PROJECT LOG - AIRAWARE SMART AIR QUALITY PREDICTION SYSTEM

-by N.N. Sai Sreya

1.DAY 1

Topic: AirAware Smart Air Quality Prediction System

1.1 Overview:

AirAware is a smart air quality monitoring and prediction system that uses real-time pollution data and machine learning. It focuses on predicting PM2.5, CO2, NO2 levels and giving awareness about hazardous air conditions. The project includes frontend, backend, database, dashboards, and ML integration.

1.2 Objectives:

- Collect dataset containing PM2.5, CO2, NO2, etc.
- Forecast air quality using ML models such as Linear Regression, Random Forest, SVM.
- Compare Predicted vs Actual values.
- Show 3-4 months of data with analytics on a dashboard.
- Alert users if air quality becomes hazardous.
- Use Kaggle datasets or similar sources.
- Preprocess data (cleaning, normalization) and store in database.
- Build UI + API-based full-stack application.

1.3 Milestones:

1st – 25%

2nd – 50%

3rd – 75%

4th – 100%

Each milestone must include a working demo.

2. DAY 2

2.1 Team Formation:

Team 1 – N.N.Sai Sreya,Rajalaksmi, Rahul, Lokesh, Divija Nandana.

Dataset chosen: Delhi Air Quality Dataset (Kaggle).

2.2 Key Decisions:

- Build AirAware website to predict future pollution levels.
- Predict air quality for upcoming years and show trends in dashboard.
- React recommended for a beautiful UI dashboard.
- Dashboard to include last year’s air quality, current quality, and predictions.
- API calls for communication between frontend and backend.
- Backend receives payload from frontend, processes it, and returns prediction results.

2.3 Additional Features:

- Graph of past few years.
- Current weather display.
- Potential integration with Google weather API.
- Accuracy visualization using heatmaps.
- Dummy sensor data input for now.
- Future enhancement: real sensors for surrounding air quality.

3. DAY 3

3.1 Topics Covered:

- Pandas, NumPy, Matplotlib
- Heatmap for model accuracy

3.2 API Concepts:

Two main Python APIs: Flask and FastAPI

- FastAPI:

```
from fastapi import FastAPI
app = FastAPI()
@app.get("/")
```

- Flask API:

```
from flask_api import FlaskAPI
from flask import request
app = FlaskAPI(__name__)
@app.route("/", methods=['GET'])
```

3.3 Key Points:

- GET method sends payload in header.
- POST method sends payload in body.
- Postman is used to test backend API endpoints.
- API connects frontend and backend through requests and responses.
- Frontend payload structure must be known before writing backend.
- Changing payload breaks API functionality.
- FastAPI supports async, better performance, high concurrency.

3.4 Additional:

- WebSocket, streaming responses.
- Deployment discussions pending.

3.5 Task:

Build basic working version of AirAware dashboard with API connectivity.

4. Day 4

Topic: Git and Version Control

4.1 Commands learned:

- git add . - Adds all new/changed files to staging.
- git add <file> - Adds only the selected file to staging.
- git commit -m "msg" - Saves changes with a message.
- git push - Uploads your commits to the GitHub repo.
- git branch - Shows the branch you are currently on.
- git branch –all - Shows all branches in the project.
- git fetch –all - Gets updates from all remote branches.
- git pull - Fetches and merges latest changes from the remote branch.
- git stash - Temporarily saves uncommitted changes without pushing.
- git checkout -b <branch> - Creates and switches to a new branch.
- git checkout <branch> - Switches to an existing branch.

4.2 Virtual Environment:

- python -m venv venv
- venv/Scripts/Activate
- pip install -r requirements.txt
- deactivate

5. Day 5

Topic: Database Concepts

5.1 DB & DBMS

CRUD operations: Create, Rename, Update, Delete

5.2 Types of Databases:

Structured – MySQL, SQL, PostgreSQL

Unstructured – MongoDB

5.3 Normalization Forms:

1NF, 2NF, 3NF, BCNF, 4NF, 5NF

Concepts: Partial dependency, Transitive dependency

5.4 Keys:

Primary key, Foreign key, Composite key

5.5 Data Types:

- Master data (static)
- Transaction data (dynamic)

5.6 Frontend Pages:

- Analytical Dashboard
- About Page
- Report Page
- Login Page
- Admin Page

5.7 AI Tools:

OpenAI, Gemini, Grok, Llama, Vertex AI

5.8 ML Concepts:

- Supervised (Regression, Classification)
- Unsupervised (Clustering)
- Reinforcement learning

5.9 NLP Concepts:

- Embeddings
- Vectors
- NLTK

6. Day 6

- Project Progress Planning

6.1 Milestone Deadlines:

- 25% on 20-21 November
- 50% by 30 November
- No UI template changes after 30 November
- Final output on 1 December
- Then enhancements begin

