POLITECNICO MILANO 1863

Software Engineering for Geoinformatics 2023

Design Document

EarthTech Group DD Presentation by

Mohammad Kordgholiabad

Fatemeh Soleimaniansomarin

Ghulam Abbas Zafari

EarthTech

Air Quality Monotoring

Deliverable: DD

Title: Design Document

Authors: Mohammad Kordgholiabad, Fatemeh Soliemaniansomarin
Ghulam Abbas Zafari

Version: 1.0
Date: May 25, 2023

Revision history

Version	Date	Change
1.0	25/05/2023	First submitted version

Table of Contents

1 Introduction 4
1.1 Introduction 4
1.2 Purpose 4
1.3 Scope 4
1.4 Short overview 4
1.5 Target Audience 5
1.6 Anacronyms and other definitions 5
2 Architectural Design 6
2.1 Overview 6
2.2 Component Diagram 6
3 Software Design 7
3.1 Database Design 7
3.1.1 Specification of tables in Database 8
3.1.2 Entity Relationship 8
3.1.3 ER Diagram 9
3.2 User interface Design 9
3.2.1 Overview 9
3.2.2 Design 10
3.2.3 Content Tables 11
4 Implementation Plan 12
4.1 Implementation Approach 12
4.2 Development Tasks 12
4.3 Resource Allocation 12
4.4 Deployment Strategy 13
5 Testing Plan 13
5.1 Testing Approach 13
5.2 Test Types 13
5.3 Test Cases 14
5.4 **Test Environment** 14
5.5 Test Execution and Reporting 15
6 Bibliography 16

1 Introduction

1.1 Introduction

An important step in software development is the design of software architecture and testing of the functionality of implemented software. This document will provide a detailed overview of the design aspects of our interactive application, including the system architecture, database, user interface design, and more. Its purpose is to guide our development team in building the application effectively and ensure that we meet all the functional and non-functional requirements.

1.2 Purpose

In the Lombardy region, public authorities collect air quality and weather observations from ground sensor stations on a continuous basis. These observations result in a comprehensive dataset consisting of pollution measurements from Benzene, nitrogen dioxide, sulphur dioxide, carbon monoxide, ozone, and nitrogen oxides, recorded at 128 stations throughout the entire year of 2022. This extensive dataset offers detailed information about air quality conditions across various locations in Lombardy, enabling us to perform in-depth analysis and generate meaningful visualizations.

1.3 Scope

The scope of our project is to design and develop an interactive Jupyter dashboard that retrieves air quality data from our local PostgreSQL database. The dashboard will provide users with seamless access to the collected dataset, allowing them to query and visualize air quality conditions across different regions and time periods within Lombardy. By incorporating advanced data processing techniques, intuitive visualizations, and interactive features, our application aims to empower public authorities and citizens to gain valuable insights into air pollution trends, identify high-risk areas, and make informed decisions to improve overall air quality and environmental health.

1.4 Short overview

Air pollution is a significant global challenge that requires collective efforts from individuals, communities, governments, and industries to address. By taking steps to reduce our contribution to air pollution and advocating for policies and practices that promote cleaner air, we can help protect the environment and safeguard our health for generations to come.
In a time of information saturation and global climate crisis, there is a need for distributed accessible high-quality information. This combined with the increasing propensity for the individual to find their own specific information sources tailored to their own needs means that tools are needed which allow individuals to cross-reference the information they receive using reliable and objective sources.
The application to be developed, therefore, aims to be a tool that provides easy access to air quality data in order to empower them both in their personal lives and as

members of a community. The application is not a source in and of itself, but rather a visualization tool for existing and established data sources.

1.5 Target Audience

The primary audience for our application includes public authorities responsible for air quality monitoring and management in the Lombardy region. They will utilize the dashboard to analyze pollution trends, identify potential sources of pollution, and implement targeted interventions and policies to address air quality issues effectively. Additionally, the dashboard will be accessible to ordinary citizens, enabling them to stay informed about air quality conditions in their local areas. This will empower citizens to make informed choices regarding outdoor activities, health precautions, and advocate for environmental initiatives.

