Politecnico Di Milano

Software Engineering for Geoinformatics

REPORT

Requirement Analysis and Specification Document

Authors	*Person Code*
Iyad Abdi	10808634
Filippo Bissi	10438335
Valerio Paoloni	10401564

1 Introduction

1.1 Purpose

The purpose of this study is to design and develop an interactive client-server application to facilitate the efficient and effective querying, processing, and visualization of air quality and weather data from ground sensor stations located in the top 5 cities globally with the highest GPD both in the Global North and Global South, as well as Milan, Italy (given our location and it being the city with the highest GDP in Italy):

Global North:

1. Tokyo, Japan
2. New York City, USA
3. Los Angeles, USA
4. Chicago, USA
5. London, United Kingdom

Global South:

1. Shanghai, China
2. Beijing, China
3. Mumbai, India
4. Sao Paulo, Brazil
5. Shenzhen, China

By utilizing publicly available database management systems, web server development, and data visualization, this application aims to provide users with an informative platform to access and analyze environmental data. By comparing the air quality data from cities with the highest GDPs in both regions, this project seeks to shed light on the globally common sources of air pollution, correlation between productive output and pollution, and identify effective policy solutions to improve environmental conditions and protect public health. Through this application, the research team aims to contribute to the development of a more sustainable and livable environment for people in urban areas around the world.

1.2 Scope

The scope of this application is to design and develop an interactive client-server application that enables users to access, query, and visualize air quality and weather sensor data collected from data collected from the public digital archive available on the *World's Air Pollution: Real-time Air Quality Index* website https://waqi.info and/or the *Air Pollution in the World Real-time Air Quality Index (AQI)* https://aqicn.org. The application consists of three main components: a database where the data is ingested beforehand, a web server (backend) that exposes a REST API used for querying the database and retrieving data, and a dashboard that exploits the web server to provide means for requesting, processing, and visualizing data.

The database component should be capable of storing large time-series datasets if feasible, as well as associated metadata for each observation. The backend component should be able to provide an intuitive and secure API that allows users to query the data and perform various data processing tasks. The dashboard component should provide simple, yet interactive, visualization tools that enable users to explore the data in different ways, such as maps, perhaps dynamic graphs, and other visualization tools. The application should be scalable, reliable, and maintainable, and should adhere to industry-standard software development practices and guidelines as taught in this course. The final deliverable aims to be a fully functional, user-friendly application that allows users to access, query, and visualize air quality and weather sensor data of our 11 cities of focus in a meaningful and informative way.

1.3 Short overview

The technical functionalities relate to the implementation and operation of the application, including the data integration, database management, API development, and data processing functionality. The user-enabled tools relate to the user experience and the ways in which users will interact with the data via the dashboard. The processing functionalities are integral to both the technical and user-enabled functionalities, as they will enable users to process and visualize the data in meaningful ways.

Technical functionalities:

- Ingestion of air quality and weather sensor data from https://waqi.info and/or https://aqicn.org into a database
- Design and development of a proper database schema to store the data and associated metadata
- Development of a REST API for querying and retrieving data from the database
- Implementation of data cleaning and preprocessing functionality within the web server
- Harmonization, aggregation, and filtering of data based on different patterns and criteria
- Space-time aggregations and other data transformations

User-enabled tools:

- Interactive dashboard that allows users to process, visualize, and explore the data retrieved from the web server using original manipulation strategies
- Customizable views that allow users to generate custom visualizations of the data
- Map-based views and interactive graphs that enable users to explore the data in different ways

1.4 Acronyms and other definitions

TERM	**DEFINITION**
AQI	Air Quality Index; a measure of air quality that takes into account the concentration of pollutants such as PM2.5, PM10, O_3, NO_2, SO_2, and CO
Cartogram Map	A map that shows geographical statistical diagrams of various quantities by using dots, curves, or shades
Choropleth Map	A map that shows interval data using shading, coloring, or placing of symbols within predefined areas, to indicate mean values of a specific quantity in those areas
DBMS	Database Management System; software system we are going to use to store, retrieve, and run queries on data
Flask	A Python web framework that allows us to easily build and deploy web applications
Folium	A Python library that allows us to create interactive web maps
GeoPandas	A Python library that allows us to work with geospatial data in Python
GitHub	A web-based hosting service used for the development of our open-source project
JSON	JavaScript Object Notation; a lightweight data interchange format that is easy for humans to read and write, and easy for machines to parse and generate

PostgreSQL	An open-source database management system that uses a client-server model where the server program manages the database files and accepts connections to the database from client applications
Python	A 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
REST	Representational State Transfer; a software architectural style that was created to guide the design and development of the architecture for the Web. It defines a set of constraints for how the Web should behave
REST API	An Application Programming Interface that satisfies the constraints of REST architecture and allows the interaction with RESTful web services