6.2 Documentation:

- 45-page detailed project document required
- Each page should include Day and Topic
- Include Git, API, DB, Project overview, ML, and daily logs

6.3 GitHub:

- Push code to main repo

- Team repo for demo
- Personal repo for presentation
- Add teammates as contributors
- Maintain backup copy

6.4 Corporate Culture:

- Learn to communicate freely
- Understand professional workflows

7. Day 7:

7.1 Doubt Clearance & Additional Instructions

Day 7 focused mainly on clarifying students' doubts related to the AirAware project, the backend–frontend workflow, API integration, dataset usage, and Git practices. The instructor addressed questions regarding Python APIs (Flask/FastAPI), UI design guidelines, database selection, and how to structure payloads between frontend and backend.

Additional instructions were also given for the project timeline, including how to approach Milestone-1, how to organize the GitHub repository, and how each team should maintain transparency in UI development and backend logic.

Students were reminded to follow the planned structure, avoid unnecessary UI template changes after deadlines, and ensure the project was progressing toward the 25% milestone target.

On this day, we also created both the team GitHub repository and our individual personal GitHub repositories for code collaboration and documentation.

Personal Github repository: https://github.com/SaiSreya96/saisreya_personal_repo

Team Github repository: https://github.com/SaiSreya96/grp1_team_repo_infosys

8. Day 8

8.1 Preparation for Milestone-1 Presentation

On Day 8, students were given dedicated time to prepare for the Milestone-1 presentation scheduled for the next day. No new topics were taught, as the entire session was focused on getting ready for the first project review.

Since this day was fully allotted for milestone preparation, I utilized the time to work on the essential components required for the presentation. I updated and organized the **log document**, created and refined the **project PowerPoint presentation**, and made the necessary **UI changes** to improve the overall appearance and flow of the AirAware dashboard.

The main purpose of Day 8 was to ensure that the project was in a presentable and polished state for the milestone evaluation, and I worked on documentation, presentation, and UI enhancements accordingly.

9. DAY 9

Milestone 1

10.DAY 10

Milestone 1

11. Day 11

11.1. Artificial Intelligence (AI)

- AI is a set of methods that enables machines to perceive data, perform tasks, and make predictions—tasks that normally need human intelligence.

11.2. Key Subfields of AI

- ML – Machine Learning
- DL – Deep Learning
- NLP – Natural Language Processing
- RL – Reinforcement Learning
- Knowledge-based systems

11.3. Deep Learning (DL)

- DL uses **neural networks** like:
  - RNN
  - CNN

  - LSTM
  - Transformers

11.4. ANN — Artificial Neural Network

11.5. CNN — Convolutional Neural Network

11.6. LSTM — Long Short-Term Memory

- Has 3 gates:
  - Forget gate
  - Input gate
  - Output gate

11.7. ML Workflow

- Collect data
- Prepare & engineer features
- Train the model

12. Day 12

12.1. NLP — Natural Language Processing

- Helps machines understand, interpret, and generate human-like language.
- Main challenges: emotion, ambiguity, meaning, grammar
- Example application: Chatbots

12.2. NLTK Library

- Used for basic NLP operations
- Command: pip install nltk

12.3NLP Tasks

a). Tokenization

- Breaking text into words

- Code:
  from nltk.tokenize import word_tokenize

b). Stop Words

- Words with little meaning → removed
- Import: from nltk.corpus import stopwords

c). Stemming

- Converts words to **base/root**
- Example: playing → play
- Import: from nltk.stem import PorterStemmer

d). Lemmatization

- Also gives root word
- Example:
  - stem("better") → better
  - lemma("better") → good
- Import: from nltk.stem import WordNetLemmatizer

e). POS Tagging (Part of Speech Tagging)

- Identifies noun, verb, adjective etc.
- Uses: word_tokenize

f). Named Entity Recognition (NER)

- Extracts names, places, dates, organizations

12.4Text Preprocessing Steps

a) Lowercase

b) Remove punctuation

c) Remove numbers

d) Remove extra spaces

e) Tokenization

f) Stop word removal

g) Lemmatization / stemming

12.5 Other ML Concepts Mentioned

- Logistic Regression: used for sentiment analysis
- SVM — Support Vector Machine
- TF-IDF — Term Frequency Inverse Document Frequency
- Term Frequency (TF): how often a term appears
- Inverse Document Frequency (IDF): importance of term across documents

13. Day 13

13.1. TF–IDF

- TF (Term Frequency): Measures how often a word appears in a document.
- IDF (Inverse Document Frequency): Measures how rare a word is across all documents.
    - If a word appears in 1/1000 documents → High IDF
    - If a word appears in 999/1000 documents → Low IDF

