POLITECNICO
MILANO 1863

Software Engineering for Geoinformatics

Requirements Analysis and Specification Document (RASD)

Hananeh AsadiAghbolaghi

Hoda Sadat Mousavi Tabar

Sadra Zahed kachaee

Firoozeh Rahimian

Prof. Quattrocchi

July 2025

Contents

1. Introduction

1.1 Purpose

This Requirements Analysis and Specification Document (RASD) formally describes the necessary features and constraints of the Air-Quality Web Application for Italian air-quality sensor data. Its objective is to ensure all team members and stakeholders share a common understanding of how the system ingests, processes, and presents hourly pollutant measurements—supporting informed development, testing, and deployment.

1.2 Scope

This project will deliver:

- A PostgreSQL/PostGIS database for storing raw hourly air-quality readings (e.g., $PM_{2.5}$, PM_{10}, NO_2, O_3, SO_2, CO) and station metadata across Italy.
- A Flask-based REST API exposing endpoints for sensor metadata, raw measurements, daily aggregates, and available dates.
- A Dash-based web dashboard providing interactive maps and time-series visualizations with controls for pollutant type, date selection, and sensor choice.

Out of scope for this release:

- Automated alert notifications (e.g., email or SMS).
- User authentication; all endpoints are open, read-only.
- Advanced data-entry GUIs for administrators.
- Custom polygon drawing or geographic-area filtering beyond province-level selection.
- Missing data interpolation, outlier detection, or advanced analytics (to be planned for future phases).

2. Overall Description

2.1 System Context

Users access a Dash web application served by Flask, which reads from a PostGIS-enabled PostgreSQL database populated through a Python ETL script. The system supports importing CSV exports of sensor metadata and measurements, real-time API queries, and dynamic visualizations in a browser.

2.2 Product Functions

- **Data Ingestion**: Python ETL script (manage_data.py) loads station metadata and raw hourly measurements from CSV files into the database, filters invalid values, and computes daily aggregates. **Run this first** to prepare data for schema initialization.
- **Database Schema Setup**: Python script (create_table.py) initializes the database schema. **Run this after manage_data.py** to enable PostGIS, drop any existing tables, and recreate the tables

(sensors, raw_measurements, measurements, sensor_pollutants) with appropriate geometry columns and foreign-key constraints.

- **API Services**: Flask app (app.py) provides JSON REST endpoints:
    - /api/sensors and /api/sensors/<id> for station metadata and geometry
    - /api/raw_measurements for filtered hourly data
    - /api/measurements for filtered daily aggregates
    - /api/dates for the list of available dates in the data
- **Interactive Dashboard**: Dash app (dash.py) offers:
    - A map view colored by pollutant daily averages, selectable by pollutant and exact date
    - A time-series chart of hourly values for a chosen sensor and date range
    - Responsive layout using Bootstrap
- **Data Export**: Users can download JSON (via API) and CSV (future enhancement) of both raw and aggregated datasets.

2.3 User Classes and Characteristics

- Public Users: Unauthenticated; can view all maps, charts, and export data via API or UI.
- Administrators: Run ETL scripts to refresh data; no GUI required in this phase.

2.4 Operating Environment

- **Database Server:** PostgreSQL 13+ with PostGIS enabled on Linux or Windows.
- **Backend:** Python 3.9+ with Flask and psycopg2.
- **Dashboard:** Dash & Plotly 2.x, served via Flask; accessible in modern browsers (Chrome, Firefox, Edge).
- **ETL:** Python pandas for CSV processing.
- 2.5 Acronyms and Definitions

Acronym	Definition
API	Application Programming Interface
CSV	Comma-Separated Values
DB	Database
Dash	Plotly Dash web framework
Flask	Python micro-framework for web apps
GeoJSON	JSON format for encoding geographic data
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
$PM_{2.5}$	Particulate Matter ≤ 2.5 µm
PM_{10}	Particulate Matter ≤ 10 µm

PostGIS	PostgreSQL extension for geographic objects
REST	Representational State Transfer
SQL	Structured Query Language
UI	User Interface

3. Functional Requirements

3.1 Data Collection

Requirement ID: DC-01

Title: Pollutant Data Collection

Description: The system shall collect and parse hourly pollutant readings ($PM_{2.5}$, PM_{10}, NO_2, O_3, SO_2, CO) fromCSV files provided by the "Dati Lombardia" network.
Priority: High

Acceptance Criteria:

- All hourly readings for each pollutant and station are successfully imported into the database.
- No malformed or missing records remain; any rejected rows are logged in an ingestion report.

