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: **2.0**
Date: **Jun 18, 2023**

Copyright: **Copyright © 2023 M K, F S, G A Z__All rights reserved**

Revision history

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

Table of Contents

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.

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
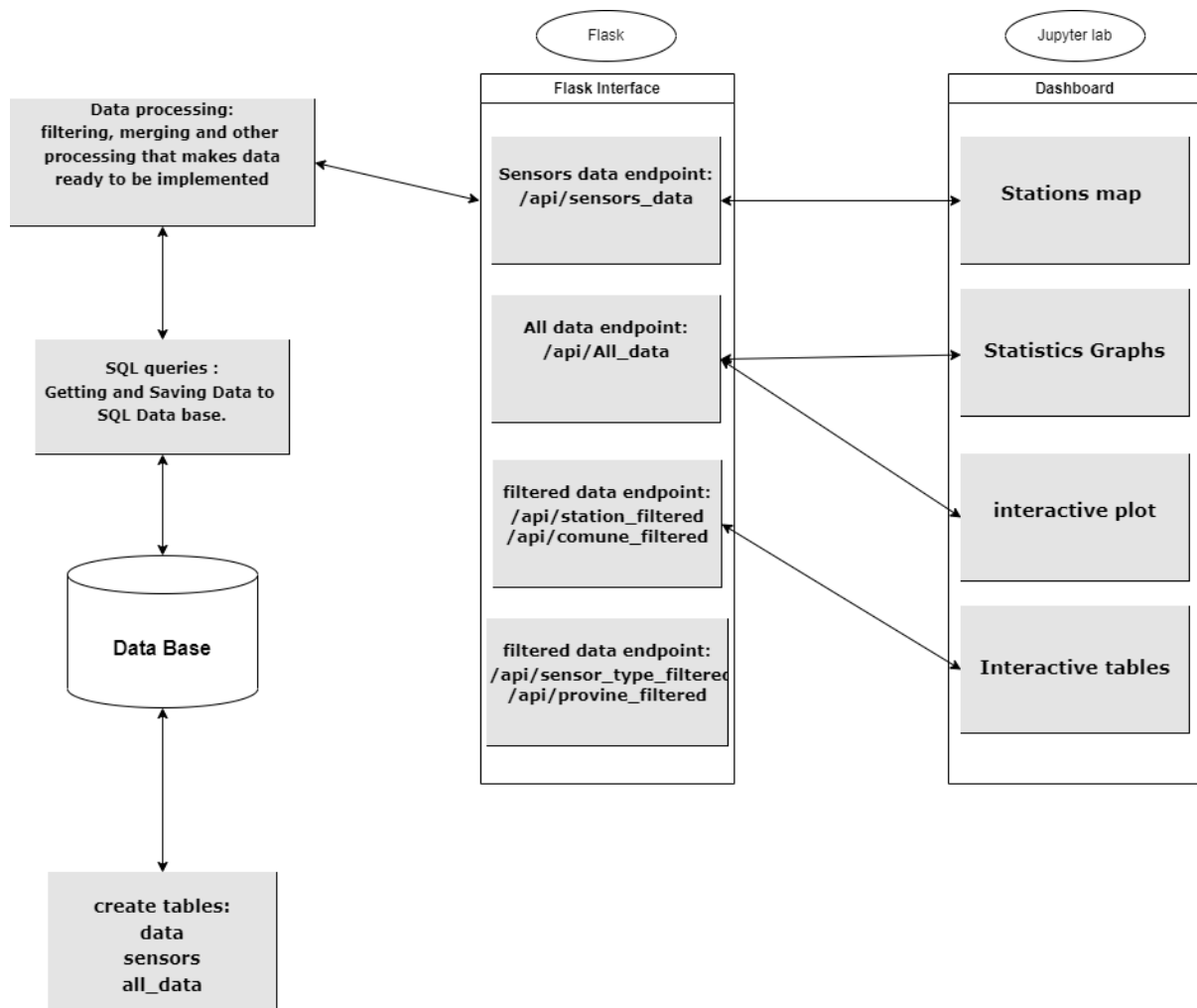
3 Software Design

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. 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.

2. 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. All_data table:
    This is a table we created by merging two tables sensors and data by station ID. It stores the data of stations with the type of their sensors and the spatial location of them.