13.2. Support Vector Machine (SVM)

- SVM comes under supervised learning.
- Mainly used for classification.
- **Margin:** Distance between the separating line and the closest data points from each class.
- Two types of margins:
    - Hard Margin**:**
- Perfect classification
- No misclassification
- Data is clean, no noise
    - Soft Margin**:**

    - For non-linear separations
    - Allows some misclassification
    - Boundary line is not straight

14. Day 14

14.1. RL (Reinforcement Learning)

- RL stands for Reinforcement Learning.
- Concepts:
    - Agentic AI / AI Agent
    - An agent learns by interacting with the environment and receiving rewards.
    - Multi-agent architecture: More than one agent working together.
    - Agents can operate with or without human involvement.
    - Mention of Autogen (agent framework).

14.2. CSS Flexbox

- Flexbox is a layout design method.
- Intrinsic size: Element's original size.
- Extrinsic size: Size given by the developer.

14.3. Handling Overflow (CSS)

- CSS overflow properties:
    - visible
    - hidden
    - scroll
    - auto
- overflow-x and overflow-y can be set separately.
- Minimum and maximum sizes can be controlled.

14.4. Box Sizing

- Includes content, margin, padding, border.
- Two models:
  - Content-box
  - Border-box

14.5. Layout Design

- Approaches:
  - CSS Grid
  - Flexbox
- Flex attributes include:
  - display: flex, inline-flex, grid, none
  - flex-direction

14.6. HTML & CSS Basics

- Introduction to HTML basics.
- Basic tags with small examples.

14.7. Python Integration

- Using Python with:
  - Flask API
  - FastAPI

14.8. Git & GitHub Differences

- Noted differences between Git and GitHub.

15. Day 15

15.1. Flex Wrap (CSS Flexbox)

- flex-wrap is used to adjust how items move to the next line.
- Types:
  - nowrap (default): Items stay in one line.
  - wrap**:** Items move to the next line when space is insufficient.
  - wrap-reverse: Items wrap, but in reverse order.

15.2. Flex Container Concepts

- A <div> can be made a flex container.
- All direct children become flex items.
- A *cards container* can also be made a flex container.

Align Self (for individual items)

- flex-start
- center
- flex-end
- stretch
- auto (default)

15.3. Summary of Flex-Wrap & Align-Self

Flex-Wrap

1. nowrap
2. wrap
3. wrap-reverse

Container → align-self options

1. flex-start

2. center
3. flex-end
4. stretch
5. auto

15.4. Order Property

Used to control the order of flex items.

1. 0 (default) — comes first
2. positive values — appear later
3. negative values — come before 0’s

15.5. API Key Creation (OpenAI)

Steps:

1. Search "OpenAI API key free" in Google.
2. Go to API Keys section → Generate new key.
3. Install library:
4. pip install openai
5. Import in Python:
6. from openai import OpenAI
7. Initialize client:
8. client = OpenAI()

15.6. Upcoming Tasks (as noted in your page)

- Work planned for Thursday & Friday → Milestone 2
    - Work includes using API key
- Build chatbot and Word document generator

- Push completed code to group repository
- Prepare PPT

15.7. Additional Notes

- Real-time data:
    - Need to link database with UI
    - Display real-time data
- Create API key (to be done by your monday)

15.8 Additional Notes

a)Python Chat Completion Code (OpenAI)

```
resp = client.chat.completions.create(
    model="gpt-4o-mini",
    messages=[
      {"role": "system", "content": "You are a helpful assistant."},
      {"role": "user", "content": "Write a two-line poem about chai."}
    ],
    max_tokens=120
)
```

b)Explanation of Key Terms

- model → the AI model you are using.
- messages → list of messages exchanged.
- system → defines the assistant's behavior.
- user → actual prompt/question from user.
- max_tokens → controls output length.
- temperature (0.1 to 1.0) → controls creativity of response.

c) Frontend to Backend Connection (JS Fetch API)

```
let response = await fetch("http://localhost:8000/chat", {
    method: "POST",
    headers: {"Content-Type": "application/json"},
    body: JSON.stringify({ message: message })
});
```

Note: Used inside a <script> tag to send user message to backend API.