Requirement ID: DC-02

Title: Station Metadata Collection

Description: The system shall collect and parse station metadata (station ID, name, latitude, longitude, municipality, station type) from CSV files.

Priority: High

Acceptance Criteria:

- All station entries appear in the database with correct geospatial coordinates.
- Metadata queries return the expected number of stations without errors.

Requirement ID: DC-03

Title: CSV Schema Validation

Description: The system shall validate incoming CSV records against predefined schema rules (e.g., valid pollutant values, ISO-formatted timestamps, coordinate ranges) and record any validation errors.
Priority: Medium

Acceptance Criteria:

- Invalid rows are excluded from ingestion and detailed in a validation report.
- Valid rows are imported correctly, and the database contains no schema violations.

3.2 Visualization

Requirement ID: SV-01

Title: Interactive Map View

Description: The system shall render pollutant concentrations ($PM_{2.5}$, PM_{10}, NO_2, O_3, SO_2, CO) on an interactive map, with controls for selecting pollutant type and stepping through time. **Priority:** High

Acceptance Criteria:

- Users can open the map and see station markers or a heatmap colored by pollutant value.
- Changing the pollutant selector or time-slider immediately updates the map view (≤ 2 s).

Requirement ID: SV-02

Title: Pollutant-Type Filtering

Description: The system shall allow users to filter all visualizations (map and charts) by choosing one pollutant type at a time.

Priority: Medium

Acceptance Criteria:

- Only data for the selected pollutant appears on the map and in any accompanying charts after filter application.

Requirement ID: SV-03

Title: Time-Series Charting

Description: The system shall generate time-series charts showing the selected pollutant's trend for one or more stations or user-defined regions.

Priority: High

Acceptance Criteria:

- Charts plot correct values with descriptive axis labels and legend.
- Switching the station or region refreshes the chart within 2 s.

Requirement ID: SV-04

Title: Date-Range Filtering

Description: The system shall allow users to adjust the date range for all visualizations via an interactive date-picker or slider control.

Priority: Medium

Acceptance Criteria:

- Visualizations update accurately to reflect only the data within the selected date window.

Requirement ID: SV-06

Title: Threshold-Based Highlighting

Description: The system shall highlight stations or regions where pollutant levels exceed user-defined thresholds, using distinct symbols or colors.

Priority: Medium

Acceptance Criteria:

- All exceedance points are styled differently (e.g., red markers).
- A summary table lists each exceedance event with timestamp, station, and pollutant value.

3.3 Data Processing

Requirement ID: DP-02

Title: Outlier Detection and Flagging

Description: The system shall identify pollutant measurements that lie beyond three standard deviations from the station's rolling 24-hour mean and mark them as outliers.
Priority: Medium

Acceptance Criteria:

- Each outlier record is tagged with an is_outlier boolean in the database.
- A daily summary report lists all outlier events per station.

Requirement ID: DP-03

Title: Scheduled Aggregation After Raw Ingestion

Description:
The system shall compute and store aggregate statistics (daily average, minimum, and maximum values) for each pollutant and station immediately after bulk ingestion of raw data. Aggregation currently occurs as part of the ingestion pipeline and does not update in real time after individual measurements.

Priority: Medium-High

Acceptance Criteria:

- Aggregated daily statistics (daily_avg, daily_min, daily_max) are inserted into the measurements table during batch ingestion from CSVs.
- Aggregation is based only on valid (value ≥ 0 and $\neq$ -9999) raw data.
- The measurements table contains entries for each valid day–sensor–pollutant triplet in the raw data.

