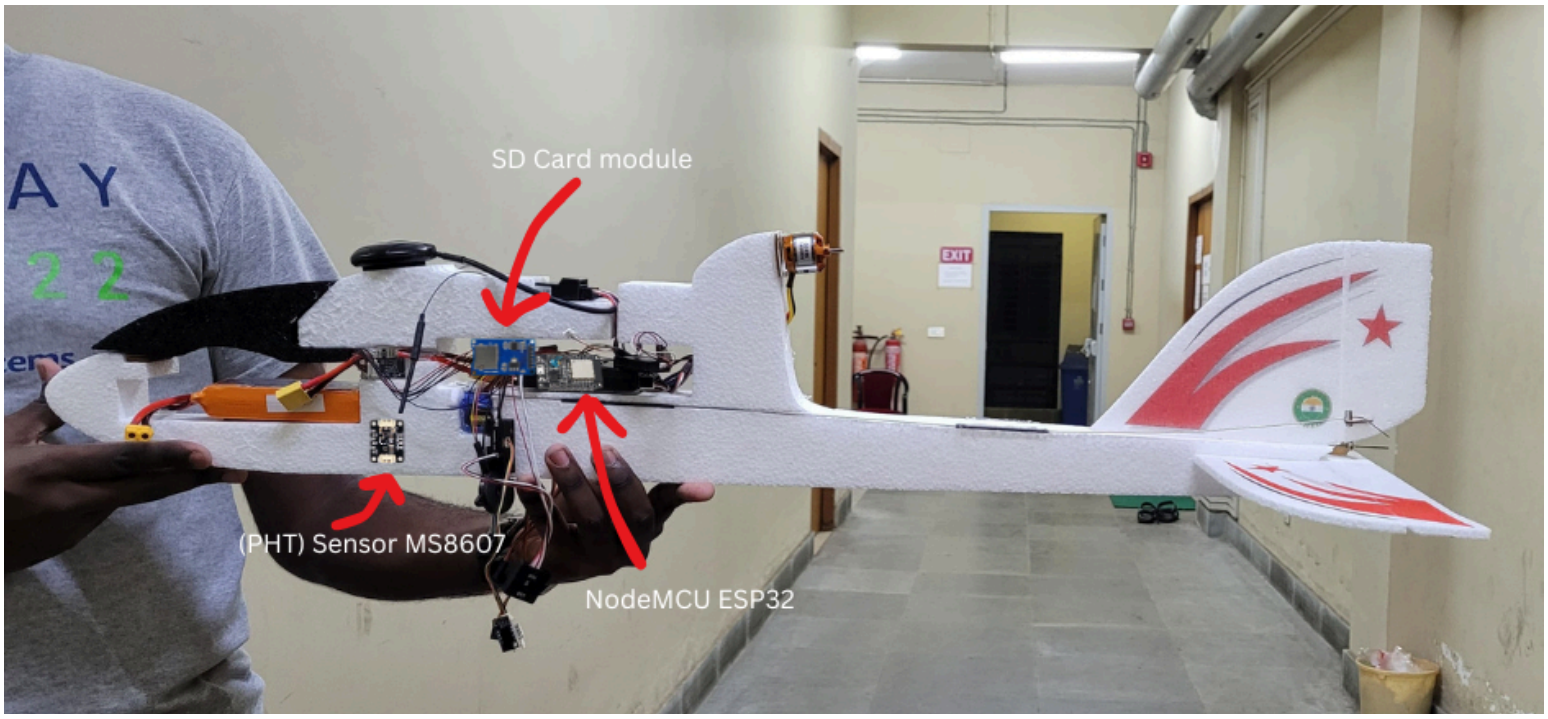
### 4. MQ 135 Gas Sensor Modules

The MQ-135 gas sensor module serves as a versatile tool in our drone-based atmospheric data collection system, offering capabilities for detecting a wide range of gases, including ammonia, benzene, alcohol, smoke, and various other harmful pollutants. Its principle of resistance change in response to gas concentrations enables us to monitor air quality and pollution levels in real-time, providing valuable insights into environmental health and safety. By integrating the MQ-135 sensor module into our drone platform, we can identify sources of pollution, assess the impact of industrial activities on surrounding ecosystems, and support efforts to mitigate environmental degradation. This data is essential for promoting sustainable development, protecting biodiversity, and safeguarding the health and well-being of communities affected by pollution.

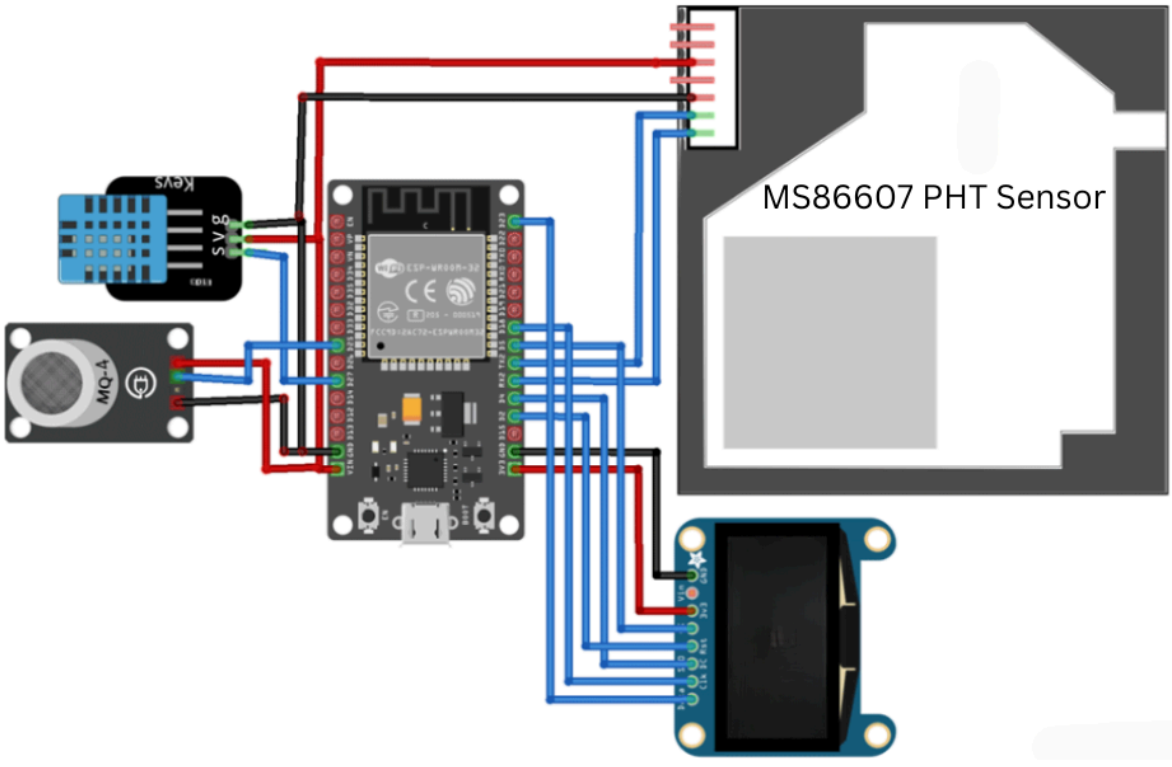
## 5. PM2.5 GP2Y1010AU0F Dust Smoke Particle Sensor