1.6 Anacronyms and other definitions

“The web app” or “app” : The application that is being developed.

“Use case” : An example of user interaction with the system.

“User” : any individual using the website that satisfies the description under the “ user characteristics” section.

“Bookmark” : Saving a snapshot of the information presented at a location on the map.

JSON : JavaScript Object Notation.

JS : JavaScript.

df : Data frame, a pandas data frame object.

gdf : geodata frame, a geopandas data frame object

RASD : Requirements Analysis and Specification Document, easy later access.

DD : Design Document

“API” : Application Programming Interface

"REST" : Representational State Transfer

“Python” : High-level programming language we are going to use for building our software. It will be the basis of every functional tool used in our web application.

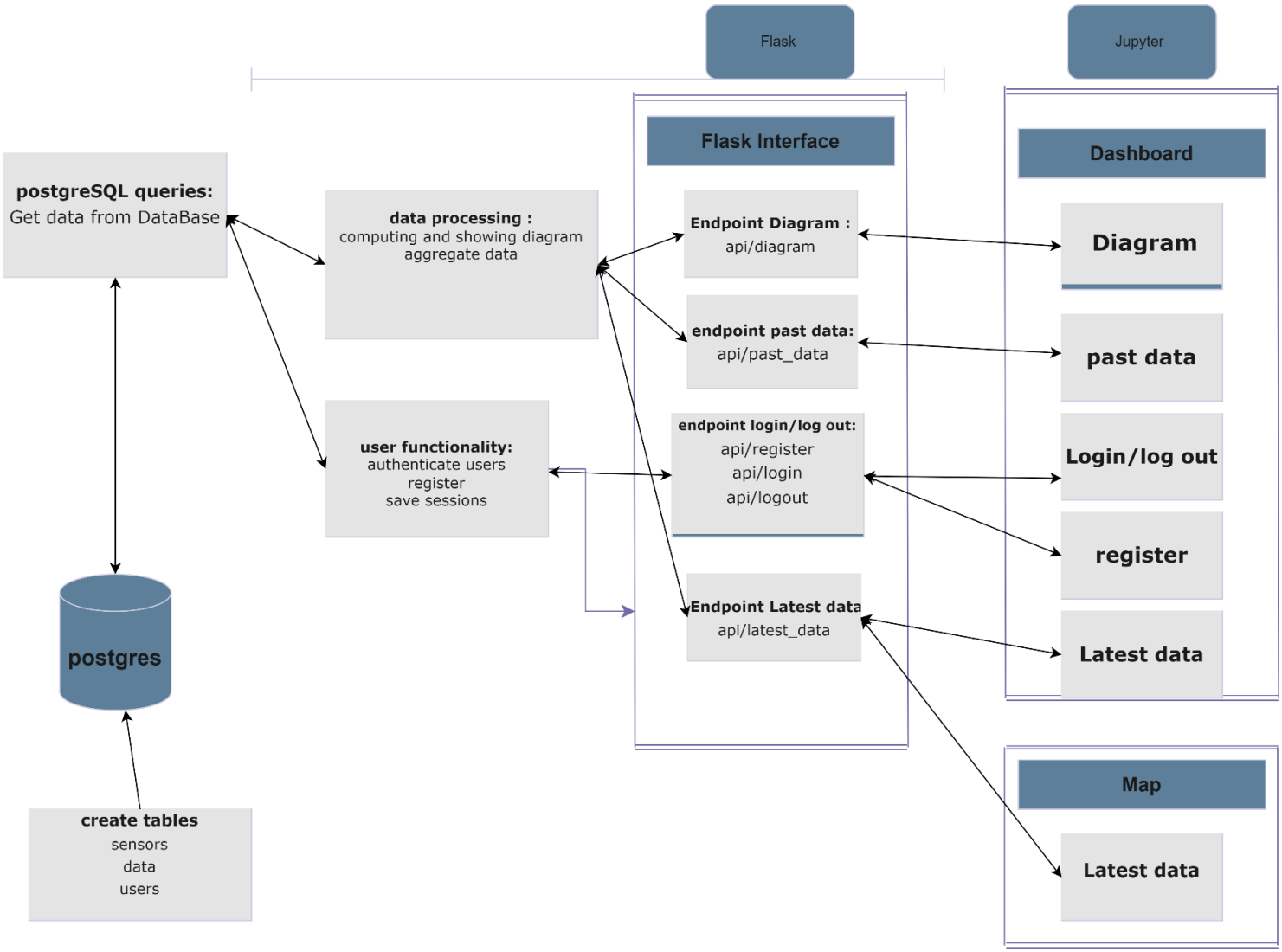
"AQI" : Air Quality Index.

