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_2
- NO_2
- O_2 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**4 Milestones**

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**

<u>Day 3 – API Basics & Backend Concepts</u>

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

Day 7 – Doubt Clearence

Doubt Clarifications:

- Instructor addressed queries on the Aware project, including backend-frontend workflow.
- Questions on APS Integration, dataset usage, and Git practices were clarified.
- Guidance provided on Python APIs, specifically Flask, and database selection.
- Structuring communication between frontend and backend was explained.

Project Instructions:

- Steps for approaching Milestone-1 were discussed.
- Guidelines for organizing the GitHub repository were provided.
- Teams were instructed to maintain transparency in UI development and backend logic.

Best Practices & Reminders:

- Follow the planned structure of the project.
- Avoid unnecessary UI template changes after deadlines.
- Ensure project progress aligns with the 25% milestone target.

Day 8 – Document Presentation

- Allocated time to prepare all required documentation for **Milestone 1**.
- Identified and finalized the **project objectives**.
- Completed the **functional and non-functional requirements** for the project.
- Structured the documentation in an organized and clear format.
- Ensured the document aligns with the **Milestone 1 guidelines** provided.
- Reviewed foundational project components for clarity and completeness.

Day 9 – PPT & UI Review Session

- The day focused on reviewing Milestone-1 submissions from all teams.
- The instructor evaluated the PPT presentations and basic UI layouts prepared by students.
- Feedback was given on slide clarity, presentation flow, and design quality.
- UI screens were checked for completeness, navigation, and alignment with project requirements.
- Marks for Milestone-1 were allotted based on presentation quality, UI readiness, and adherence to instructions.

Day 10 – Introduction to Artificial Intelligence (AI)

AI Overview
- The day focused on understanding the basics of Artificial Intelligence and how machines learn, analyze patterns, and make decisions similar to humans.
- AI was introduced as a collection of techniques that enable systems to perceive data, learn, and perform tasks autonomously.

Subfields of AI
- Machine Learning
- Deep Learning
- Natural Language Processing
- Reinforcement Learning
- Knowledge-Based Systems

Machine Learning Basics
- ML was explained as algorithms that learn from data and improve automatically.
- Types of learning covered: Supervised, Unsupervised, and Reinforcement Learning.

Deep Learning Concepts
- Deep Learning uses multi-layered neural networks to learn complex patterns.
- Common architectures: ANN, CNN, RNN, LSTM, Transformers.

LSTM Overview
- LSTM networks were explained as models designed for sequential or time-series data.
- Key components include forget, input, and output gates.
- Applications: stock prediction, weather forecasting, and time-series analysis.

Other AI Domains
- NLP: Enables systems to understand human language.
- Reinforcement Learning: Learns using rewards and penalties.
- Knowledge-Based Systems: Uses predefined rules for decision-making.

AI/ML Project Workflow
- Data collection
- Data cleaning and feature engineering
- Model training using proper data split ratios
- Model evaluation with performance metrics

Day 11 – Natural Language Processing (NLP)

Introduction to NLP

- The session covered the basics of Natural Language Processing, a branch of AI that enables machines to understand and work with human language.
- NLP helps systems interpret grammar, context, meaning, emotion, and ambiguous expressions.

NLTK Overview

- NLTK was introduced as a beginner-friendly Python library used for text processing and core NLP operations.

Core NLP Tasks

- Tokenization: Splitting text into individual words or pieces.
- Stop-word Removal: Removing common words that add little meaning.
- Stemming: Reducing words to their root form by trimming endings.
- Lemmatization: Converting words to their proper dictionary base form.
- POS Tagging: Assigning grammatical roles like noun, verb, adjective.
- NER: Identifying key entities such as names, dates, places, and organizations.

Text Preprocessing

- Steps such as converting text to lowercase, removing punctuation/numbers, cleaning spaces, splitting into words, removing stop-words, and applying stemming or lemmatization were discussed as essential preparation before training NLP models.

SVM in NLP

- Support Vector Machine was explained as a classifier used for tasks like text categorization, spam filtering, and sentiment analysis by finding optimal boundaries between classes.

TF-IDF Concepts

- TF-IDF was introduced as a technique to identify important words in documents.
- Frequent but less informative words get low weight, while rare but meaningful terms get high weight.
- Applications include search engines, keyword extraction, and document ranking.

Day 12 – Stemming, SVM & Reinforcement Learning

Stemming

• The session began with a brief explanation of stemming.
• Stemming reduces words to a simple root form by cutting off endings, though the resulting word may not always be meaningful.
• It is mainly used for quick text normalization in NLP tasks.

Support Vector Machine (SVM)

• SVM was introduced as a supervised learning algorithm mainly used for classification, with additional applications in regression and anomaly detection.
• The core idea is to separate data into classes using the best possible boundary called a **hyperplane**, which maximizes the margin between classes.
• The nearest data points to this margin are known as **support vectors**, and they define the position of the hyperplane.

Types of Margins

• **Hard Margin:** Assumes perfect data separation; suitable only for noise-free datasets.
• **Soft Margin:** Allows minor misclassification and can adapt to non-linear boundaries; preferred for real-world problems.

Reinforcement Learning (RL)

• Reinforcement Learning was explained as a trial-and-error-based learning method where an agent improves by receiving rewards or penalties for its actions.
• An **AI agent** interacts with an environment, makes decisions, and refines its behavior based on feedback.

Agent Architectures

• **Single-Agent System:** One agent handles the entire workflow.
• **Multi-Agent System:** Multiple agents collaborate, each performing specialized tasks.