- Sample checks confirm that aggregates (e.g., daily max $PM_{2.5}$) match calculations done externally in pandas or PostgreSQL queries.

Requirement ID: DP-04

Title: Spatial Region Attribution via Metadata

Description:
The system uses **existing metadata** in the sensor CSV to assign each station to a province. No PostGIS spatial joins are currently used to assign municipality or administrative boundaries.

Priority: Medium

Acceptance Criteria:

- Each station inserted into the `sensors` table includes a `province` field extracted from the source CSV.
- No automated spatial matching (e.g., municipality assignment via shapefile) is currently performed.
- Queries that group measurements by `province` yield consistent and expected results.
- Future extensions may integrate PostGIS region shapefiles to support full spatial joins and boundary resolution.

Requirement ID: DP-06

Title: Timestamp Parsing and Storage

Description:
The system parses timestamp strings from the input CSV files (assumed to be in local Europe/Rome time format) and stores them as **naive `TIMESTAMP` objects** in the database, without explicitly converting or localizing timezones. No timezone normalization is currently performed.

Priority: Medium

Acceptance Criteria:

- All timestamps are parsed using the format `%d/%m/%Y %H:%M:%S`, matching the source CSV format.
- Timestamps appear consistent with Europe/Rome time in raw and aggregated tables, assuming the source data is already localized.
- The system does **not** currently detect, convert, or store timezone information (e.g., UTC offset).

- Time-series plots follow the correct visual order and granularity as expected by users in the CET/CEST zone.

Requirement ID: UM-01

Title: Open, Read-Only Access

Description: The system shall allow any user to access all dashboard features—visualizations, data exports, maps, and charts—without requiring login or registration.

Priority: High

Acceptance Criteria:

- Users can open the Jupyter Notebook interface and use every feature without encountering a login prompt.
- Attempting to access any endpoint or export functionality does not redirect to an authentication page.

Requirement ID: UM-02

Title: No User Credentials Management

Description: The system shall not implement user account creation, credential changes, or role-based access control in this release; all access is uniformly read-only.
Priority: Medium

Acceptance Criteria:

- There are no UI elements or configuration options for creating, modifying, or deleting user accounts.
- All users share the same level of access; no "admin" dashboard or user-management endpoints exist in the codebase.

3.5 Data Maintenance

Requirement ID: DM-01

Title: Pollutant Risk-Level Categorization

Description:
pollutant concentrations are visualized using **continuous color scales** in map and time-series charts without risk-level labels.

Priority: Low (feature not implemented yet)

Acceptance Criteria:

- Maps and charts apply default color gradients (e.g., continuous color='daily_avg') without reference to categorical risk levels.
- Historical data is visualized consistently, but without explicit risk annotations.

Requirement ID: UF-01

Title: Create Custom Data Query

Description: The system shall allow users to select pollutant type(s) and define a time range to fetch data for a selected station.

Priority: Low

Acceptance Criteria:

• Users can select one or more pollutants and choose a date range.
• A time-series chart visualizes the selected pollutant levels over the given range.
• Data is queried from pre-aggregated daily measurements, not raw data preview.

Requirement ID: UF-03

Title: Access Station and Pollutant Metadata

Description: The system shall allow users to view all available stations and the list of pollutants they measure.

Priority: Low

Acceptance Criteria:

• The map and dropdowns load station names and coordinates from the /api/sensors endpoint.
• Users can inspect station metadata and associated pollutant types from the dropdown context.

Requirement ID: UF-04

Title: View Interactive Map

Description: The system shall allow users to interact with a dynamic map, selecting a date and pollutant to view daily pollutant levels across stations.

Priority: Low

Acceptance Criteria:

• Users can pan/zoom the map and update its data by changing the pollutant and date.
• Marker size and color reflect daily average concentration levels.