2 Architectural Design

2.1 Overview

The architectural design of our project, an interactive application for air quality monitoring, focuses on establishing a robust and scalable system that supports users in accessing, querying, and visualizing air quality and weather sensor data. The system comprises three main components: a database for data storage, a web server (backend) exposing a REST API for data retrieval, and a Jupyter Notebook-based dashboard for data processing and visualization.

2.2 Component Diagram



3 Software Design

3.1 Database Design

The application relies on a local PostgreSQL database to store and manage the necessary data. The database consists of three tables: Users, Sensors, and Data. The Users table stores information about the application's users, while the Sensors table contains details about the various sensors deployed in the region. The Data table holds the actual pollution data collected from these sensors. This database forms the backbone of our interactive air quality monitoring application, facilitating data storage, retrieval, and analysis.

1. Users Table:
   Columns:
   - User ID: A unique identifier for each user in the system.
   - UserName: The name associated with the user.
   - User Password: The password used for user authentication.

The Users table maintains records of all registered users of the application. It stores their unique identifiers, associated names, and securely hashed passwords. This table enables user authentication and access control within the system.

2. Sensors Table:
   Columns:
   - Sensor ID: A unique identifier for each sensor.
   - Sensor Type: The type or category of the sensor (Benzene, nitrogen dioxide, sulfur dioxide, carbon monoxide, ozone, nitrogen oxides).
   - Sensor Unit: The unit of measurement for the sensor readings.
   - Station ID: The unique identifier of the station where the sensor is deployed.
   - Station Name: The name of the station.
   - Province: The province where the sensor is located.
   - City: The city where the sensor is located.
   - Latitude (Lat): The geographic latitude coordinate of the sensor.
   - Longitude (Long): The geographic longitude coordinate of the sensor.

The Sensors table contains detailed information about the deployed sensors and their corresponding stations. It records unique sensor IDs, sensor types, units of measurement, station IDs, station names, location details (province, city, latitude, and longitude), and other relevant metadata. This table enables efficient organization and retrieval of sensor data, facilitating data analysis and visualization in the application.

3. Data Table:
   Columns:
   - Sensor ID: The identifier of the sensor that collected the data.

- Data: The specific type of data collected by the sensor.
- Value: The measured value of the data.

The Data table stores the actual pollution data collected from the sensors. It includes the sensor ID, the specific type of data collected (e.g., Benzene, nitrogen dioxide), and the corresponding measured value. This table forms the core of the application's data storage and enables efficient querying and retrieval of air quality data for further processing and visualization.