The GP2Y1010AU0F sensor plays a crucial role in our drone-based air quality monitoring system, enabling the detection of fine particulate matter (PM2.5) in the atmosphere. Its optical sensing method allows us to measure PM2.5 concentrations with high accuracy and precision, providing valuable data for assessing air pollution levels and identifying potential health risks associated with particulate matter exposure. By integrating the GP2Y1010AU0F sensor into our drone platform, we can map spatial variations in PM2.5 concentrations, identify pollution hotspots, and support efforts to mitigate the adverse effects of air pollution on public health and the environment. This data is essential for informing policy decisions, implementing pollution control measures, and raising awareness about the importance of clean air for sustainable development and human well-being.

## COMPONENTS ON DRONE




## CIRCUIT DIAGRAM



**CADRE DASHBOARD**

Creating the project dashboard with Flask web framework offered several advantages:

**Customization:** Flask allows for high customization, letting you tailor the dashboard to your project's specific needs and requirements.

**Lightweight:** Flask is lightweight, making it efficient for handling tasks like serving web pages for weather data visualization.

**Pythonic:** Since Flask is a Python micro-framework, it aligns well with data processing tasks often involved in weather monitoring projects, leveraging Python's extensive libraries for data manipulation and analysis.

**Integration:** Flask integrates smoothly with other Python libraries commonly used in weather monitoring, such as Pandas for data handling, Matplotlib or Plotly for visualization, and NumPy for numerical calculations.

**Scalability:** Flask applications can be scaled easily, allowing for potential expansion or additional features in the future as project requirements evolve.

**SQL (Structured Query Language)** was used for managing and manipulating data acquired in relational databases. It allowed us to perform tasks like inserting, updating, and deleting unwanted/faulty data, and querying data to retrieve specific information.

In frontend development, HTML and CSS were used to create the visual structure and styling of web pages.

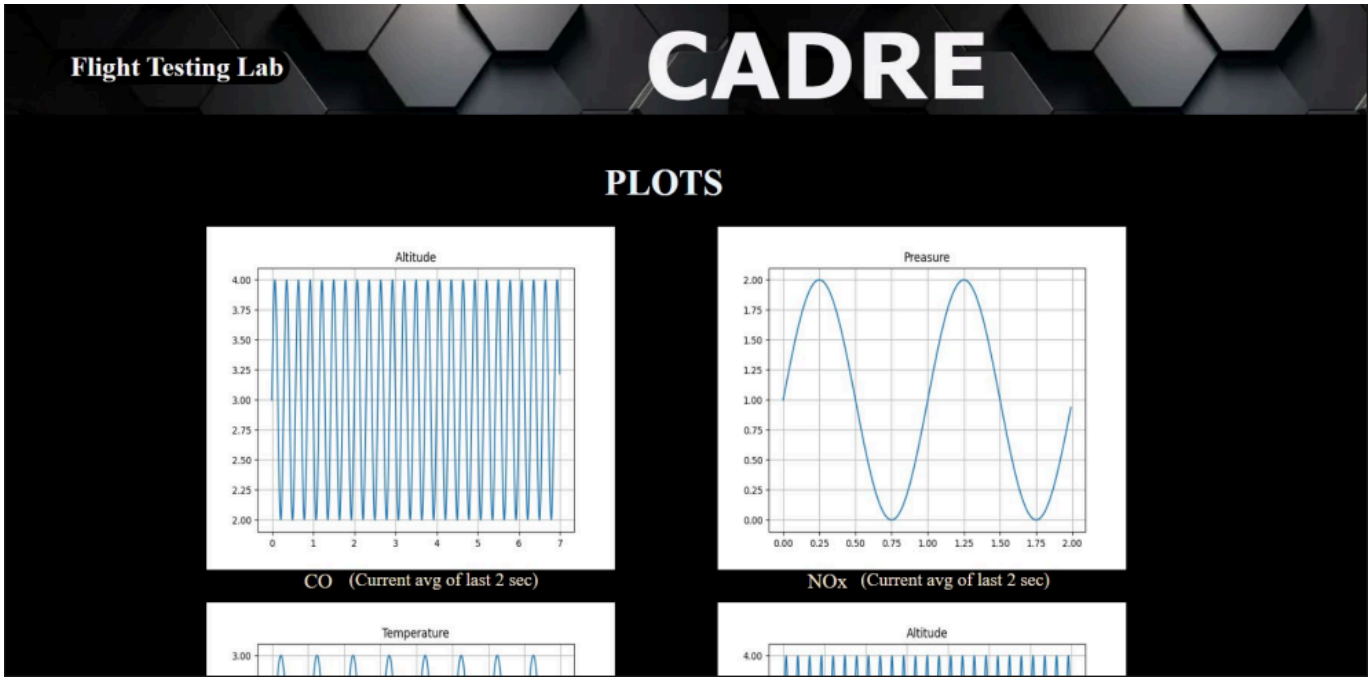
Python offers several powerful libraries for plotting graphs, charts, and updating real-time data. Here's a short note on the libraries we used:

**Matplotlib:** Matplotlib is a comprehensive plotting library that provides a MATLAB-like interface for creating static, interactive, and publication-quality visualizations. It supports various plot types, including line plots, scatter plots, bar charts, histograms, and more. Matplotlib is highly customizable and widely used for data analysis and visualization tasks.

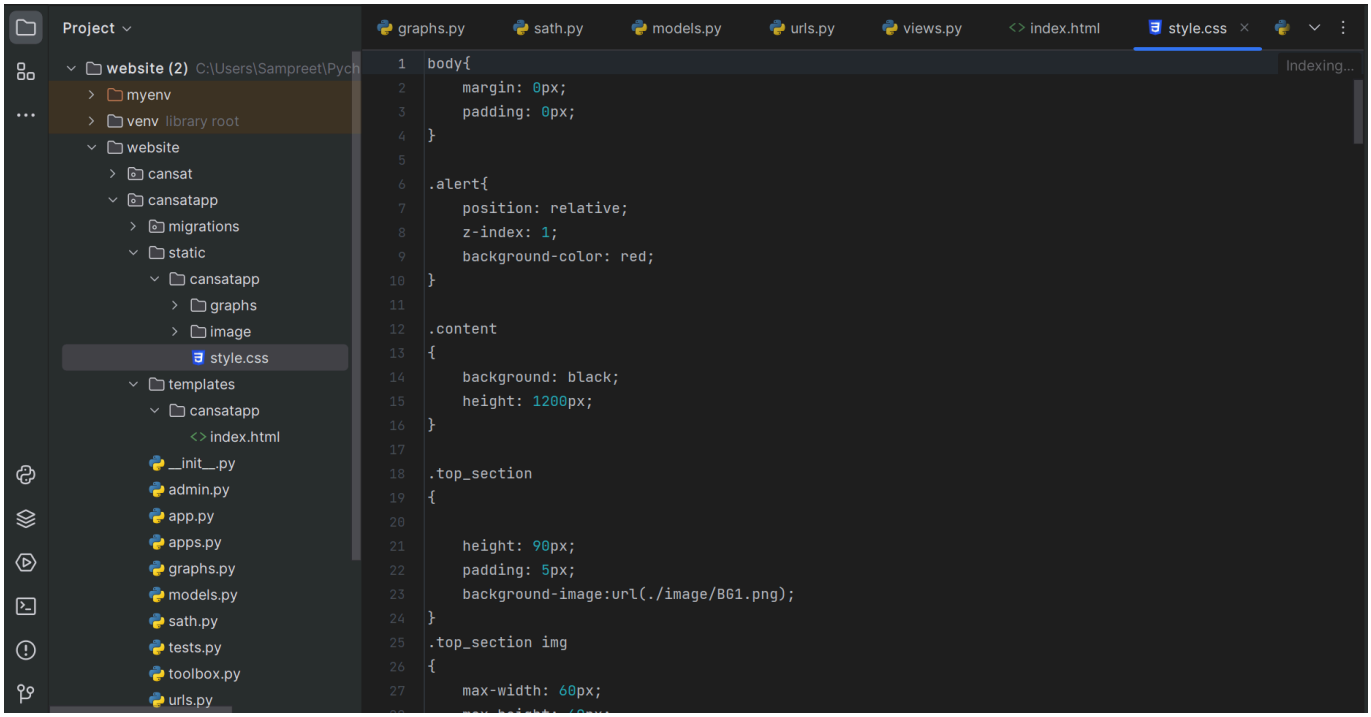
**Seaborn:** Seaborn is built on top of Matplotlib and offers a high-level interface for creating informative statistical graphics. It simplifies the process of creating complex visualizations by providing easy-to-use functions for common tasks like visualizing distributions, correlations, and categorical data. Seaborn also offers built-in themes and color palettes to enhance the aesthetics of plots.

