

# INTERNSHIP REPORT – AirAware Smart Air Quality Prediction System

## Day 1 – Project Introduction & Overview

### Topic: AirAware – Smart Air Quality Prediction System

AirAware is an AI-driven system designed to **monitor, analyze, and predict air quality** using **real-time pollution data**, machine learning algorithms, and visual analytics.

### Key Pollutants Considered

- PM2.5
- CO<sub>2</sub>
- NO<sub>2</sub>
- O<sub>2</sub> levels

### System Components

- **Frontend:** UI dashboard for displaying real-time and predicted air quality
- **Backend:** Python (Flask / FastAPI) for ML model inference and API handling
- **Database:** To store weather and air quality records
- **Dataset:** Kaggle / HuggingFace air quality datasets

### Objectives

- Collect 3–4 months of real pollution data
- Clean, normalize and preprocess the dataset
- Use ML models like **Linear Regression, Random Forest, SVM** for prediction
- Display **Predicted vs Actual** air quality
- Provide **alerts** when air quality becomes hazardous
- Build a complete awareness-based dashboard

1. **25%** – UI layout + Basic backend
  2. **50%** – Dataset preprocessing + initial ML model
  3. **75%** – Dashboard integration + API communication
  4. **100%** – Full project demo with predictions & reports
-

## **Day 2 – Team & Project Expansion**

### **Team Members**

- Rajalakshmi
- Rahul
- Sreya
- Lokesh
- Divija
- Nandana

### **Focus**

- Use **Delhi Air Quality Dataset (Kaggle)**
- Predict future air quality trends (next 4–5 years)
- Make a pleasant, component-based frontend (React recommended)

### **Dashboard Requirements**

- Last year's air quality data
- Current air quality
- Predictions
- Heatmap accuracy
- Weather status
- Different UI for each team

### **Technical Workflow**

- UI → Payload → Backend (Processing) → Response → UI
- Use API calls to send data between frontend and backend
- Data displayed dynamically from database

### **Future Scope**

- Real sensor integration
  - Enterprise-level prediction application
  - Aim for **95%-100% model accuracy**
-

## Day 3 – API Basics & Backend Concepts

### APIs Discussed

- **Flask API**
- **FastAPI**

### Code Basics

#### FastAPI

```
from fastapi import FastAPI
```

```
app = FastAPI()
```

```
@app.get("/")
```

#### Flask

```
from flask_api import FlaskAPI
```

```
from flask import request
```

```
app = FlaskAPI(__name__)
```

```
@app.route("/", methods=['GET'])
```

### Key API Concepts

- GET → payload in **header**, used to retrieve data
- POST → payload in **body**, used to send/receive data
- Payload = input sent from frontend
- Response = output returned by backend
- Postman is used for API testing

### Important Notes

- Backend must strictly follow **payload structure** sent by frontend
  - Change in payload → API failure (400/402 errors)
  - Flask = simple & synchronous
  - FastAPI = advanced with async, high performance
  - WebSocket vs HTTPS
-

## **Day 4 – Git & Version Control**

### **Git Commands Learned**

- `git add .` – Add all new files
- `git commit -m "msg"` – Create commit
- `git push` – Push to repo
- `git pull` – Pull latest updates
- `git fetch --all` – Get all branch changes
- `git branch`, `git checkout`, `git checkout -b`
- `git stash` – Save local changes temporarily

### **Virtual Environment**

- `python -m venv venv` – Create environment
  - `venv/Scripts/activate` – Activate
  - `pip install -r requirements.txt` – Install dependencies
  - `deactivate` – Exit environment
-

## **Day 5 – Database Concepts**

### **DB & DBMS**

- DB: Collection of data
- DBMS: Software to manage the database

### **CRUD Operations**

- Create
- Rename/Alter
- Update
- Delete

### **Types of Databases**

- **Structured:** MySQL, SQL, PostgreSQL
- **Unstructured:** MongoDB

### **Normalization**

- 1NF, 2NF, 3NF, BCNF, 4NF, 5NF
- Concepts: Partial & Transitive Dependency

### **Keys**

- Primary Key
- Foreign Key
- Composite Key

### **Pages Required in UI**

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

## AI Model Providers

- OpenAI, Google, Grok, Microsoft
- Models: GPT, Gemini, Llama, Copilot

## ML Basics

- **Supervised:** Regression & Classification
- **Unsupervised:** Clustering (K-Means)
- **Reinforcement:** Learning from mistakes

## Vectors & Embeddings

- Text → Vector (numerical form)
  - Used in PostgreSQL (pgAdmin)
  - NLTK for natural language preprocessing
-

## **Day 6 – Milestones & Project Flow**

### **Milestone Deadlines**

- Milestone 1 → **25%** → Thursday & Friday
- Milestone 2 → **50%** → Nov 27–28
- No UI changes after Nov 30
- Final output on Dec 1

### **Team Instructions**

- Keep backup copies
- Push both personal & group repos to GitHub
- Transparent UI
- Add teammates as contributors
- Prepare PPT and 45-page internship document

### **Final Document Requirement**

- 45-page project report
  - Each day as a separate topic
  - Includes:
    - Git
    - API
    - Database
    - ML
    - Project architecture
    - Daily progress
-