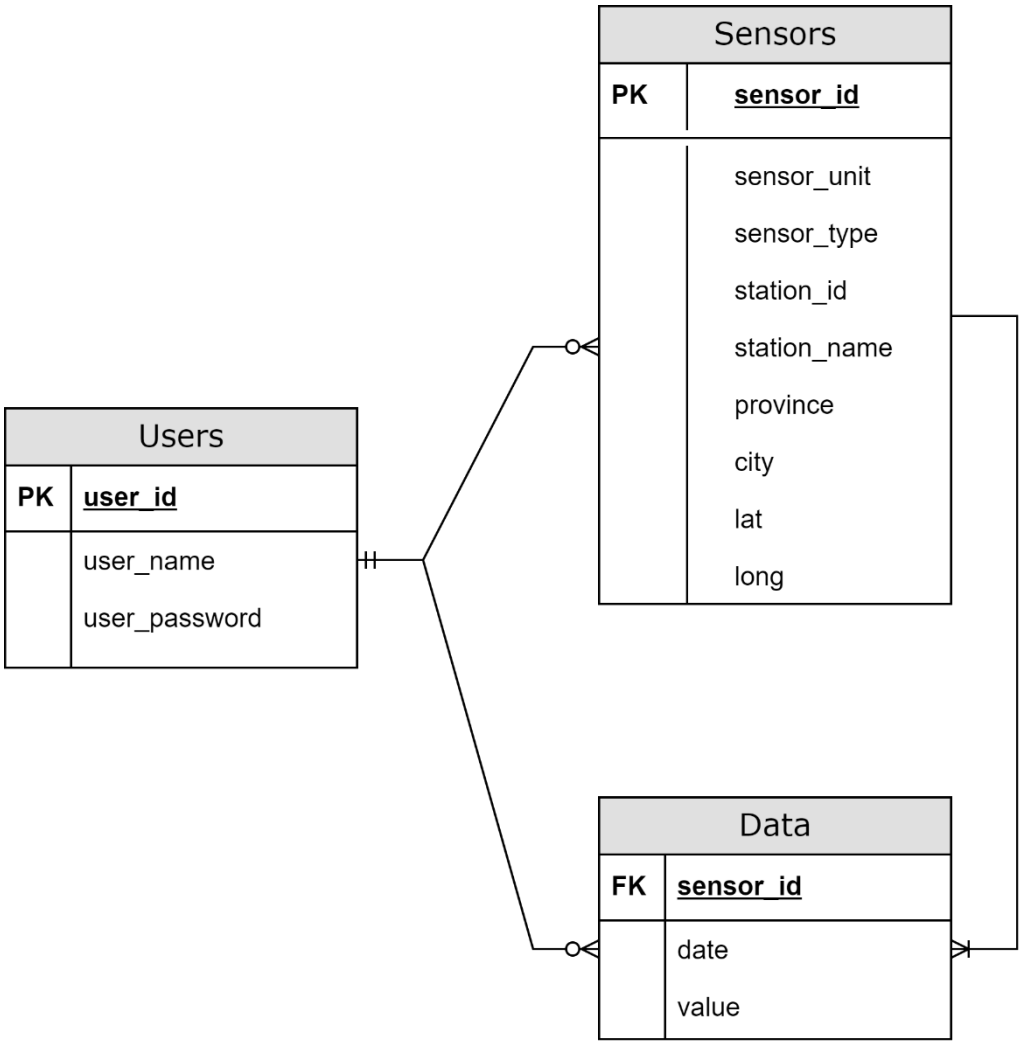
3.1.1 Specification of tables in Database

Table	Usage	Columns
Users	Keep track of all registered users	user_id (autogenerated, identity) user_name (unique and not null) user_password (not null)
Sensors	Stored details about sensors and stations	sensor_id (identity) sensor_type sensor_unit station_id station_name province city lat long
Data	The main stored data by sensors	sensor_id (identity) date value

3.1.2 Entity Relationship

The Entity Relationship (ER) model is used to depict the relationships between entities in a database system. It helps define the structure and organization of data. In our project, we will utilize the ER model to design the database schema for air quality and weather sensor data. The ER diagram will showcase entities, attributes, and relationships, providing a concise overview of the system's data architecture.

3.1.3 ER Diagram



3.2 User interface Design

3.2.1 Overview

User interface for an air quality dashboard refers to the design and layout of the application that allows users to interact with and access information related to air quality. It includes various visual and interactive elements that enable users to navigate, view, and analyse air quality data effectively.
user interface design focuses on creating an aesthetically pleasing, user-friendly, and intuitive interface that allows users to easily access, visualize, and interact with air quality information. It prioritizes usability, responsiveness, customization options, and clear data representation to enhance the user experience.

3.2.2 Design

- Dashboard:

The dashboard is the main screen of the application and provides an overview of the current air quality conditions. It typically includes data such as pollutant levels.
The dashboard usually presents the current amount of pollutants, which is a numerical value or color-coded indicator that represents the overall air quality at a particular station in Lombardy region.

- Maps and Location Selection (the sensors location):

The interface features an interactive map that allows users to select their desired location or view air quality data for specific station in the area. Our users can either search for a location or use the necessary information about pollution.