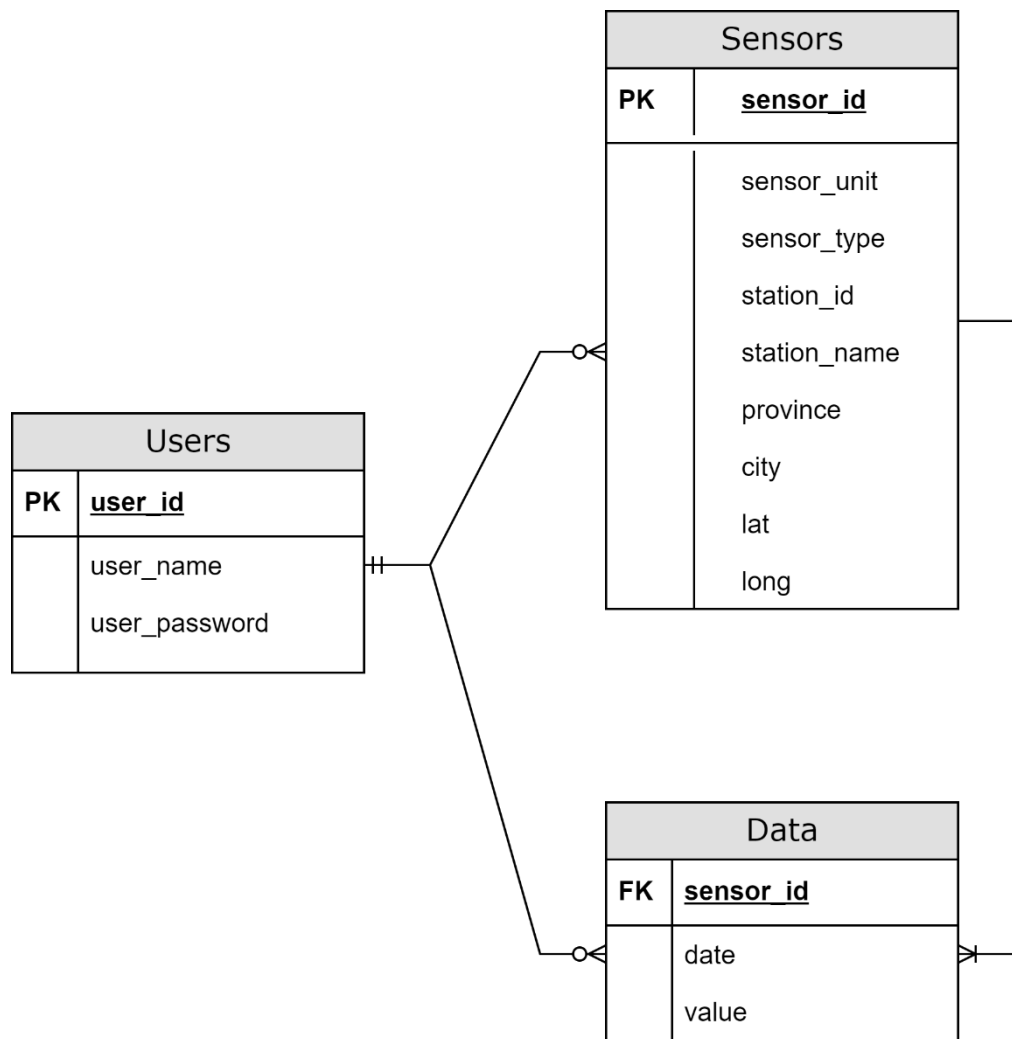### 3.1.1 Specification of tables in Database

Table	Usage	Columns
Sensors	Stored details about sensors and stations	SensorID
		Sensor Type
		unit
		StationID
		StationName
		province
		comune
		lat
		lng
Data	The main stored data by sensors	SensorID
		Date
		value
All_data	Merged data helping us to show value of sensors on a map.	SensorID
		Sensor Type
		Unit
		StationID
		StationName
		Province
		Comune
		Lat
		Lng
		Date
		Value

3.1.1 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.

3.1.2   ER Diagram

Sensors

PK	sensor_id
	sensor_unit
	sensor_type
	station_id
	station_name
	province
	city
	lat
	long

Users

PK	user_id
	user_name
	user_password

Data

FK	sensor_id
	date
	value

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

➕ Maps and Location Selection (the sensors location):

The interface features an interactive map that allows users to select their desired location and see some information like station name and the type of the pollutant the station shows. They can also see a density map of the density of stations, so they will know what areas are monitored and in which area we have more observations.

➕ Data Visualization:

Air quality data is typically presented using visualizations to make it easier for users to understand. Graphs and charts 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).

➕ Filtering:

The user interface provides a functionality, allowing users to choose specific locations, dates, or pollutant types in a drop down menu. Filtering options can help users narrow down the displayed data based on various parameters, such as data in each commune or specific time ranges.
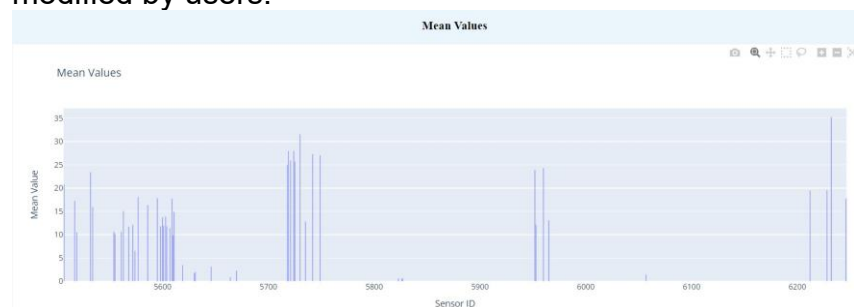
3.2.3 Content Tables

