POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

An application to support air quality analysis using "Dati Lombardia" sensor data

Deliverable: RASD

Title: Requirement Analysis and Specification Document

Authors: HU TIANQI, YAN SHINUO, SU JIAYI

Version: 1.0

Date: April 10, 2025

Download page: https://github.com/Ricky-HU96/WO-CIAO_SE4GEO-2025-Project

Copyright: Copyright © 2024, H.Y.S – All rights reserved

Version	Date	Change
1.0	April 10,2025	First submitted version

Table of Contents

List of Tables

List of Figure

1 Introduction

1.1 Purpose

The purpose of this project is to develop a client-server application designed to support users in querying, visualizing, and analyzing air quality data retrieved specifically from the Dati Lombardia air quality sensor dataset. The application aims to provide valuable insights into pollution distribution, trends, and exposure risks within the Lombardia region 1.

1.2 Scope

This project encompasses the development of a complete client-server system. The system will integrate and process air quality data from Dati Lombardia and associated sensor geolocation data. It will consist of three core components:

1. A database to ingest and store the processed data for analysis.
2. A web server (backend) exposing a REST API for querying and retrieving the stored data.
3. An interactive dashboard (client), built using Jupyter Notebooks, which will utilize the API to allow users to query, process, analyze, and visualize the air quality data through maps, charts, and statistical representations.

The scope includes data retrieval, processing, storage, API development, and dashboard implementation focusing on visualization (map-based and attribute-based) and basic analysis functionalities, including the generation of custom data views. An optional extension includes developing a strategy for continuous data ingestion.

1.3 Overview

The application will serve as a tool for decision-makers, such as environmental agencies and public health organizations, and potentially other interested parties, to better understand air pollution trends and exposure risks within Lombardia. By providing interactive visualizations (maps, charts) and analytical capabilities based on historical or real-time air quality measurements, the platform aims to facilitate informed decision-making related to public health and environmental monitoring

1.4 Acronyms and definitions

Name	Definition
API	Application Programming Interface: A set of rules and protocols for building and interacting with software applications.
Client-Server	An application architecture that separates tasks between providers of a resource or service (servers) and service requesters (clients).
Dashboard	A user interface that organizes and presents information in an easy-to-read format. In this project, it's the interactive client built with Jupyter Notebooks.
Database	An organized collection of structured information, or data, typically stored electronically in a computer system.
Dati Lombardia	The specific dataset source for air quality sensor data used in this project.
Dataset	A collection of related sets of information that is composed of separate elements but can be manipulated as a unit by a computer.
Flask	A micro web framework written in Python, suggested for the web server component.
GeoPandas	An open-source project to make working with geospatial data in Python easier, extending Pandas data types.
Git	A distributed version-control system for tracking changes in source code during software development.
GitHub	A provider of Internet hosting for software development and version control using Git.
JSON	JavaScript Object Notation: A lightweight data-interchange format, used by the REST API to return data.
Jupyter Notebook	An open-source web application that allows you to create and share documents that contain live code, equations, visualizations, and narrative text.
Pandas	A software library written for the Python programming language for data manipulation and analysis.
PostGIS	A spatial database extender for PostgreSQL object-relational database, allowing GIS objects to be stored.
PostgreSQL	An open-source object-relational database management system

	(DBMS).
Python	An interpreted, high-level, general-purpose programming language. The primary language for the backend.
RASD	Requirement Analysis and Specification Document: This document, outlining the requirements for the software.
REST (API)	Representational State Transfer: An architectural style for designing networked applications, used for the web server's.
Web Server (Backend)	The part of the client-server application responsible for handling requests from the client, interacting with the database, and exposing the REST API.

Table 1: Acronyms and definitions

2 Application domain and phenomena description

2.1 User Characteristics

The primary users of this application are expected to be decision-makers, such as individuals within environmental agencies and public health organizations in Italy, who need to monitor and analyze air quality data. Other potential users could include researchers or analysts studying environmental trends. Users are expected to have basic computer literacy and familiarity with web applications, including interacting with maps and charts. While some users may have expertise in environmental science or data analysis, the dashboard interface should aim for clarity and ease of use, potentially catering to non-expert users as well.

2.2 Dataset

The core dataset is the Dati Lombardia air quality sensor dataset. This includes:

- Air quality measurements (e.g., pollutant concentrations) typically recorded on an hourly basis.
- Sensor data, including geolocation (latitude, longitude) of the monitoring stations.
- Measured pollutant type associated with each measurement.