- Data Visualization:

Air quality data is typically presented using visualizations to make it easier for users to understand. Graphs, charts, and color-coded maps can be utilized to represent pollutant levels and historical trends. Users can customize the time range and pollutants they want to view.

- Pollutant Information:

Users should be able to access comprehensive information about different pollutants, such as Benzene, Ozone (O3), Sulfur dioxide (SO2), Nitrogen dioxide (NO2), Carbon monoxide (CO) and Nitrogen oxides (NOx).

- Search and Filtering:

The user interface provides search functionality, allowing users to search for specific locations, dates, or pollutant types. Filtering options can help users narrow down the displayed data based on various parameters, such as pollutant level thresholds or specific time ranges.

- Export:

The "Export" interface typically provides users with options to download or export air quality data and related information.
It allows users to select the specific air quality data they want to export. They can choose parameters such as pollutant type, location, time range, or any other relevant criteria.
Users can specify the desired format for the exported data. Common options include CSV (Comma-Separated Values) or JSON (JavaScript Object Notation).

3.2.3 Content Tables

Content for map view

Main Headline	Title
Register as a user	“Sign Up" button is used and understood, indicating that the button is used to create a new user account and register for access to the web application. After this procedure the user will be registered as a new user in the system and is redirected to the Log In page.
Login as a user	Login" button. These labels are widely understood and intuitive for users, indicating that the button is used to authenticate and access the dashboard with their user credentials and the system automatically redirect the user to the homepage.
Side bar	Landmarks leading to other pages of the web app
Map	Interactive map with dynamic markers
Searching	Exit condition: User sees the correct data with wanted location. User: registered users go to the searching bar and chooses the option search by name or search by location. User inserts the name of the city or location (Lon and Lat). The application receives the data and shows the data in a table. It also shows whether the air is healthy by color. The user can see the result in a table.
Comparing data	Exit condition: user sees the diagram with desired information. User: registered users. User clicks the “more options” button and chooses “compare”. Application sends data request to API. Application receives the requested data from API. Application does the needed process to draw the diagram of both locations on the same char, with different color. User sees the diagram and the comparison on a chart.

Content for Dashboard view

Main Headline	Title
Pop-ups on the map	Are interactive windows or information boxes that appear when a user interacts with a specific point or area on the map. They provide additional details, data, or actions related to the selected location. User moves and keeps the mouse cursor on a location. on the map. Application collects the position of cursor and sends data request of related location to API.
Side bar	Landmarks leading to other pages of the web app
Interactive charts	Data Exploration of Interactive charts allow users to explore the underlying data by zooming in, focus on panning, or selecting specific data points of more interest. Users can interact with the chart to specific regions or time periods, providing a detailed view.
Parameter toggler:	It allows the user to switch or toggle between different parameters or options within an application or interface.

4 Implementation Plan

The implementation phase of the project involves the actual development and deployment of the interactive application for air quality monitoring. The following sections outline the implementation approach, development tasks, resource allocation, and deployment strategy.

4.1 Implementation Approach

The implementation will follow an iterative and incremental approach, allowing for continuous development, testing, and feedback incorporation. The application components, including the database, web server, and dashboard, will be developed in parallel and integrated during the implementation process.

4.2 Development Tasks

The development tasks can be divided into the following categories:

1. Database Development:
   - Design and create the database schema.
   - Implement data ingestion mechanisms to populate the database with the selected dataset.
   - Define and enforce proper data integrity constraints.

2. Web Server Development:
   - Implement the REST API endpoints for data querying and retrieval.
   - Perform data cleaning and pre-processing operations as required.
   - Ensure proper authentication and authorization mechanisms for user access.

3. Dashboard Development:
   - Design and develop interactive visualizations using Jupyter Notebooks.
   - Implement data processing and manipulation strategies to generate custom views.
   - Integrate additional base maps and layers to enhance the map-based views.

4. Integration and Testing:
   - Integrate the database, web server, and dashboard components.
   - Conduct unit testing, integration testing, and system testing to ensure functionality and interoperability.
   - Identify and resolve any issues or bugs during the testing phase.