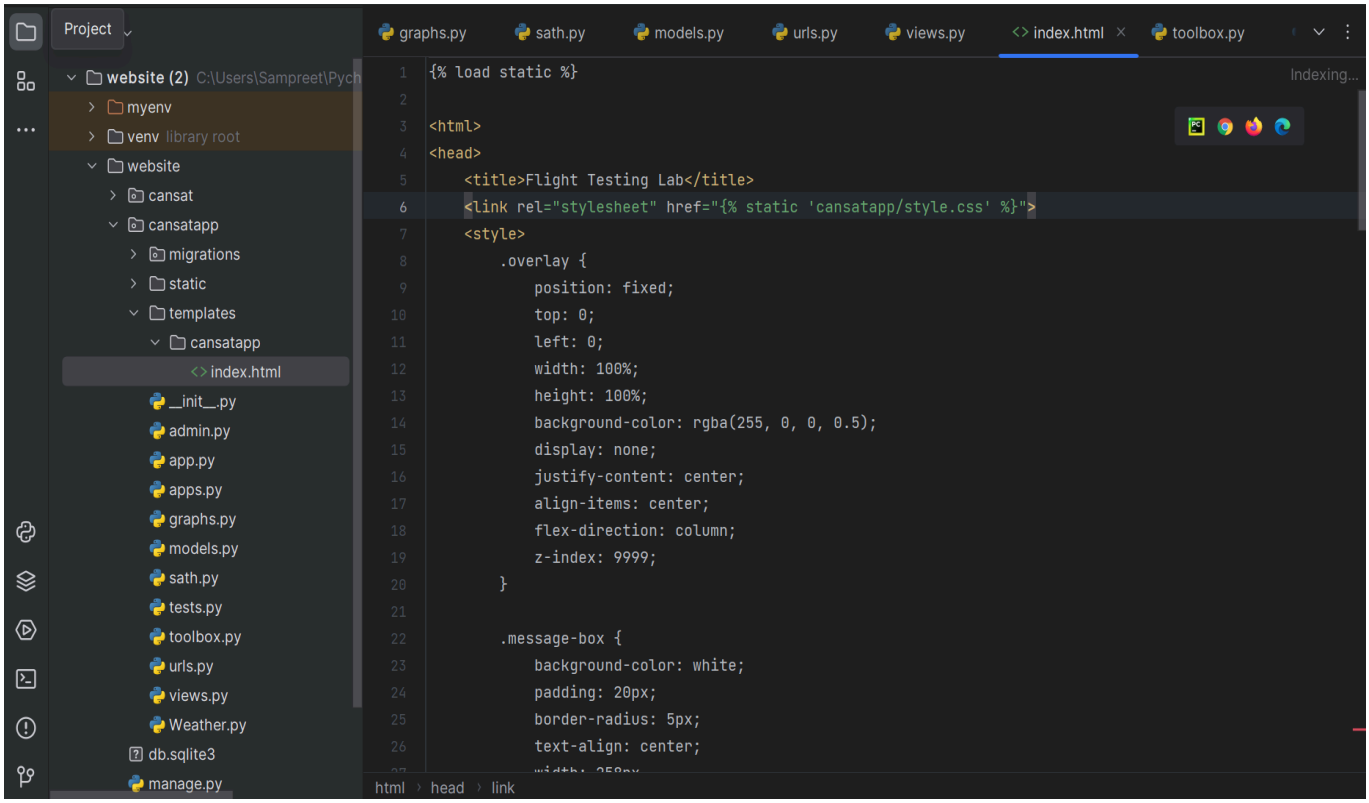
**Plotly:** Plotly is a versatile library for creating interactive and web-based visualizations. It supports a wide range of plot types, including line charts, scatter plots, bar charts, heatmaps, and more. Plotly generates HTML-based plots that can be embedded in web applications or shared online.

These libraries offer a range of capabilities for plotting graphs, charts, and updating real-time data, catering to different use cases and preferences.

