**SOURCE CODE**

1. **index.html (input concentration of different pollutants)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>Air Quality Index</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

margin: 0;

padding: 0;

background: linear-gradient(to right, #3a7bd5, #00d2ff);

color: white;

height: 100vh;

display: flex;

flex-direction: column;

align-items: center;

justify-content: start;

padding-top: 60px;

}

.navbar {

position: fixed;

top: 0;

width: 100%;

background: rgba(0, 0, 0, 0.3);

padding: 15px 0;

text-align: center;

font-size: 1.5rem;

font-weight: bold;

box-shadow: 0 2px 6px rgba(0,0,0,0.2);

}

.datetime-box {

position: absolute;

top: 45px;

right: 50px;

padding-left: 20px;

padding-right: 20px;

padding-top: 15px;

padding-bottom: 15px;

text-align: right;

font-size: 17px;

font-family: Arial;

color: rgb(40, 37, 37);

font-weight: bold;

background-color:rgb(255, 252, 252);

border-radius: 10px;

}

.container {

background: rgba(0, 0, 0, 0.4);

padding: 40px;

border-radius: 15px;

box-shadow: 0 0 20px rgba(255, 255, 255, 0.2);

max-width: 400px;

width: 90%;

margin-top: 100px;

text-align: center;

}

.container h2 {

margin-bottom: 25px;

}

input[type="number"] {

width: 100%;

padding: 12px 15px;

margin: 10px 0;

border: none;

border-radius: 8px;

box-sizing: border-box;

font-size: 1rem;

}

button {

background-color: #fff;

color: #007acc;

font-weight: bold;

padding: 12px 20px;

border: none;

border-radius: 8px;

cursor: pointer;

font-size: 1rem;

transition: all 0.3s ease;

margin-top: 10px;

}

button:hover {

background-color: #007acc;

color: white;

transform: scale(1.05);

}

@media (max-width: 600px) {

.container {

padding: 25px;

}

.navbar {

font-size: 1.2rem;

}

}

</style>

</head>

<body>

<!-- Live Date/Time Box -->

<div class="datetime-box">

<div class="date" id="current-date">Loading date...</div>

<div class="time" id="current-time">Loading time...</div>

</div>

<div class="navbar">

<h1>Air Quality Index</h1>

</div>

<div class="container">

<h2>Check Air Quality</h2>

<form method="POST" id="aqi-form" action="/result.html">

<input type="number" name="PM25" placeholder="PM2.5 (µg/m³)" required>

<input type="number" name="PM10" placeholder="PM10 (µg/m³)" required>

<input type="number" name="NO2" placeholder="NO₂ (µg/m³)" required>

<input type="number" name="SO2" placeholder="SO₂ (µg/m³)" required>

<input type="number" name="CO" placeholder="CO (mg/m³)" required>

<input type="number" name="O3" placeholder="O₃ (µg/m³)" required>

<button type="submit">Submit</button>

</form>

</div>

<script>

function updateDateTime() {

const now = new Date();

const days = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday"];

const months = ["January", "February", "March", "April", "May", "June", "July",

"August", "September", "October", "November", "December"];

const day = days[now.getDay()];

const month = months[now.getMonth()];

const date = now.getDate();

const year = now.getFullYear();

let hours = now.getHours();

let minutes = now.getMinutes();

let seconds = now.getSeconds();

const ampm = hours >= 12 ? 'PM' : 'AM';

hours = hours % 12 || 12;

minutes = minutes < 10 ? '0' + minutes : minutes;

seconds = seconds < 10 ? '0' + seconds : seconds;

document.getElementById('current-date').textContent = `${day}, ${month} ${date}, ${year}`;

document.getElementById('current-time').textContent = `${hours}:${minutes}:${seconds} ${ampm}`;

}

setInterval(updateDateTime, 1000);

updateDateTime();

document.getElementById("aqi-form").addEventListener("submit", async function(event) {

event.preventDefault();

const formData = new FormData(event.target);

const params = new URLSearchParams(formData);

try {

const response = await fetch(`/predict?${params.toString()}`);

const data = await response.json();

if (data.aqi !== undefined) {

window.location.href = `/result?aqi=${data.aqi}`;

} else {

alert("Error calculating AQI.");

}

} catch (err) {

alert("Failed to connect to server.");

}

});

</script>

</body>

</html>

1. **result.html (shows calculated AQI)**

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0"/>

<title>AQI Result</title>

<style>

body {

font-family: 'Segoe UI', Arial, sans-serif;

margin: 0;

padding: 0;

background: linear-gradient(to right, #3a7bd5, #00d2ff);

color: white;

display: flex;

flex-direction: column;

align-items: center;

justify-content: center;

height: 100vh;

text-align: center;

}

.result-box {

background: rgba(0, 0, 0, 0.4);

padding: 30px 40px;

border-radius: 15px;

box-shadow: 0 0 20px rgba(255,255,255,0.2);

max-width: 400px;

}

h1 {

font-size: 2rem;

margin-bottom: 15px;

}

#aqi-result {

font-size: 2rem;

font-weight: bold;

margin-bottom: 10px;

}

#advisory {

font-size: 1.1rem;

margin-top: 10px;

padding: 10px;

border-radius: 10px;