4.3 Resource Allocation

The resources required for implementation include:

- Development Team: Assign developers with relevant skills and expertise in database development, web server development, and data visualization.
- Development Environment: Set up development machines with the necessary software tools and dependencies.
- Development Database: Create a local instance of PostgreSQL and PostGIS for database development and testing.
- Version Control System: Utilize a version control system to manage source code and collaboration.

4.4 Deployment Strategy

The application will be deployed locally, utilizing the chosen technologies and dependencies. The deployment process may involve configuring the web server, setting up the database, and ensuring the necessary libraries and packages are installed. The deployment steps will be documented in the Software Release Document (SRD) to facilitate the installation and running of the application.

5 Testing Plan

A comprehensive testing plan is essential to ensure the quality and functionality of the interactive application for air quality monitoring. The testing activities will include unit testing, integration testing, and system testing. The following sections outline the testing approach, test types, and test cases.

5.1 Testing Approach

The testing approach will follow a combination of manual and automated testing techniques. It will involve testing individual components, as well as the integration and overall functionality of the application. The testing process will focus on verifying the correctness of data retrieval, processing, visualization, and user interactions.

5.2 Test Types

The testing plan will include the following test types:

1. Unit Testing:
   - Test the individual components, such as database functions, API endpoints, and dashboard modules, to ensure they work as expected.
   - Validate the correctness of data retrieval, manipulation, and processing operations.

2. Integration Testing:
    - Test the integration between the database, web server, and dashboard components to ensure proper data flow and functionality.
    - Verify the interoperability of different modules and components.

3. System Testing:
    - Conduct end-to-end testing of the application to validate its overall functionality, including data querying, processing, visualization, and user interactions.
    - Test various scenarios and use cases to identify and address any potential issues or bugs.

5.3 Test Cases

The test cases will cover various aspects of the application's functionality, including:

1. Database Testing:
    - Test the data ingestion process to ensure data is correctly stored in the database.
    - Verify the enforcement of data integrity constraints, such as primary key and foreign key relationships.

2. API Testing:
    - Test the REST API endpoints for data querying and retrieval to ensure accurate and timely responses.
    - Validate the handling of different request parameters and error conditions.

3. Dashboard Testing:
    - Test the interactive visualizations and user interface components to ensure proper rendering and functionality.
    - Validate the manipulation and customization of views, such as selecting specific sensors, time ranges, and aggregation options.

4. User Interaction Testing:
    - Test user interactions, such as selecting a location on the map, filtering data based on specific criteria, and generating custom visualizations.
    - Validate the responsiveness and usability of the application across different devices and screen sizes.

5.4 Test Environment

The testing environment should replicate the production environment as closely as possible. It should include the following components:

- Local instance of the PostgreSQL database with the relevant schema and sample data.
- Configured web server with the REST API endpoints.
- Jupyter Notebook environment with the necessary libraries and dependencies for the dashboard.

5.5 Test Execution and Reporting

The test execution will follow a structured process, including:

- Test Case Preparation: Define the test cases, including input data, expected results, and any preconditions.
- Test Execution: Execute the test cases, record the actual results, and document any issues or deviations.
- Defect Reporting: Report any identified defects or inconsistencies in a structured manner, including steps to reproduce, expected behaviour, and actual behaviour.
- Issue Tracking: Utilize a tracking system or issue management tool to log and monitor the reported issues until they are resolved.
- Test Documentation: Document the test results, including a summary of executed tests, identified issues, and any necessary improvements or recommendations.

6 Bibliography

PostgreSQL Documentation. (2022). PostgreSQL: The World's Most Advanced Open-Source Relational Database. Retrieved from https://www.postgresql.org/docs/

Flask Documentation. (2022). Flask: A Microframework for Python. Retrieved from https://flask.palletsprojects.com/

https://cordis.europa.eu/docs/projects/cnect/3/609023/080/deliverables/001-MYWAYD21RequirementSpecificationAndAnalysiscnectddg1h520142537023.pdf

Robert L. Lux and C. Arden. Pope. Air quality index. U.S. Environmental Protection Agency, 2009.