The system requires the integration of these two data sources into a unified dataset stored within the project's database. The data may be historical or near real-time, depending on the source and ingestion strategy. Links to suggested datasets are provided in the project description.

2.3 Operations

2.3.1 User operations

Users interact primarily through the dashboard and can perform the following operations:

- Query Data: Specify parameters (e.g., time range, geographic area/sensor location, pollutant type) to retrieve specific air quality data subsets.
- Visualize Data: View the queried data on interactive maps showing geographic distribution and on interactive charts/graphs illustrating trends over time or other relationships.
- Analyze Data: Access system-generated statistical analyses or summaries related to pollution levels, trends, and potential exposure risks.
- Customize Views: Manipulate visualization parameters to generate custom views of the data relevant to their specific needs.

2.3.2 System operations

The system performs the following operations, often initiated by user actions or scheduled processes:

- Data Ingestion: Retrieve raw air quality and sensor data from external sources (Dati Lombardia, Sensor Data links).
- Data Processing: Clean, preprocess, and integrate the retrieved data into a structured format suitable for storage and analysis.
- Data Storage: Store the processed data in the designated database (e.g., PostgreSQL/PostGIS).
- API Request Handling: Receive requests from the client (dashboard) via the REST API.
- Database Querying: Execute queries against the database based on API request parameters.
- Data Formatting: Format query results into JSON for transmission back to the client.
- Dashboard Serving: Host and serve the Jupyter Notebook-based dashboard interface to the user's browser.

2.4 Phenomena

2.4.1 World phenomena

- Air Quality Measurement: The concentration value of a specific pollutant at a given time and location.

- Pollutant: Specific chemical substances monitored for air quality (e.g., NO2, PM2.5, PM10, O3).
- Sensor/Station: A physical device at a specific geographic location (geolocation) that measures air pollutants.
- Time: Temporal aspect of measurements (e.g., hourly readings, daily trends, historical periods).
- Location/Region: Geographic coordinates (latitude, longitude) and administrative areas within Lombardia.
- Pollution Trend: Patterns or changes in pollutant levels over time or across space.
- Exposure Risk: Potential health impact related to observed pollution levels.

2.4.2 Machine phenomena

- Database Query: An instruction to the DBMS to retrieve, insert, update, or delete data.
- REST API Request/Response: Communication between the client and server using HTTP methods and JSON data format.
- Data Cleaning/Preprocessing: Algorithmic manipulation of raw data to handle errors, missing values, or transform formats.
- Data Aggregation: Summarizing data over time periods or spatial areas.
- Map Rendering: Displaying geographic data visually on a map interface.
- Chart Generation: Creating graphical representations (lines, bars, etc.) of data.
- JSON Parsing: Interpreting JSON formatted data received from the API.

2.4.3 Shared phenomena

- User Query Input: Parameters selected by the user in the dashboard interface to filter data.
- Displayed Map View: The visual representation of geographic data shown to the user.
- Displayed Chart/Graph: The visual representation of attribute data shown to the user.
- Analysis Results: Statistical summaries or insights presented to the user on the dashboard.
- Custom View Configuration: User-defined settings for visualizing data.

3 Use case

This section describes the main interactions between the user and the system.

Use Case 1: Query Air Quality Data

Field	Description
Name	Query Air Quality Data
User	Any user accessing the dashboard.
Condition	User has accessed the dashboard interface via a web browser.
Flow of Events	1. User selects query parameters using dashboard controls.
	2. User initiates the query.
	3. Dashboard sends a request containing parameters to the backend REST API.
	4. Backend queries the database based on received parameters.
	5. Backend returns the queried data in JSON format to the dashboard.
	6. Dashboard receives and parses the data, preparing it for display.
Exit condition	Data relevant to the query is retrieved and ready for visualization/analysis on the dashboard.
Exceptions	- Invalid query parameters entered by the user.
	- No data found matching the criteria.
	- API or database connection error.

Table 2: Use Case 1: Query Air Quality Data

Use Case 2: Visualize Data Geographically

Field	Description
Name	Visualize Data Geographically
User	Any user accessing the dashboard.
Condition	Air quality data has been successfully queried (Use Case 1) and is available to the dashboard.

Flow of Events	1. Dashboard uses the retrieved data (including geolocation) to render a map view.
	2. Sensor locations are displayed on the map, potentially color-coded or sized based on pollution levels.
	3. User interacts with the map (pans, zooms).
	4. User may select specific sensors on the map to view detailed information (pop-ups).
Exit condition	User has viewed the spatial distribution of the queried air quality data.
Exceptions	- Map rendering errors.
	- Missing or invalid geolocation data for some sensors.