2 Application specifications

2.1 Users

Based on the scope of the application, we envision the users falling into 3 main categories:

1. **Policymakers, public authorities**: Experts can analyze the impact of pollution on economic growth/development and vice versa to better assess environmental policy and economic development. They can use this to gain a comparative understanding of the complex relationships between the environmental and health conditions with economic activities.
2. **Lay user**: Lay users can utilize the web application to explore AQI data in a user-friendly way and understand the relation between of pollution levels and economic activity on a relative basis.
3. **Researchers**: Technicians can take advantage of the usability of the application to run simply and fast quite complex operations like queries and geo-distributed statistics on data of environmental interest.

2.2 Dataset

Data will be retrieved using the REST API of our data source. The real-time data will be collected and integrated from 4 sensors spread across each of the 11 cities in our focus. The data will be stored locally in our database and retrieved on a continuous basis (every hour).

Point positions will provide precise information on the location of air quality and weather sensors which can be used to plot sensor data on a map, visualize spatial patterns in air quality observations, and filter data by location, allowing users to focus on specific areas of interest.

The AQI data variables represent different pollutants that can be measured by air quality sensors:

- **PM2.5**: fine particulate matter with a diameter of less than 2.5 micrometers, which can penetrate deep into the lungs and cause respiratory problems.
- **PM10**: larger particulate matter with a diameter of less than 10 micrometers, which can also contribute to respiratory problems.
- **O3**: ground-level ozone, which can cause respiratory problems and aggravate asthma.
- **NO_2**: nitrogen dioxide, which is a common pollutant from traffic and can cause respiratory problems.

- **SO_2**: sulfur dioxide, which is typically released from burning fossil fuels and can cause respiratory problems and acid rain.
- **CO**: carbon monoxide, which is a colorless and odorless gas that can cause headaches, dizziness, and nausea at high levels of exposure.

Collectively, these AQI data variables provide important information on the levels of different pollutants in the air, which can help inform policies and interventions aimed at reducing air pollution and improving public health.

2.3 Operations

Operations of the web application can be organized into two categories: **user** and **system**.

User operations regard the user's interaction with the application and front-end functionalities via the dashboard. The dashboard provides interactive visualization tools, such as maps and graphs, to allow users to explore air quality data. The frontend communicates with the backend through a REST API, allowing users to request data and retrieve it in the JSON format. The frontend is responsible for providing an intuitive user experience and allowing users to explore and visualize air quality data efficiently. These operations include city selection, AQI data retrieval, data filtering, and data visualization.

System operations regard the backend functionalities of the application. The server ingests air quality and sensor data from our selected ground sensor stations to be stored in a database where it can be queried and retrieved using our data source's API. Once the data is ingested, the web server exposes a REST API to allow users to query and retrieve data based on their requirements. The web server performing data processing tasks before returning results to users such as data, spatial, and temporal aggregations, allowing users to retrieve data for specific time periods and geographic areas. The API returns results in the JSON format, which can be easily consumed by the frontend of the web app.

2.4 Product function

The web application will function by displaying a global map-based view with pinpoints of our selected cities. The map will allow for zooming in and out in order to assess the specific AQI readings of the 4 sensors in each city.

The pinpoint on each city will provide an average aggregate value reading of the respective city's AQI and the corresponding color related to the Air quality. Zoomed-in views on specific cities will show the accurate readings of the select city's 4 sensors as well as the pollutant levels.

Users will be able to select 2 cities at a time to make comparative readings of AQI. (E.g. Comparing AQI of Los Angeles with Sao Paolo.)

2.5 Relevant phenomena

Air quality and air pollution is related to economic activity, as industrial and transportation-related emissions contribute significantly to the concentration of pollutants in the air. As a result, large cities, which concentrate people and higher levels of economic activity, tend to have higher levels of air pollution. This relationship between economic activity and air quality has important implications for public health and environmental policy. By exploring this relationship through the AQI web app, users can gain insights into how economic activity impacts air quality.

2.6 Shared phenomena

The objective of this web application relates to public health and climate change. As these phenomena are not limited to a single individual or group, but rather have a broad societal impact, our user case will not require registration or login functionalities to access the web app. Rather, all visitors of the application's website will be able to interface with it.