Human Interaction Modes

• **Human-in-the-loop:** The agent seeks human approval for major decisions to maintain safety and control.
• **Fully Autonomous:** The agent operates independently, though modern systems still prefer human supervision for reliability.

Autogen Framework

• Autogen was introduced as a modern Python framework designed for creating automated AI agents.
• Autogen AI Studio provides tools for building, testing, and managing agent workflows.

Day 13 – HTML, CSS & Layout Concepts

CSS Sizing Concepts
• The session introduced intrinsic and extrinsic sizing.
• Intrinsic size refers to an element's natural size, while extrinsic size is defined through CSS properties like width and height.

Overflow Handling
• Overflow behavior was discussed for cases where content exceeds the container.
• Options include: visible, hidden, scroll, and auto.
• Overflow can also be controlled separately using overflow-x and overflow-y.

Min/Max Dimensions
• CSS provides min-width, min-height, max-width, and max-height to control the smallest and largest limits for elements.

Meta Tags
• Meta elements provide additional webpage information such as character encoding, viewport settings, keywords, and description.

Box Sizing
• Two modes were explained:
  content-box: Default; padding and border are added outside width/height.
  border-box: Width includes content, padding, and border, making layout simpler.

Page Layout Techniques
• Two major layout systems were covered:
  Flexbox for one-dimensional layouts
  CSS Grid for two-dimensional layouts

Key Flexbox Properties
• display: flex
• flex-direction for setting row/column
• justify-content for main-axis alignment
• align-items for cross-axis alignment

HTML Basics
• HTML was described as the structure of a webpage.
• Common tags: headings (h1–h6), p, div, img, video, audio.
• Structural elements include header, nav, main, and footer.

CSS Overview
• CSS is used for styling – colors, fonts, layout, spacing, and alignment.

Flexbox vs Grid
• Flexbox: One-dimensional layout
• Grid: Two-dimensional layout supporting rows and columns

Day 14 – API Basics (OpenAI / Gemini)

Flexbox Basics

- flex-wrap: nowrap (default), wrap, wrap-reverse
- align-self: flex-start, center, flex-end, stretch, auto
- order: 0 is default, positive → later, negative → earlier
- Flex container → direct children become flex items

AI Model Basics (OpenAI/Gemini)

- APIs allow communication with AI models using requests
- Steps: install library → import → add API key → send prompt → get response
- Model parameters:
    - system message
    - user message
    - max_tokens controls output length

Day 15 – Machine Learning Algorithms

1. Logistic Regression

- Used for binary classification (yes/no, spam/not spam).
- Converts input values into a probability between 0 and 1.
- If probability is high → class 1; otherwise → class 0.

2. Decision Tree

- Works like a flowchart with questions and decisions.
- Splits data based on the best conditions.
- Uses “purity” of data to decide how to split.
- Easy to visualize and understand.

3. Random Forest

- Collection of many decision trees.
- Each tree gives an answer → final answer is the majority vote.
- More stable and accurate than a single tree.

4. K-Nearest Neighbors (KNN)

- Looks at the closest data points around the input.
- Classifies based on what most neighbors belong to.
- “Similar things stay close.”

5. K-Means Clustering

- Unsupervised algorithm (no labels).
- Groups data into K clusters based on similarity.
- Repeatedly adjusts group centers until stable.

6. Linear Regression

- Predicts continuous numeric values (price, sales, marks).
- Finds the best straight line that fits the data.

7. XGBoost

- Advanced boosting algorithm.
- Builds trees one after another — each new tree fixes previous mistakes.
- Very powerful and used in competitions.

Day 16 – MySQL

What is SQL?

SQL (Structured Query Language) is used to store, access, and manage data in a database.

Common SQL Commands

- SELECT → Read data
- INSERT → Add new data
- UPDATE → Modify existing data
- DELETE → Remove data
- CREATE → Create database/table
- ALTER → Modify structure
- DROP → Delete table/database
- TRUNCATE → Remove all rows (faster than DELETE)
- RENAME → Rename table

Selecting Data

- SELECT * FROM table; → Show whole table
- SELECT col1, col2 FROM table; → Show specific columns
- SELECT DISTINCT col FROM table; → Remove duplicates
- WHERE → Filter rows

Operators

- =, >, <, >=, <=, != or <>
- BETWEEN (range)
- LIKE (pattern match)
- IN (matches multiple values)
- AND, OR, NOT

LIKE Patterns

- a% → starts with a
- %a → ends with a
- %a% → contains a
- _a% → second letter a
- a__% → starts with a and has at least 3 letters

Aggregate Functions

- SUM, MIN, MAX, COUNT
- Used with GROUP BY.

JOINS

Used to combine data from multiple tables.

1. INNER JOIN

Returns matching rows from both tables.

2. LEFT JOIN

Returns all rows from left table, matching rows from right.

3. RIGHT JOIN

Returns all rows from right table, matching rows from left.

4. CROSS JOIN

Returns every combination (cartesian product).

UNION vs UNION ALL

- UNION → Removes duplicates
- UNION ALL → Keeps duplicates

HAVING

Like WHERE, but used for aggregate results (after GROUP BY).

Day 17 – Preparation for Milestone 2

- Prepared document, PPT, and project code for Milestone 2.
- Document included explanation of classes and project structure.
- PPT covered project flow and features.
- Implemented OpenAI API for chatbot functionality.
- Any suitable AI model could be used for the implementation.

Day 18 – Review and Feedback Session

- Instructor reviewed each student's document, PPT, and project code for Milestone 2.
- Checked clarity, completeness, formatting, and correctness.
- Evaluated implementation of chatbot functionality.
- Provided suggestions, corrections, and recommendations to improve project quality.