Table 3: Use Case 2: Visualize Data Geographically

3.3 Use Case 3: Visualize Data via Charts/Graphs

Field	Description
Name	Visualize Data via Charts/Graphs
User	Any user accessing the dashboard.
Condition	Air quality data has been successfully queried (Use Case 1) and is available to the dashboard.
Flow of Events	1. Dashboard uses the retrieved data to generate interactive charts/graphs.
	2. User views the charts.
	3. User may interact with charts.
Exit condition	User has viewed trends and patterns in the queried air quality data through charts/graphs.
Exceptions	- Chart generation errors.
	- Insufficient data for meaningful visualization.

Table 4: Use Case 3: Visualize Data via Charts/Graphs

3.4 Use Case 4: View Data Analysis

Field	Description
Name	View Data Analysis
User	Any user accessing the dashboard.
Condition	Air quality data has been successfully queried (Use Case 1) and is available to the dashboard.
Flow of Events	1. Dashboard processes the retrieved data to calculate relevant statistics or insights.
	2. Dashboard displays these analyses/statistics in a dedicated section or alongside visualizations.
	3. User reviews the presented analysis.
Exit condition	User has viewed system-generated analysis regarding pollution distribution, trends, or exposure risks.
Exceptions	- Errors during statistical calculation.
	- Insufficient data for robust analysis.

Table 5: Use Case 4: View Data Analysis

3.5 Use Case 5: Generate Custom Data View

Field	Description
Name	Generate Custom Data View
User	Any user accessing the dashboard.
Condition	Air quality data has been successfully queried (Use Case 1) and is being visualized (Use Case 2/3).
Flow of Events	1. User interacts with dashboard controls to modify visualization parameters.
	2. Dashboard dynamically updates the map/chart/analysis display based on user's selections.
	3. User observes the updated, custom view of the data.
Exit condition	Dashboard display is updated to reflect the user's custom view preferences.

Exceptions	- Selected customization options are incompatible.
	- Errors occur while updating the visualization.