3 Use case

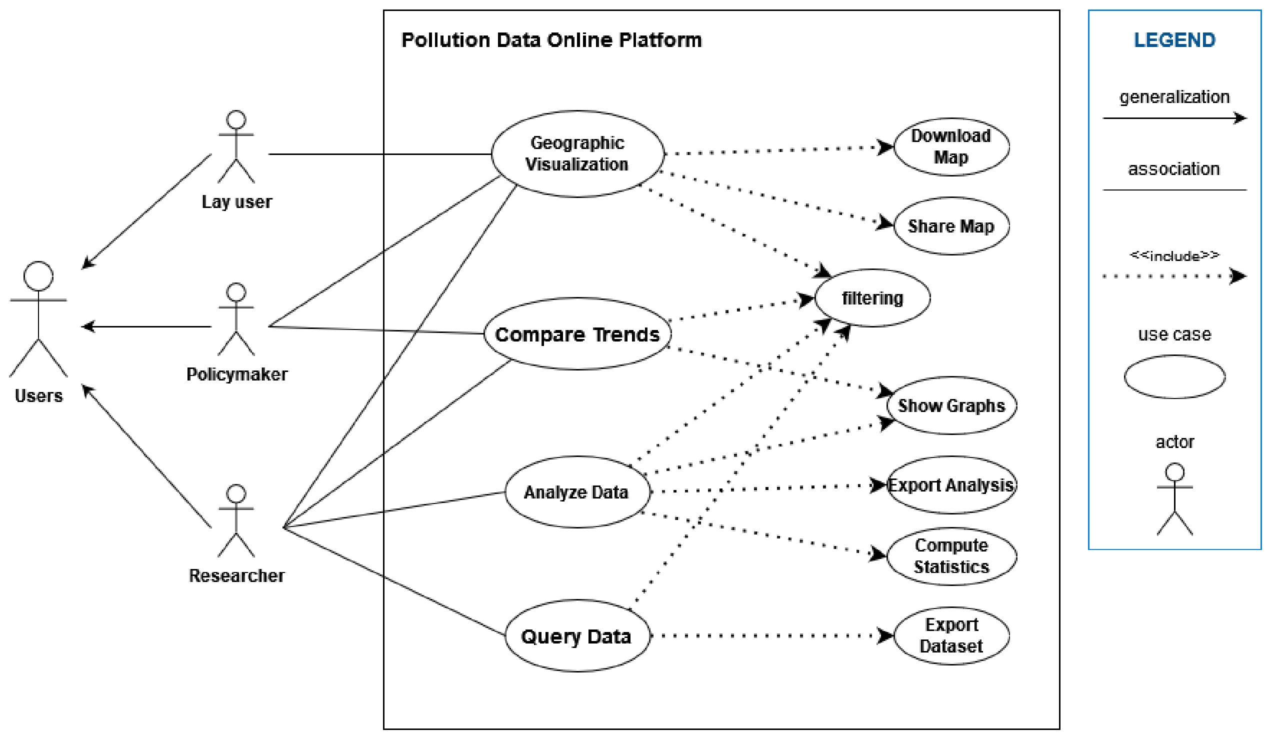
3.1 Introduction

According to the functionalities that the platform is aiming to provide, in the following chart each user is profiled by the potential activities he could undertake in the service.

User \ Action	Policymakers, public authorities	Lay users	Researchers
1		x	x
2	x	x	x
3	x		x
4	x		x
5	x	x	x
6			x
7			x
8			x
9		x	x

List of the actions: (1) General navigation – (2) Querying the data picking specific periods and variables (pollutant/weather) – (3) Visualize correlation between pollutants and weather conditions in specific place (histogram, chart value) – (4) Change interactively the inputs to observe pollution dynamics – (5) Export a graphical output (map/data charts) – (6) Compute Statistics on selected data (7) Export statistic results (georeferenced statistics/summaries) – (8) Export a dataset – (9) Share an output

3.2 Use Case Diagram



3.3 Use Case stories

As a common citizen I can enter the website and have a look about air pollution around me, hoping I can find a clean spot to go jogging	As a researcher I can select a period and some polluted cities to download pollution data I need, without collecting them separately	As a policymaker I can observe some pollution trends in a chart, so that I can take more informed decisions	As a student I can compute some statistics to choose which is the less polluted city to go on Erasmus

3.4 Flow of events

- Geographic Visualization pollution and weather conditions values around the cities:

    1. The user enters the web site (a map is showing updated AQI values in markers)
    2. The user can optionally select a set of pollutants and/or weather quantities
    3. The user can optionally select a past time instant by a calendar
    4. The user clicks on “update” button
    5. The web app sends a request for data to the OpenAQ API to acquire the correct information
    6. The Interactive web map receives the georeferenced data used to fill the cloud of markers around the 11 cities on the map
    7. The user can zoom in/out and scroll the map
    8. The user can download and share the map view image.