16. Day16

16.1. Logistic Regression

- Used for classification problems (output is Yes/No, 0/1).
- Logistic Regression falls under classification algorithms.
- It uses a linear equation:

$$z = w_1x_1 + w_2x_2 + \cdots + b$$

- Applies sigmoid function to convert z into probability:

$$p = \sigma(z) = \frac{1}{1 + e^{-z}}$$

- Decision rule:
  - If $p > 0.5 \rightarrow$ class = 1
  - Else → class = 0
- Code:
- from sklearn.linear_model import LogisticRegression
- model = LogisticRegression()

16.2. Decision Tree

- Splits a big question into smaller chunks.
- Example:
    - Should I play?
        - Sunny or Rainy?
            - Rainy → Should not play
            - Sunny →
                - Hot → Should not play
                - Normal → Can play
- Uses concepts of:
    - Entropy

$$\text{Entropy} = -p\log\, p - q\log\, q$$

    - Information Gain (IG)

$$IG = \text{Entropy(parent)} - \text{Entropy(children)}$$

- Code:
- from sklearn.tree import DecisionTreeClassifier
- dt = DecisionTreeClassifier()

16.3. Random Forest

- Builds multiple decision trees, each seeing a random part of the data.
- Final output:
    - Majority vote → Classification
    - Average → Regression
- It is an ensemble model (uses many trees).
- Code:
- from sklearn.ensemble import RandomForestClassifier

rf = RandomForestClassifier()

16.5. Height Classification Example

- Given heights:
    - A = 170, B = 168 → Fit for sports
    - C = 150, D = 152 → Not fit for sports
- New heights to classify: 169, 151

(This is an example of applying ML classification based on height.)

16.6. K-Means Clustering

- Type: Unsupervised Learning
- Purpose: Group data into K clusters based on similarity

Steps:

1. Choose K centroids.
2. Assign each data point to the nearest centroid.
3. Recalculate centroids (mean of points in cluster).
4. Repeat until stable.

Objective Function:

$$\sum \mathrm{sqrt}(x_i - \text{centroid})^2$$

Code:

from sklearn.cluster import KMeans

kmeans = KMeans(n_clusters=3)

16.7. Linear Regression

- Predicts continuous values
  (e.g., sales, temperature, price).

- Equation:

$$y = mx + c$$

16.8. Mean Squared Error (MSE)

- Used as a loss function in regression.
- Formula:

$$MSE = \frac{1}{n}\sum(y - \hat{y})^2$$

16.9. Linear Regression

- Used for predicting continuous values.
- Code:

```
from sklearn.linear_model import LinearRegression

lr = LinearRegression()
```

16.12. XGBoost

- A powerful gradient boosting algorithm.
- It updates the model step-by-step:

$$F_{\text{new}}(x) = F_{\text{old}}(x) + \eta \cdot h(x)$$

Where:

- η (eta) = learning rate
- h(x) = a small decision tree (weak learner)

Key Points:

- Builds trees sequentially.
- Each new tree corrects the errors of the previous one.
- Adds regularization to reduce overfitting.

Code:

```
from xgboost import XGBClassifier

model = XGBClassifier()
```

17. Day 17

17.1 MySQL

SQL (Structured Query Language) is used to store, manage, and retrieve data from databases.
Basic syntax example:
SELECT * FROM table_name;

17.2 Common SQL Commands

- SELECT – retrieve data
- INSERT – add new data
- UPDATE – modify existing data
- DELETE – remove data
- CREATE – create tables/databases
- ALTER – modify table structure
- DROP – delete tables/databases
- RENAME – rename tables
- TRUNCATE – remove all rows from a table

17.3 WHERE Clause & Operators

Used to filter records.
Example: SELECT * FROM Customer WHERE country = 'Mexico';

17.4 Operators:

- Comparison: =, >, <, >=, <=, <> (or !=)

- Range: BETWEEN
- Pattern Matching: LIKE
- Set: IN, NOT IN
- Logical: AND, OR, NOT
- Sorting: ORDER BY

17.5 LIKE Pattern Matching

- a% → starts with a
- %a → ends with a
- %a% → contains a
- _a% → second letter is a
- a_% → starts with a, at least 2 characters
- a__% → starts with a, at least 3 characters
- a%o → starts with a and ends with o

17.6 Aggregate Functions

- MIN() – smallest value
- MAX() – largest value
- COUNT() – total number of rows
- AVG() – average value

17.7 Aliases

Used to rename columns or tables temporarily.
Example: SELECT name AS student_name FROM student;

17.8 Joins

Used to combine data from multiple tables.

17.9 Types of Joins:

- INNER JOIN – matching rows from both tables
- LEFT JOIN – all rows from left table + matched rows from right
- RIGHT JOIN – all rows from right table + matched rows from left
- FULL JOIN (if supported) – all rows from both tables
- CROSS JOIN – cartesian product

Example tables:
Table 1: (student) → std_id, std_name, address
Table 2: (subject) → sbj_id, stud_id, sbj_name

17.10 UNION & UNION ALL

- UNION – combines result sets, removes duplicates
- UNION ALL – combines result sets, keeps duplicates

17.11 GROUP BY & HAVING

- GROUP BY – groups rows based on a column
- HAVING – applies conditions on grouped data (used with aggregates)

18. DAY 18

Milestone -2 preparation

19. DAY 19

Milestone -2

20. DAY 20

Milestone -2

21. DAY 21