Requirement ID: UF-05

Title: Validate Data Coverage

Description: The system shall visually indicate when no data exists for a selected date and pollutant on the map.

Priority: Low

Acceptance Criteria:

• If no data is available for the selected combination, an empty map with a message is shown.
• Data availability gaps are not explicitly listed, but missing data is visually noticeable.

Requirement ID: UF-06

Title: Modify Data Query

Description: The system shall allow users to change pollutant, sensor, or time range inputs and immediately update visualizations.

Priority: Low

Acceptance Criteria:

• Dash callbacks ensure that visualizations update dynamically upon user interaction.
• No page refresh or manual re-querying is required.

Requirement ID: UF-07

Title: Highlight Data by Filters

Description: The system shall allow pollutant-based filtering and map coloring but does not support station-type or threshold-based highlighting yet.

Priority: Low

Acceptance Criteria:

• Users can view differences in pollution levels via color and size encoding on the map.
• Advanced highlighting by station type or threshold values is not currently supported.

4. Non-Functional Requirements

Requirement ID: NF-01

Title: Interoperability – CSV Format Handling

Description: The system shall ingest CSV files from "Dati Lombardia" with expected column formats. It currently assumes fixed column names and delimiters but may require updates for changes.
Priority: Medium

Acceptance Criteria:

• CSVs with the current column order and delimiter are correctly parsed.
• Parsing will fail with unexpected formats, requiring minor script updates.
• Future-proofing for schema variations is not yet implemented.

Requirement ID: NF-02

Title: Maintainability – Modular Architecture

Description: The system separates concerns across ingestion (manage_data.py), API (app.py), and visualization (dash.py), enabling focused debugging and development.

Priority: High

Acceptance Criteria:

• Ingestion, REST API, and dashboard can be updated independently.
• Adding new pollutants or sources only affects manage_data.py.
• No critical interdependency between modules when modifying data logic.

Requirement ID: NF-03

Title: Performance – Visualization Responsiveness

Description: The dashboard provides responsive interactivity using pre-aggregated daily data. Time-series and maps generally update within 1–2 seconds under normal conditions.
Priority: High

Acceptance Criteria:

• Dashboard graphs respond within 2 seconds for average-size queries.
• Responsiveness may degrade with extremely large date ranges or raw hourly queries.

Requirement ID: NF-04

Title: Security – Data Transmission

Description: The system currently runs locally using unencrypted HTTP connections. TLS/SSL encryption is not yet configured.

Priority: Low (for local testing)

Acceptance Criteria:

• To be upgraded during production deployment.

• All external exposure should go through HTTPS with proper certificates.

Requirement ID: NF-05

Title: Data Protection – Encryption at Rest

Description: PostgreSQL data is stored unencrypted in a local environment. Security measures such as encryption at rest are not yet enabled.

Priority: Low (for local dev)

Acceptance Criteria:

• On production deployment, tablespaces and backups must be encrypted.
• Current environment assumes physical security.

Requirement ID: NF-06

Title: Compatibility – Cross-Browser Support

Description: The dashboard interface built with Dash works reliably on modern browsers like Chrome and Firefox. Jupyter compatibility is not in scope.
Priority: Medium

Acceptance Criteria:

• Manual testing confirms correct rendering and interactivity in Chrome and Firefox.
• No critical UI errors during map or chart interactions.
• Jupyter Notebook is not part of the current user workflow.