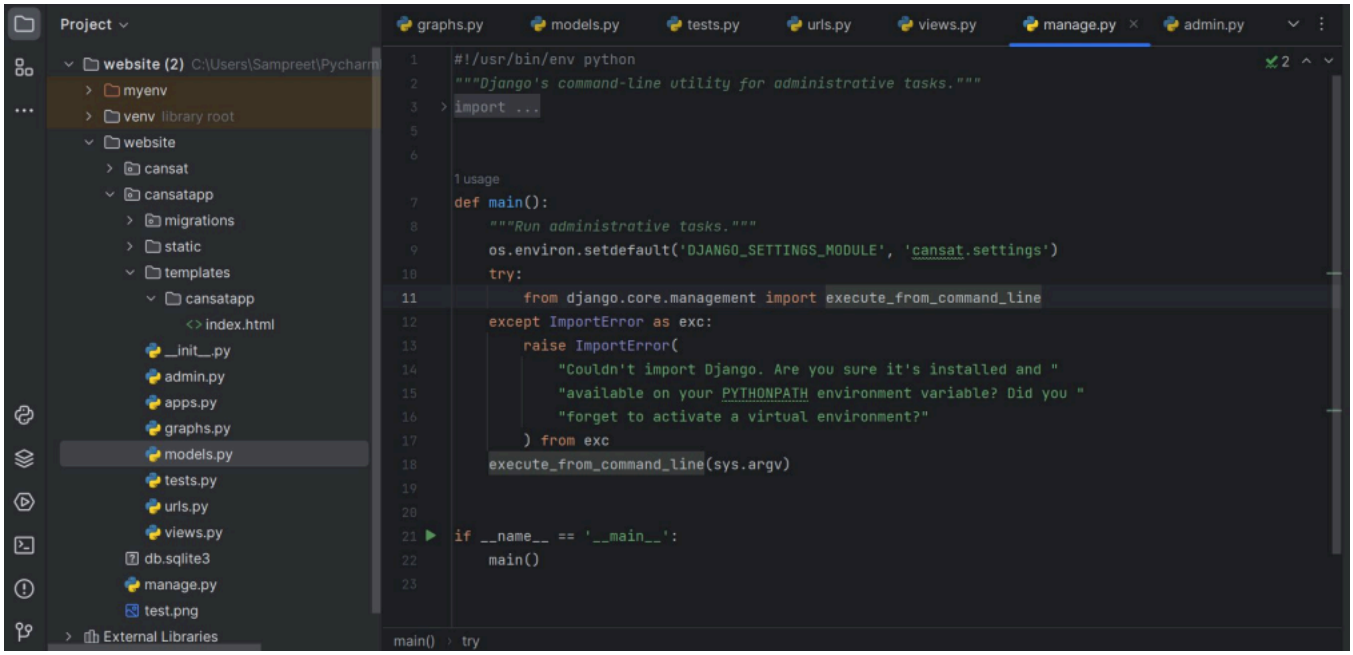


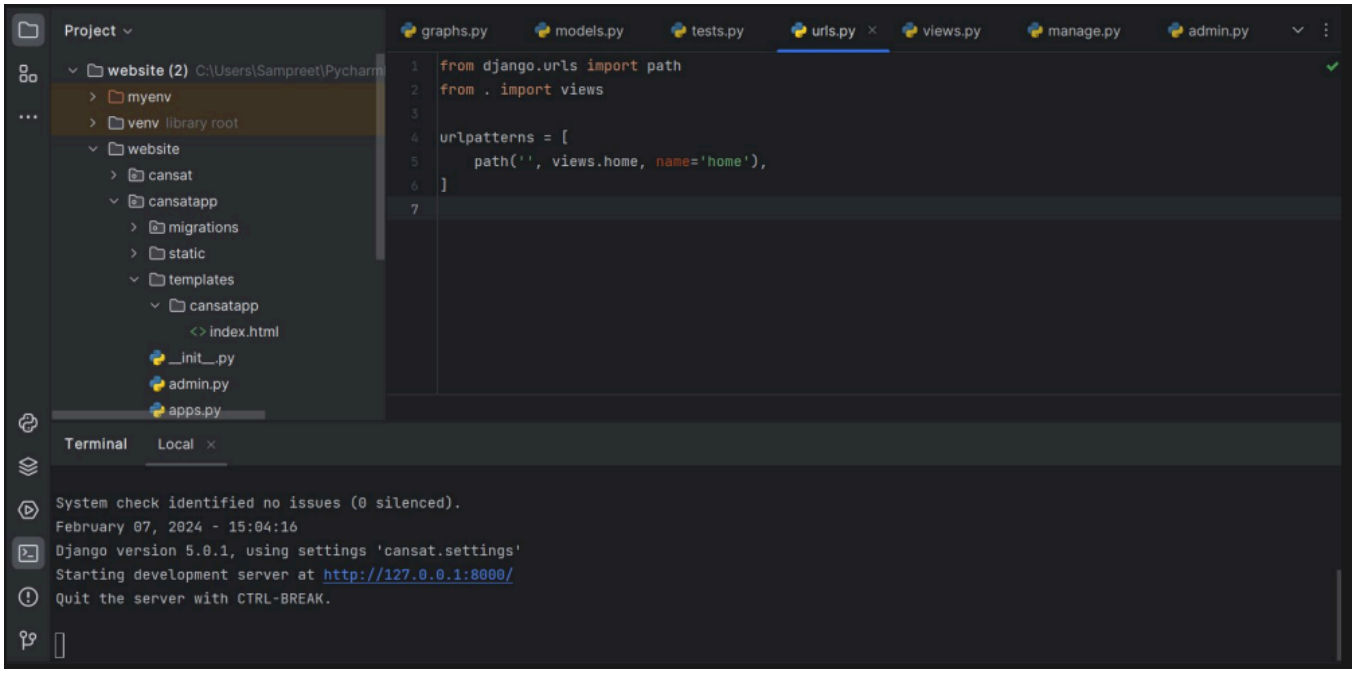

# Glimpse of the Frontend Code:






# Glimpse of the Backend code:






The entire code can be accessed via this Github repository link: https://github.com/staraquarius72/CADRE

**WEATHER PREDICTION MODEL**

We used time series analysis to forecast the daily weather conditions. A strong predictive model is built by analyzing historical weather data, which includes variables like temperature, humidity, precipitation, and wind speed. The findings demonstrate the effectiveness of time series analysis in predicting daily weather. The findings have implications for a variety of sectors, including agriculture, transportation, and urban infrastructure, as they promote resilience and sustainability in the face of changing weather patterns.
The aim of the weather prediction model is to accurately forecast temperature and other variables based on various weather parameters such as wind speed, wind direction, atmospheric pressure, humidity and temperature collected through previous datasets.

**Goal**: The goal is to develop accurate predictions using both traditional time series forecasting methods and deep learning techniques like LSTM (Long Short-Term Memory) networks. By comparing the performance of different models, the aim is to find the most accurate and robust forecasting method for weather prediction.

We will integrate the model with our hardware model to collect datasets and forecast the measured parameters i.e. temperature, humidity, pressure etc.

**Dataset**: The dataset is extracted from Weather UndergroundAPI (wunderground.com) (link)
Location: New Delhi

Timeline: 1996-2016

Code: Colab Notebook Link

## DESCRIPTION OF THE DATASET

**Dependent Variable**: Temperature in degree Celsius (°C)

**Independent Variables**: The dataset includes various weather-related features that might influence temperature. These features are considered independent variables for the time series analysis.

- Windgust: Wind gust is the sudden increase in speed of wind in kph.
- Vis: Visibility (in km)
- Dewpoint in degree Celsius: the atmospheric temperature at which water droplets begin to condense.
- Snow in inches: the height of snowfall.
- Windspeed in kph: the speed of the wind.
- Wdird: Wind direction in degrees.
- Pressure is in millibars (mb): Atmospheric pressure.
- Humidity is in percentage.
- Windchill in kph.
- Heatindex in degrees Celsius.
- Precipitation in mm

## PREPROCESSING OF THE DATASET

The weather data underwent several steps to prepare it for time series analysis. Initially,columns containing significant missing values (precipitation, windchill, etc.) were excluded to avoid introducing artificial data. The remaining data, likely hourly measurements, was resampled to daily averages to create a more manageable time series suitable for daily forecasting.

To capture daily temperature variations, two new features were created: minimum temperature (minTemp) and maximum temperature (maxTemp). This involved grouping the data by date and extracting these values. Finally, any remaining missing data points were filled using forward filling, assuming gradual temperature changes within a

few days. Lastly, outliers exceeding realistic temperature ranges for Delhi (verified using Wikipedia) were removed to ensure the data's accuracy.

**STATIONARITY CHECK**

Stationarity analysis was meticulously conducted on the preprocessed temperature data, where the initial time series plots revealed apparent seasonal patterns with distinct peaks and troughs, suggesting potential non-stationarity. However, to rigorously determine the underlying statistical properties of the dataset, several tests were employed. The Augmented Dickey-Fuller (ADF) test, a widely used method for testing stationarity, returned a highly significant p-value close to zero, which firmly rejected the null hypothesis of non-stationarity at a confidence level exceeding 95%. This result strongly indicated that the data was stationary after accounting for preprocessing adjustments. Complementing the ADF test, the KPSS test was also administered, which further corroborated the stationarity of the dataset with its high p-value, thus aligning with the findings from the ADF test. Despite the visual seasonality observed in the initial plots, the combined outcomes from these statistical tests confirmed the stationarity of the temperature data after outliers had been removed and preprocessing had been applied.