    Exit condition: The user sees the interested AQI values and weather conditions on the map.

Exceptions: The OpenStreetMap server or the system crashed.

Special requirements: A default initial query and processing step is needed. Time instant must be in the past.

- Query Data about pollution and weather conditions over the cities:

  1. The user clicks on the “query” button
  2. Appears a box in which the user can select the set of pollutant and/or weather quantities, a period, the set of interested cities and query parameters
  3. The user clicks on “find”
  4. The web app sends a request for data to the OpenAQ API to acquire the correct information
  5. The program receives the quired data and a tick becomes green
  6. The user can export the data by clicking on the "save" button.

  Exit condition: The system allows to download the correct dataset.

  Exceptions: The system crashed. Logical errors in query conditions.

  Special requirements: Time period must be in the past.

- Analyze Data of pollution and weather conditions:

  1. The user clicks on the “compute statistics” button
  2. Appears a box in which the user can select the analysis to make, the variables to use and the filtering of data
  3. The user clicks on “compute”
  4. The web app sends a request for data to the OpenAQ API to acquire the correct information.
  5. The application computes the statistics and show them in a summary
  6. The Interactive web map shows the main quantities of the result in the markers
  7. The user can export the results and the map view.

  Exit condition: The system returns the correct result of the analysis.

  Exceptions: The system crashed.

  Special requirements: The selection must be complete.

- Compare Trends of pollution and weather conditions in two cities:

  1. The user clicks on the “compare” button
  2. Appears a box in which the user can select the two locations, the variables and periods on interest
  3. The user clicks on “view chart” button
  4. The web app sends a request for data to the OpenAQ API to acquire the correct information
  5. The application receives the data, process them and show the results on a chart
  6. The user can dynamically change the chart
  7. The user can download and share the chart.

  Exit condition: The user sees the comparison trends chart between the two cities.

  Exceptions: The system crashed.

  Special requirements: The selection must be complete. The periods must be in the past and of a prefixed and same length (1 month/1 year).

4 Functional requirements and Domain Assumptions

4.1 General user interface requirements

The goal of our web application is to enable a user-friendly way to analyze, compare, query, and visualize the AQI data of our 11 cities. As such, we offer some general UI requirements:

1. Intuitive and user-friendly interface dashboard, with clear navigation and easy access to all features and functions.
2. English-only interface.
3. Open access site that does not require user registration.
4. Interactive and engaging visualizations that allow users to explore and analyze air quality data in various ways, such as maps and charts.
5. Clear and concise documentation and help resources.
6. Potential integration with social media and other external platforms to allow users to share their custom analysis.

4.2 Functional requirements

The functional requirements consider all our use-cases. As such, it will ensure:

1. Data Ingestion: The application should be capable of ingesting air quality and weather sensor data from selected public digital archives, such as the World Air Quality Index project.
2. Data Storage: The application should be able to store large time-series datasets and associated metadata for each observation.
3. Data Processing: The application should be able to perform data cleaning, preprocessing, and harmonization to ensure consistency and quality of the data.
4. Data Retrieval: The application should expose a REST API to allow users to query the database and retrieve data in a timely and efficient manner.
5. Data Visualization: The application should provide users with interactive visualization tools that enable them to explore the data in different ways, such as maps, graphs, and possibly other visualization tools.
6. Customization/filtering: The application should allow users to customize their views of the data by selecting variables, time ranges, and other parameters.
7. Performance: The application should be scalable and reliable, able to handle a large volume of data and users.
8. Accessibility: The application should be hosted on a website to ensure wide accessibility.

4.3 Domain Assumptions

Our domain assumptions in developing this web application are as follows:

1. The air quality and weather sensor data retrieved from the public digital archives are accurate and reliable.
2. The data is publicly available and can be accessed through an API.
3. The database system used is PostgreSQL and can store large time-series datasets, as well as associated metadata for each observation.
4. The web server can handle many requests simultaneously, and is designed to be scalable, reliable, and maintainable.

5. The frontend dashboard is designed to be user-friendly and intuitive, allowing users to easily explore and visualize air quality and weather sensor data retrieved from the backend.

References

Specification document: “SE4GEO Project Assignment A.Y. 2022-2023”.

Giovanni Quattrocchi — Software Engineering for Geoinformatics Course Lecture’s Material

UML Diagrams: https://www.diagrams.net/.

Requirements Analysis Document Template. Available at:
https://www.cs.fsu.edu/~lacher/courses/COP3331/rad.html.

Requirements Document Example. Available at: http://web.cse.ohio-state.edu/~bair.41/616/Project/Example_Document/Req_Doc_Example.html.