Requirement ID: NF-07

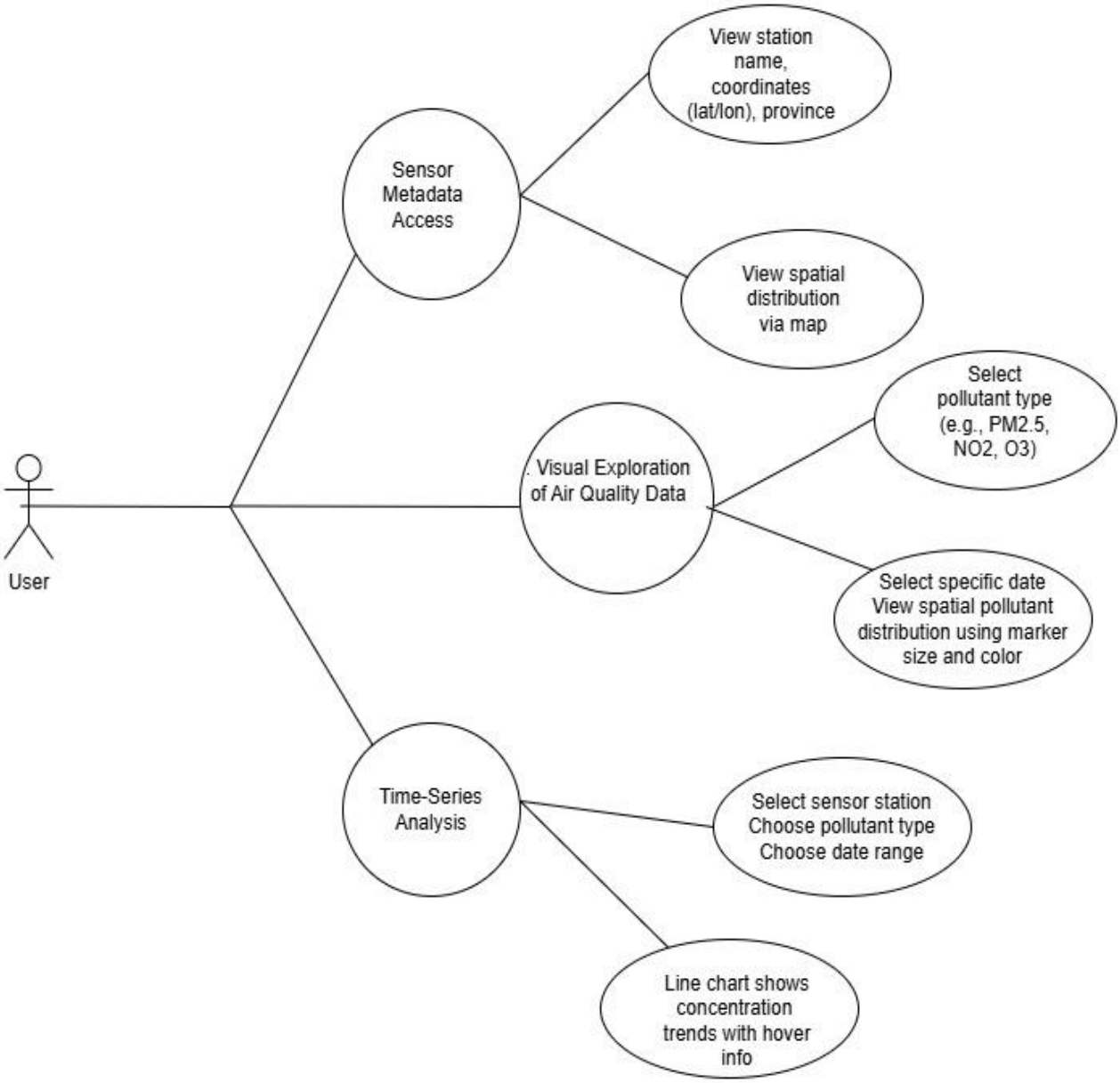
Title: Scalability – Infrastructure Support

Description: The current system is designed for local deployment and does not yet support automated scaling or distributed infrastructure.
Priority: Low–Medium

Acceptance Criteria:

• PostgreSQL and Flask can be containerized or deployed to cloud services.
• Horizontal scaling and load balancing require future DevOps setup.