Building upon the foundational stationarity analysis, further exploration of the data was conducted using Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots. Initially, these plots suggested the appropriateness of an ARIMA(1,0) model. However, a more detailed analysis revealed a significant seasonal component within the data, with the ACF plot showing a pronounced seasonal peak at lag 365, a characteristic reflecting Delhi’s climatic rhythm. This discovery was pivotal for the selection of a time series forecasting

model, leading to the integration of both trend components through an ARIMA model and seasonality adjustments to account for the annual temperature fluctuations. To ensure robust model training and validation, the dataset was strategically split into training and testing sets, with 80% allocated for training (X_train, y_train) and the remaining 20% reserved for testing (X_test, y_test). This division was designed to optimize the learning phase on a comprehensive portion of the data while enabling thorough evaluation on unseen data to assess the model's accuracy and generalizability in real-world scenarios.

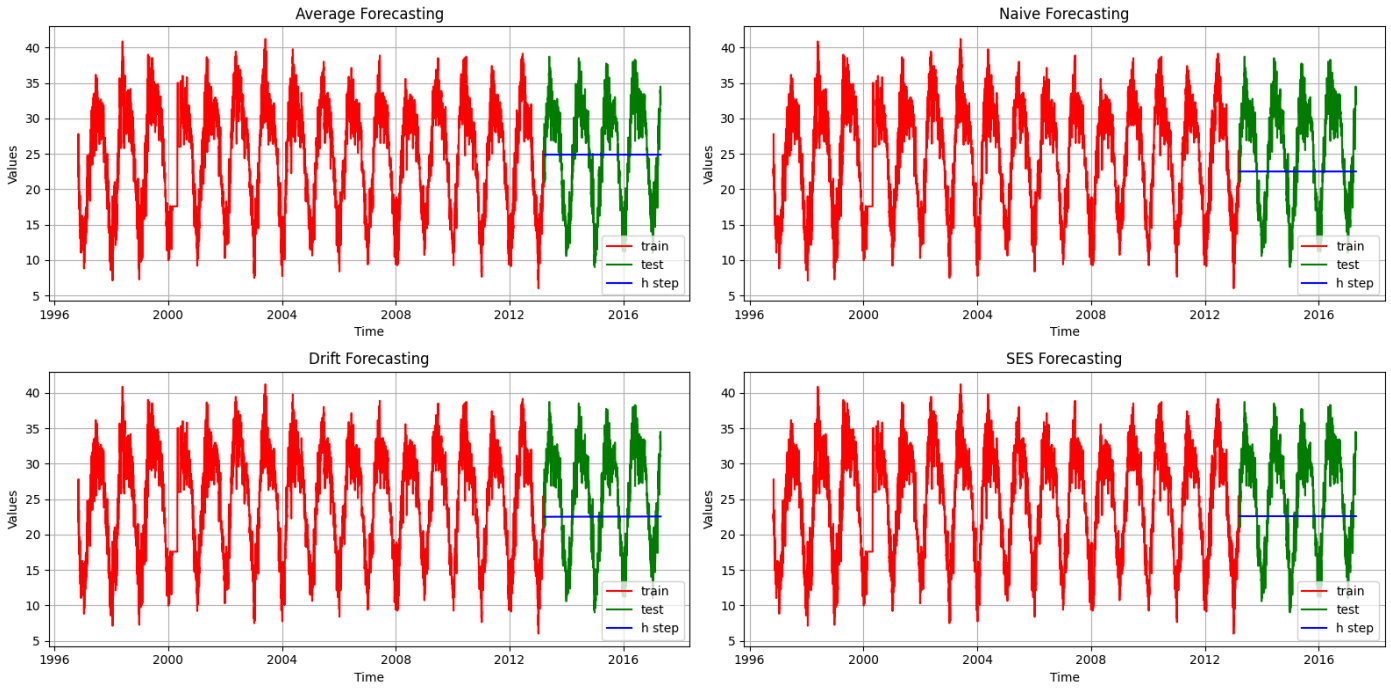
**BASE MODELS PERFORMANCE**

To establish a benchmark for more complex models, the performance of four basic forecasting approaches was evaluated on the test set: Average, Naive, Drift, and Simple Exponential Smoothing (SES).

The Average model simply predicts the average temperature for all future days. The Naive model predicts the last observed temperature for all future days. The Drift model predicts a constant linear increase or decrease based on the historical trend. Finally, SES is a basic smoothing technique that assigns weights to past observations, with more recent data receiving higher weights.

The Mean Squared Error (MSE) and Root Mean Squared Error (RMSE) were used to quantify the forecasting errors of each model. Additionally, Mean Absolute Error (MAE) was calculated to assess the average magnitude of the errors.

The results revealed that all baseline models exhibited significant shortcomings in predicting Delhi's temperatures. Visualizations confirmed these findings, as the predicted values often deviated significantly from the actual temperatures. This indicates that these models are not suitable for capturing the inherent seasonality present in weather patterns.