}

.good { background-color: #00e40044; color: #00e400; }

.moderate { background-color: #ffff0044; color: #ff0; }

.usg { background-color: #ff7e0044; color: #ff7e00; }

.unhealthy { background-color: #ff000044; color: #ff0000; }

.very-unhealthy { background-color: #8f3f9744; color: #8f3f97; }

.hazardous { background-color: #7e002344; color: #7e0023; }

.back-link {

display: inline-block;

margin-top: 20px;

padding: 10px 20px;

border-radius: 8px;

background-color: #ffffff22;

color: white;

text-decoration: none;

font-weight: bold;

border: 1px solid white;

transition: background-color 0.3s ease;

}

.back-link:hover {

background-color: #ffffff44;

}

</style>

</head>

<body>

<div class="result-box">

<h1>Your Calculated AQI</h1>

<p id="aqi-result">Calculating...</p>

<div id="advisory" class="">Loading advisory...</div>

<a class="back-link" href="/">← Back to form</a>

</div>

<script>

const params = new URLSearchParams(window.location.search);

const aqi = parseInt(params.get("aqi"));

const resultElem = document.getElementById("aqi-result");

const advisoryElem = document.getElementById("advisory");

if (!isNaN(aqi)) {

resultElem.textContent = `AQI : ${aqi}`;

let message = "";

let category = "";

if (aqi <= 50) {

message = "Air quality is considered satisfactory. It's a great day to be outside!";

category = "good";

} else if (aqi <= 100) {

message = "Air quality is acceptable. Some pollutants may be a concern for sensitive individuals.";

category = "moderate";

} else if (aqi <= 150) {

message = "Unhealthy for Sensitive Groups. Children, elderly, and those with respiratory issues should reduce outdoor exertion.";

category = "usg";

} else if (aqi <= 200) {

message = "Unhealthy. Everyone may begin to experience health effects; sensitive groups should avoid outdoor activities.";

category = "unhealthy";

} else if (aqi <= 300) {

message = "Very Unhealthy. Health alert: everyone may experience more serious health effects.";

category = "very-unhealthy";

} else {

message = "Hazardous. Serious health effects. Avoid outdoor activity and stay indoors.";

category = "hazardous";

}

advisoryElem.textContent = message;

advisoryElem.classList.add(category);

} else {

resultElem.textContent = "Unable to calculate AQI";

advisoryElem.textContent = "";

}

</script>

</body>

</html>

1. **app.py( flask API code to connect frontend with backend )**

from flask import Flask, request, render\_template, jsonify

from flask\_cors import CORS

import numpy as np

import joblib

app = Flask(\_\_name\_\_)

CORS(app)

# Load model and scaler at startup

model = joblib.load("models/ran\_model.pkl") # Your trained ML model

scaler = joblib.load("models/ran\_scaler.pkl") # Your fitted scaler

@app.route('/')

def home():

return render\_template("index.html")

@app.route('/result')

def result():

return render\_template("result.html")

def get\_health\_advisory(aqi):

"""Return AQI category and health advisory based on AQI value."""

if aqi <= 50:

return ("Good", "Air quality is satisfactory. It's a great day to be outside!")

elif aqi <= 100:

return ("Moderate", "Air quality is acceptable. Some pollutants may be a concern for sensitive individuals.")

elif aqi <= 150:

return ("Unhealthy for Sensitive Groups", "Children, elderly, and people with respiratory problems should limit outdoor activity.")

elif aqi <= 200:

return ("Unhealthy", "Everyone may begin to experience health effects. Sensitive groups should avoid prolonged outdoor exertion.")

elif aqi <= 300:

return ("Very Unhealthy", "Health alert: everyone may experience more serious health effects. Avoid outdoor activity.")

else:

return ("Hazardous", "Serious health effects. Stay indoors and avoid all physical activity outside.")

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

def predict():

try:

# Extract query parameters

features = [

float(request.args.get("PM25", 0)),

float(request.args.get("PM10", 0)),

float(request.args.get("NO2", 0)),

float(request.args.get("SO2", 0)),

float(request.args.get("CO", 0)),

float(request.args.get("O3", 0))

]

# Scale and predict

input\_array = np.array(features).reshape(1, -1)

scaled\_input = scaler.transform(input\_array)

prediction = model.predict(scaled\_input)

aqi = round(prediction[0])

# Get health advisory

category, advisory = get\_health\_advisory(aqi)

return jsonify({

"aqi": aqi,

"category": category,

"advisory": advisory

})

except Exception as e:

return jsonify({"error": str(e)}), 400

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

1. **db\_conn.py (loading dataset into mongoDB)**

import pandas as pd

from pymongo import MongoClient

import json

# Load the file

df = pd.read\_csv("PRSA\_Data\_Aotizhongxin\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Changping\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Dingling\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Dongsi\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Guanyuan\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Gucheng\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Huairou\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Nongzhanguan\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Shunyi\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Tiantan\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Wanliu\_20130301-20170228")

df = pd.read\_csv("PRSA\_Data\_Wanshouxigong\_20130301-20170228")

df.dropna(inplace=True)

# Create 'time' column

df['time'] = pd.to\_datetime(df[['year', 'month', 'day', 'hour']])

# Drop split time columns, but keep station name

df = df.drop(columns=['No', 'year', 'month', 'day', 'hour'])

# Connect to MongoDB

client = MongoClient("mongodb://localhost:27017/")

db = client["AQI"]

collection = db["china\_data"]

#collection.delete\_many({}) # Clear previous runs

# Insert into MongoDB

#data\_json = json.loads(df.to\_json(orient='records'))

#collection.insert\_many(data\_json)

#print(f"Inserted {len(data\_json)} records into MongoDB.")