5.Use Case Diagram



6. User Stories

User Story ID	Title	As a…	I want to…	So that…	Requirement ID(s)
UP-01	Import Pollutant Data	Administrator	upload pollutant CSV files	the system can ingest and clean hourly pollutant measurements	DP-01, DC-01
UP-02	Import Station Metadata	Administrator	upload station metadata from CSV	station locations are correctly stored with coordinates and regions	DP-04, DC-02
UP-03	Validate and Clean CSVs	System	ensure CSVs match expected schema and drop invalid rows	only clean and usable data is processed	DP-02, DC-03
UP-04	View Interactive Map	User	view daily pollutant levels across stations on a map	I can visualize spatial variation in air quality	SV-01, UF-04
UP-05	View Time-Series Chart	User	toggle between raw and aggregated time-series per sensor	I can explore pollution trends at different resolutions	SV-02, DP-03, UF-01
UP-06	Export Raw or Daily Data	User	export either raw hourly or aggregated daily readings as CSV	I can analyze or archive the data externally	DE-01, DE-02
UP-07	Filter by Pollutant and Time	User	filter map and charts by pollutant and date range	I can focus on pollutants and timeframes of interest	SV-03, SV-04, UF-06
UP-08	Highlight High Concentrations	User	visually highlight stations with high daily pollutant values	I can identify pollution hotspots easily	SV-06, UF-07
UP-09	Define Risk Categories	Administrator	configure pollutant thresholds and their risk levels	users can interpret pollution levels meaningfully	DM-01

User Story ID	Title	As a…	I want to…	So that…	Requirement ID(s)
UP-10	Update Station Info	Administrator	update names, coordinates, and regions of stations	mapping and reports stay accurate	DM-02
UP-11	Region Mapping via PostGIS	System	assign province names to stations using coordinates	users can filter and group by province	DP-04
UP-12	Timezone Normalization	System	store all timestamps in Europe/Rome time	charts and exports match local time correctly	DP-06

7. Future Considerations

1. Advanced Forecasting & Analytics

- Integrate machine learning models (e.g., time-series forecasting, anomaly detection) to **predict pollutant trends** and **identify pollution anomalies**.
- Provide **scenario simulations** (e.g., reduced emissions due to lockdowns or traffic policies) to support strategic urban planning.
- Explore clustering of stations by behavior patterns to uncover regional pollution dynamics.

2. Real-Time Data Streaming

- Upgrade from batch-based CSV ingestion to **real-time data pipelines** using technologies like **Apache Kafka** or **MQTT**, enabling continuous sensor updates.
- Trigger **live notifications** when pollutant levels exceed risk thresholds, enhancing responsiveness for health and safety.

3. Collaborative Dashboards

- Enable **dashboard sharing and annotations** for stakeholders to collaborate (e.g., between researchers and policy makers).
- Implement **role-based access control (RBAC)** so different user groups (e.g., citizens, environmental agencies) see relevant features and data views.

4. Mobile & Responsive Design

- Enhance the current dashboard UI for **responsive layout** on tablets and mobile devices.
- Optionally develop a **lightweight progressive web app (PWA)** or native mobile versions to support real-time alerts and on-the-go data exploration.

5. Integration with External Data Sources

- Link pollution data with **weather datasets (e.g., ERA5, ARPA Lombardia)**, **traffic volumes**, and **health statistics** to conduct impact studies.
- Develop **RESTful and/or GraphQL APIs** to allow external apps (e.g., city dashboards, public health tools) to consume filtered pollutant data.

6. Automated Data Acquisition

- Add **scheduled scripts or pipelines** to automatically pull pollutant and station updates from sources like **Dati Lombardia** or ARPA APIs.
- Implement dynamic schema recognition to automatically adapt to new pollutants or metadata formats without manual changes.

7. User-Defined Alerts & Notifications

- Allow users to **subscribe to alert triggers** (e.g., when $PM_{2.5}$ exceeds 50 $\mu g/m^3$ in their region).
- Provide **periodic summary reports or digests** (daily or weekly), configurable per user preferences and delivery channel (email, app, etc.).

8. Accessibility & Internationalization

- Apply **accessibility standards** (WCAG 2.1) to ensure compatibility with screen readers, keyboard navigation, and color-safe design.
- Support **multi-language deployment** (e.g., Italian, English, French), with UI translations and region-specific pollutant categories.