## HOLT-WINTERS METHOD

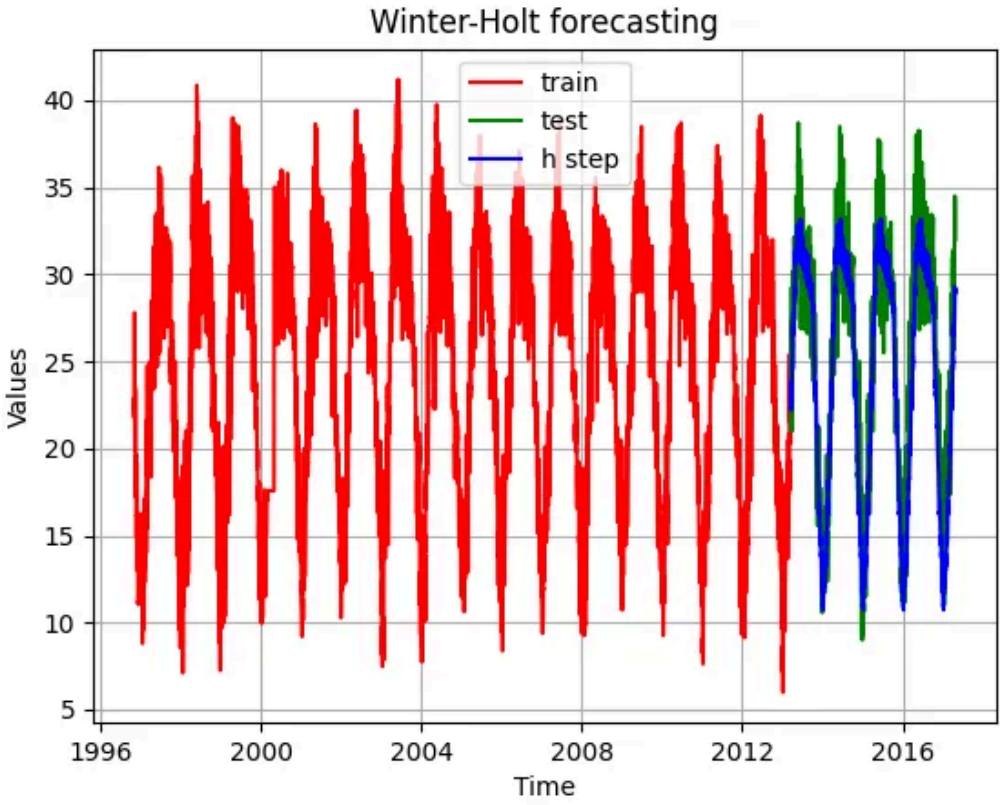
This method improves forecasts over baseline models by considering level, trend, and seasonality.

The statsmodels.tsa.holtwinters library trained the model on y_train. Seasonal='add' specified an additive seasonal component, and seasonal_periods=365 accounted for the data's annual seasonality.

ForecastES for the test set (y_test) were generated by the trained model. The model accurately predicted temperature patterns, showing seasonality.

Results summary (rounded to two decimal places):

- MSE: 7.63
- RMSE: 2.76
- MAE: 2.19



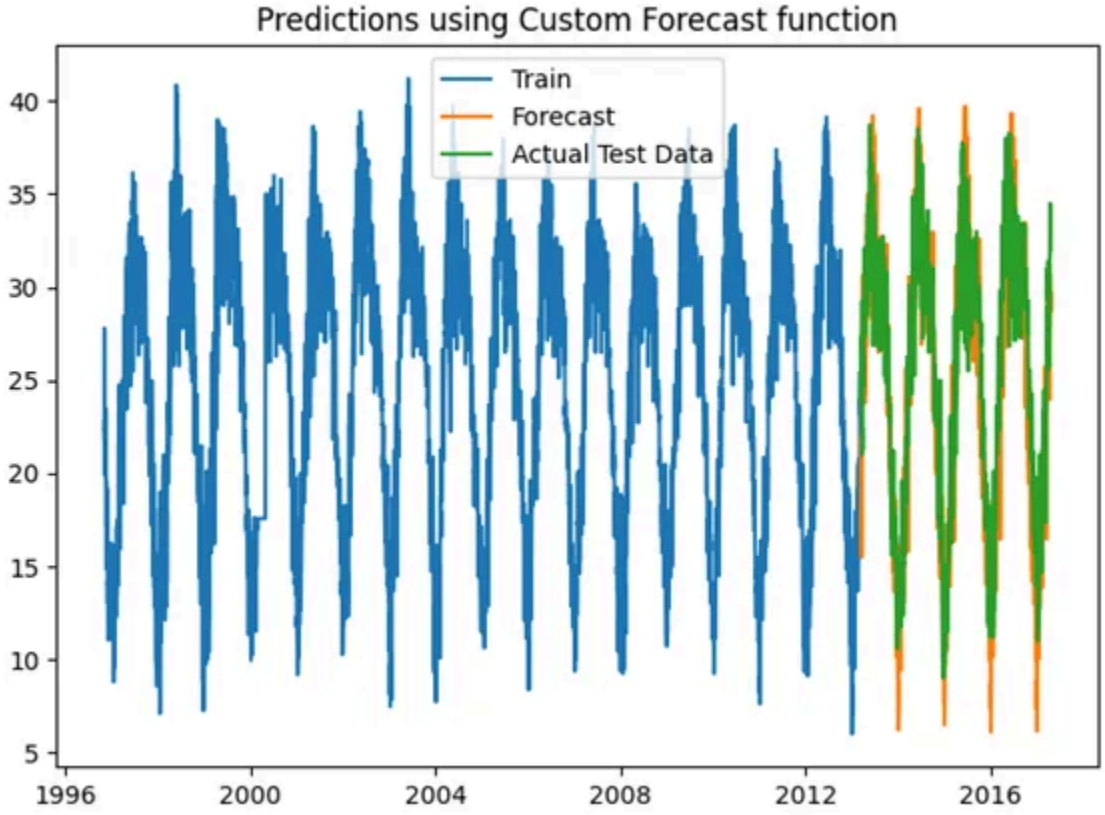

**ARIMA MODEL**

ARIMA is a statistical model used for forecasting time series data. It is a two-step process. First, the order of the model is determined using a technique called Generalised Partial Autocorrelation (GPAC). This step helps identify the number of past values that influence the current value and the number of past forecast errors that need to be considered. Second, the parameters of the model are estimated using the Levenberg-Marquardt algorithm. This step determines the coefficients of the model equation.