Table 6: Use Case 5: Generate Custom Data View

3.6 Use case diagram

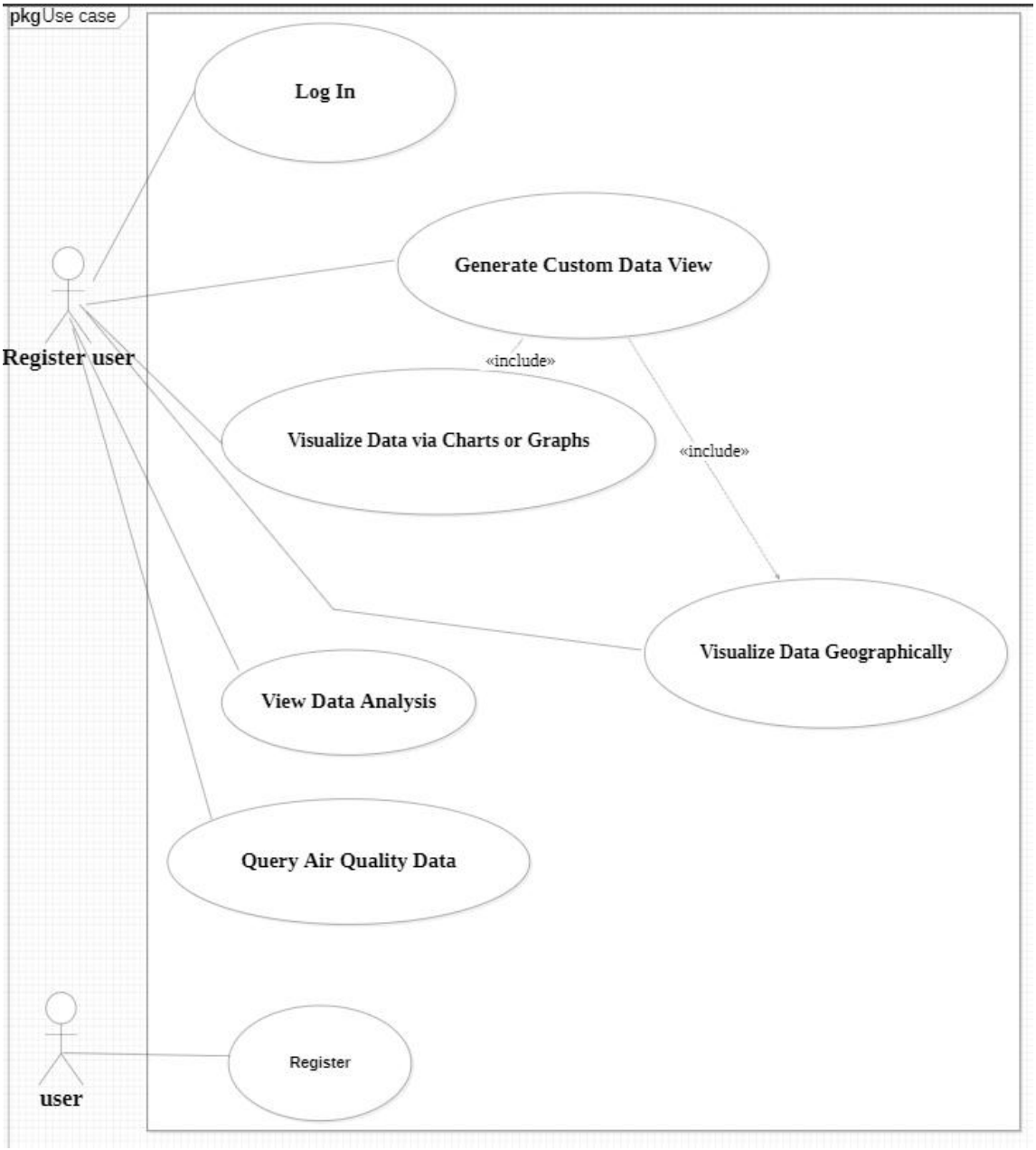

Figure 1: Use Case Diagram

4 Functional requirements and Domain Assumptions

4.1 General user interface requirements

The dashboard interface should be:

- Intuitive: Users should be able to easily navigate between different sections (query, map, charts, analysis) and understand how to interact with controls.
- Clear: Labels, buttons, and visualizations should be clearly presented and easy to understand. Avoid overly technical jargon where possible. Use English as the primary language.
- Responsive: The interface should provide visual feedback for user actions (e.g., button clicks, loading states). Visualizations should update reasonably quickly after queries or interactions.
- Effective Visualization: Maps and charts should accurately and effectively represent the underlying air quality data, facilitating understanding of spatial patterns and temporal trends.

4.2 Requirements

4.2.1 Functional requirements

- R1: Data Integration
  - ✧ R1.1: The system must retrieve air quality data from the specified Dati Lombardia source.
  - ✧ R1.2: The system must retrieve sensor data, including geolocation, from the specified Sensor Data source.
  - ✧ R1.3: The system must combine the air quality and sensor data into a unified dataset containing hourly measurements, pollutant type, and geolocation.
  - ✧ R1.4: The system must store the integrated data in a structured database.
- R2: Backend and API
  - ✧ R2.1: The web server must expose a REST API to allow the dashboard client to query and retrieve data from the database.
  - ✧ R2.2: The REST API must return data to the client in JSON format.
  - ✧ R2.3: The web server may perform necessary data cleaning and preprocessing before returning data via the API.
- R3: Dashboard Functionality
  - ✧ R3.1: The dashboard must interact with the backend REST API to fetch air quality data.

- ✧ R3.2: The dashboard must provide controls for users to query data based on parameters like time range, location/sensor, and pollutant type.
- ✧ R3.3: The dashboard must visualize queried data geographically using interactive map-based views.
- ✧ R3.4: The dashboard must visualize queried data using interactive attribute-based views (e.g., charts, graphs).
- ✧ R3.5: The dashboard must present analysis or statistical summaries related to pollution distribution, trends, and exposure risks based on the queried data.
- ✧ R3.6: The dashboard must allow users to generate custom views by manipulating visualization parameters.

4.2.2 Technological requirements

- R4: Mandatory Technologies
- ✧ R4.1: The web server (backend) developed using Python.
- ✧ R4.2: The interactive dashboard (client) built using Visual Studio Code.
- ✧ R4.3: Version control must use Git, and the project must be hosted on GitHub.

4.3 Domain assumptions

- Data Source Availability: The Dati Lombardia air quality and sensor data sources are accessible via the provided links or documented methods.
- Data Format Consistency: The data retrieved from the sources maintains a reasonably consistent format amenable to automated processing.
- Data Quality: The source data is assumed to be sufficiently accurate for the purpose of visualizing trends and general analysis, though it may contain errors or gaps that the system might need to handle.
- Network Connectivity: Users have a stable internet connection to interact with the client-server application.
- User Literacy: Users possess basic computer literacy and web navigation skills. Target users (decision-makers) have a fundamental understanding of air quality concepts.
- Geographic Context: The analysis and visualization are focused within the administrative boundaries relevant to the Dati Lombardia dataset.