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.
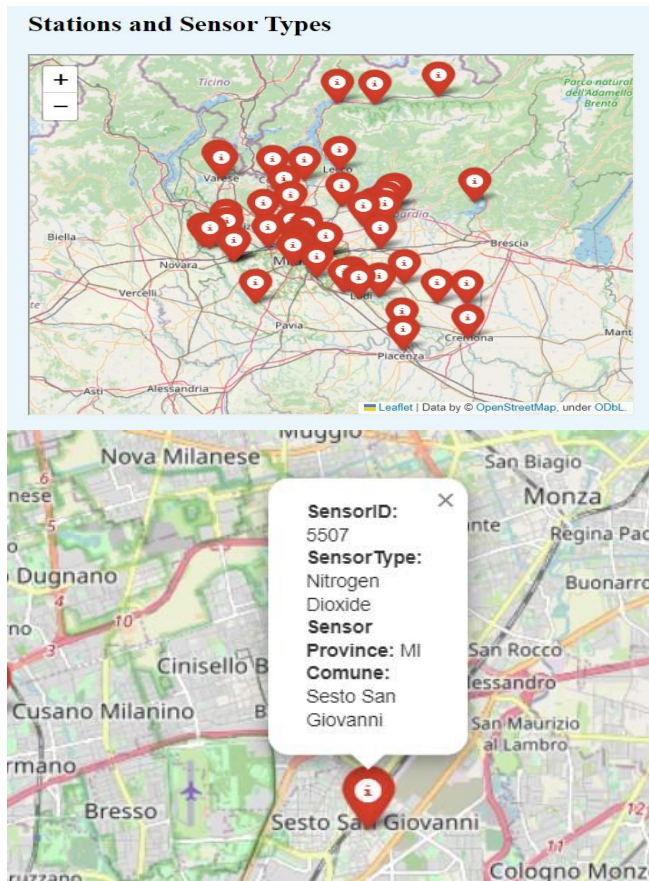
| Static Graphs | Some Graphs like bar plots give the user some overview over the pollutants in the area. Like the minimum values, maximum or average values of the observed values. This plots cannot be modified by users. |

| Drop Down menu | The menus let users to choose some specific options to filter the data. There are a couple of this option in different parts of the dashboard. This is a widget that makes the dashboard interactive. |

Tables   These tables are interactive and show users the filtered data based on what the user chooses, like for a specific commune or a specific station.

Interactive Graphs   It allows the user to see a filtered data in a specific time and sensor type on a graph, and it can change based on what user chooses with provided menus.
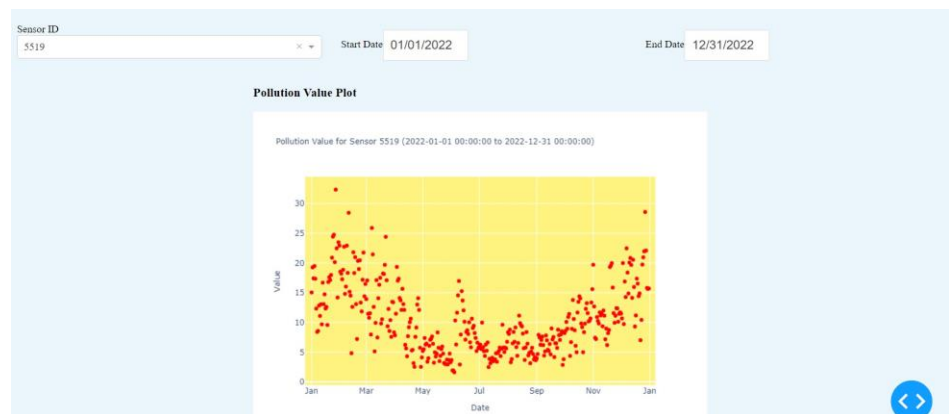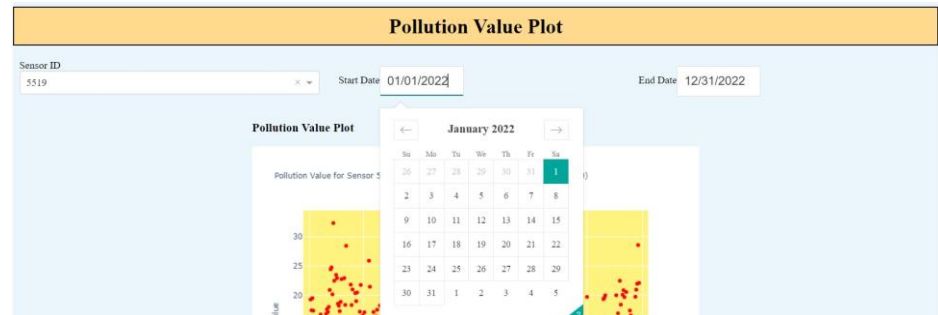
3.2.4 Endpoints:

Sensor data endpoint :	/api/sensors_data	Get data from all sensors
AQ data endpoint	/api/AQ_data	Get air quality observation table
Data filtering endpoints	/api/station_filtered /api/commune_filtered	Get filtered data based on what user chooses

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.
   - 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.

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.