The ARIMA model with a first-order seasonal differencing and an ARMA process of order (7, 0) was found to be the best fit for the data. This means that the current value is influenced by the past 7 values and there are no moving average terms. However, the confidence interval analysis of the estimated parameters showed that some of the AR terms are insignificant.

Results summary (rounded to two decimal places):

- MSE: 12.20
- RMSE: 3.49
- MAE: 2.77




## DEEP LEARNING MODEL

To apply deep learning model for this dataset, LSTM is used.

1. Two-layer LSTM networks were used.
2. The first layer contains the activation function as ‘Relu’ with a total of 64 neurons.
3. The second layer contains 50 neurons.
4. A total of 10 epochs were run on the personal computer. (Due to computational limitations, epochs could not be increased more than 10).

Long Short-Term Memory (LSTM) networks have been designed to solve the vanishing gradient problem in standard RNNs.

Because they can use past data to predict sequential data like temperature readings, RNNs are powerful. They have trouble with long-term dependencies. Imagine predicting tomorrow's temperature from last year's readings. As data propagates through a standard RNN, distant data points can lose influence.

Memory cells help LSTMs solve this problem. This cell remembers only relevant past information for the current prediction, like a gated valve. How it works in brief:

1. Forget Gate: The network considers the previous cell state after receiving new data. The forget gate chooses which cell state information to forget (assign a value close to 0) and which to retain, assigning a value close to 1.

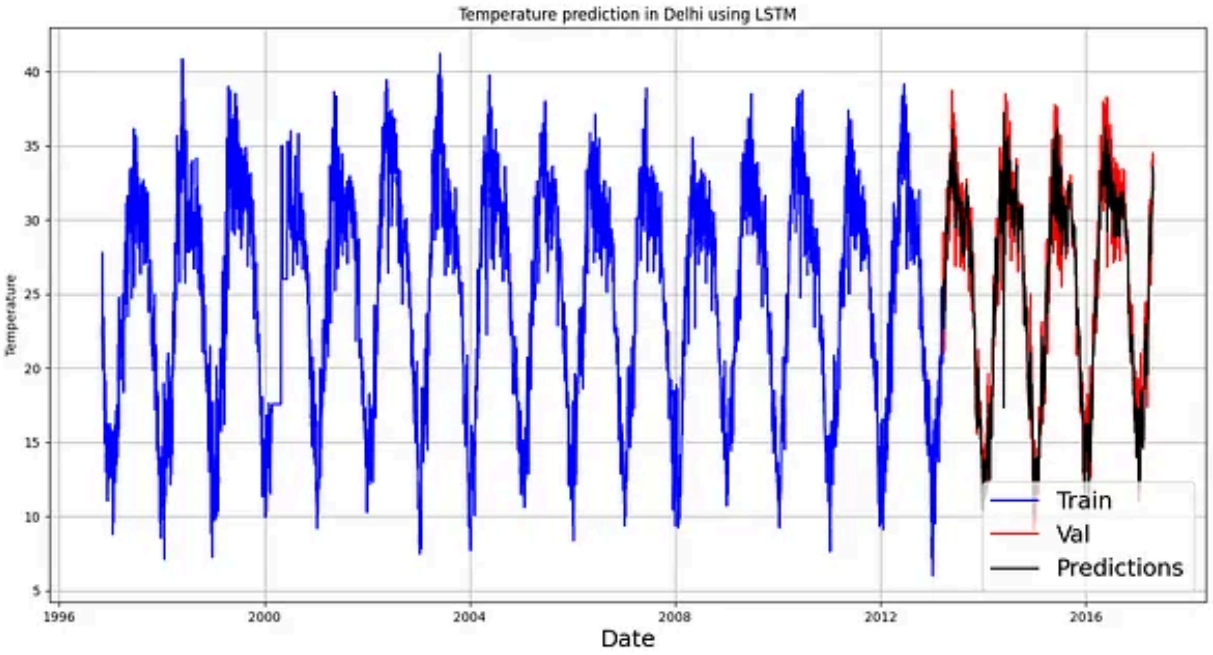
2. Input Gate: The network processes new data and forget gate output. The cell state stores new information determined by the input gate.

3. Update on Cell State: The network creates a cell state candidate from the forget gate's output and the input gate's information. The old state and candidate value make up the cell state update.

4. Output Gate: The network outputs information based on cell state and previous hidden state. This hidden state carries past information into the next time step.

LSTM’s Performance Metrics

- MSE: 3.08
- RMSE: 1.75
- MAE: 1.34




## CONCLUSION

The "Collect Atmospheric Data and Repository Engine through Drone" (CADRE-D) project effectively harnesses drone technology for sophisticated atmospheric data collection. Utilizing a diverse array of sensors, CADRE-D enhances our understanding of weather patterns and air quality. The project employs advanced data analysis techniques, including visualization and integration with existing models, to improve weather prediction accuracy. Additionally, the use of modern technology like the Flask web framework and Python libraries for real-time data handling and visualization exemplifies CADRE-D's innovative approach. Ultimately, this project not only advances meteorological research but also supports informed decision-making across various sectors, promoting environmental resilience and sustainability.

**INDIVIDUAL CONTRIBUTION**

Frontend-

1. Devjeet Sahu
2. Sarthak Behera

Backend-

1. Dhurjati V S Sampreet
2. Indra Kumar Gupta
3. Nikhil Atram

Literature Review-

1. Manjiri Rane
2. Shashank Shekhar
3. Raghav Agarwal
4. C Ramakrishnan
5. K Rahul Raj

Machine Learning model-

1. Indra Kumar Gupta
2. Bhanupriya Gupta

Material Assembly and Circuit-

1. Indra Kumar Gupta
2. Sarthak Behera
3. Bhanupriya Gupta