SE4G 2024-2025

Requirement Analysis and Specification Document

Aether-Analytics

Bingwen Hao
Hangyun Wang
Yujue Wang
Xinchen Yu

Contents

1. Introduction

1.1 Background

In recent years, the increasing threat of air pollution to public health and the environment has raised growing concern across Europe. In response, Italian public authorities have been actively collecting, processing, and publishing air quality data to improve pollution monitoring and support evidence-based policy decisions. The Dati Lombardia open data platform offers detailed air quality sensor data for the Lombardia region, including both real-time and historical pollutant measurements, sensor locations, and metadata.

This project was developed in this context to create a client-server application that enables users to query, visualize, and analyze air quality data in an interactive and meaningful way. By integrating sensor data with map-based and temporal visualizations, the system aims to support environmental agencies, public health organizations, researchers, and citizens in identifying pollution patterns, understanding exposure risks, and informing actions toward air quality improvement.

1.2 Purpose

The purpose of this project is to develop a client-server web application that supports the visualization and analysis of air quality data provided by the Dati Lombardia platform.

The system is designed to assist users at different levels — from government decision-makers to researchers and concerned citizens — in understanding pollution levels, analyzing trends, and assessing public health risks across the Lombardia region.

It offers interactive tools to explore geospatial and temporal data using maps, charts, and customizable visualizations.

1.3 Scope

The goal of this project is to develop a client-server web application that

enables users to explore, visualize, and analyze air quality data collected from the Dati Lombardia platform. The system integrates data from multiple sources, including pollutant measurements and sensor metadata, to support spatial and temporal air pollution analysis at different administrative levels within the Lombardia region.

The system consists of three main components:

A spatial database, used to store and manage historical and/or real-time sensor data;

A backend web server, responsible for data cleaning, filtering, and exposing REST APIs that return data in JSON format;

An interactive dashboard, built using Jupyter Notebooks, which retrieves data via the API and allows users to visualize and explore it through maps, graphs, and custom views.

The application is designed as a visualization and analysis tool, and does not include machine learning or real-time forecasting features. Additional layers or base maps may be integrated as an optional enhancement.

2. Overall Description

2.1 Product Perspective

This project aims to develop a client-server application for air quality analysis using open datasets provided by *Dati Lombardia*. The system is designed to support decision-makers, researchers, and general users in querying and understanding pollution patterns and potential health risks across different areas in the Lombardy region.

The data processed by the system comes from two primary sources:

- **Air quality measurement data**, including pollutant concentrations and timestamps;
- **Sensor metadata**, including geolocation, station information, and monitored pollutants.

The system architecture is composed of three main components:

1. **Data Integration and Storage (Database)**:
   - Periodically fetches air quality and sensor metadata from the official APIs;
   - Merges and cleans the data based on timestamps and geolocation;
   - Stores the unified dataset in a PostgreSQL database with PostGIS extension for spatial queries.
2. **Backend Service (Web Server + REST API)**:
   - Implemented using Flask;
   - Provides RESTful API endpoints to query data based on user-specified filters;
   - Returns data in JSON format for easy integration and visualization.
3. **Data Visualization Dashboard (Jupyter Notebook)**:
   - Displays interactive maps using libraries like Folium or IpyLeaflet;
   - Presents pollution trends and comparisons using Plotly or Matplotlib;
   - Allows users to configure views, select pollutants, and export visualizations.

This modular, extensible system serves as a practical tool for sustainable urban planning, public health assessment, and environmental monitoring in Lombardy.

2.2 Product Functions

The main functions of the system include:

- **Data Integration**: Automatically retrieve and clean both air quality and sensor data from Dati Lombardia APIs, merge them into a single dataset;
- **Database Management**: Store structured data in a spatially enabled PostgreSQL/PostGIS database optimized for time-series and geospatial queries;
- **API Services**: Provide RESTful endpoints allowing users to query by:

  - Time range (e.g., historical or real-time),
  - Geographic area (e.g., municipality or region),
  - Pollutant type (e.g., PM2.5, NO_2);

- **Visualization Dashboard**:
  - Interactive pollution maps with sensor location and AQI levels;
  - Time series graphs for selected pollutants;
  - Support for customized views, pollutant comparisons, and chart export;
- **Data Quality Indicators**: Handle missing or incomplete data gracefully and display data confidence levels where applicable;
- **User-Friendly Interface**: Designed for both technical and non-technical users, with widgets, sliders, and dropdown filters.

2.3 User Classes and Characteristics

User Type	Role	Needs	Technical Skill
Decision-makers	Government, public agencies	Monitor air quality trends, support policy	Medium
Researchers/ Analysts	Academics, data scientists	Deep analysis of pollutant patterns, data export	High
Educators/Students	Teachers, learners	Use for educational demos or projects	Medium to High
General Public	Concerned citizens	Simple access to maps and pollution indicators	Low

The system is designed with inclusivity in mind: the dashboard offers simplified interaction for general users, while the API is fully open and flexible for power users and developers.

2.4 Assumptions and Dependencies

To ensure the correct functioning of the system, the following assumptions and dependencies apply:

- **Data Availability**:
  - The Dati Lombardia APIs for both sensor and air quality data are continuously accessible;

The structure of these APIs remains stable during the development phase;

- **Network and Platform Requirements**:

  The system requires internet access to fetch data and serve client requests;

  Users need a web browser compatible with Jupyter Notebook (Chrome/Firefox recommended);

- **Technical Stack Dependencies**:

  PostgreSQL + PostGIS for data storage and spatial indexing;

  Flask for RESTful backend services;

  Pandas and GeoPandas for data manipulation;

  Folium/IpyLeaflet and Plotly/Matplotlib for interactive visualization;

  GitHub for source code versioning and team collaboration;

- **Non-functional Assumptions**:

  The system is deployed on a horizontally scalable server;

  No user login is required for accessing public dashboards or APIs, but access control could be implemented later.

2.5 Acronyms and Definitions

Acronym/ Term	Definition
API	Application Programming Interface, used to fetch data from external sources
REST	Representational State Transfer, a standard architectural style for web services
JSON	JavaScript Object Notation, a lightweight data-interchange format
PostgreSQL	A powerful open-source relational database
PostGIS	A spatial extension for PostgreSQL, enabling geographic objects and queries
Flask	A lightweight Python web framework for building APIs
Pandas	A Python library for data manipulation and analysis
GeoPandas	An extension of Pandas for working with geospatial data

Folium	A Python library for building interactive Leaflet.js maps
Plotly	A Python graphing library for creating interactive charts
Dashboard	An interface that displays data in interactive maps and graphs
Dati Lombardia	The official Open Data portal of the Lombardy Region

3.Specific Requirements

3.1 Use cases

Use Case1: Register as a User	
User	Unregistered User
Condition	1. The user has accessed the application but has not logged in. 2. A stable internet connection is available.
Flow of Events	1. User clicks "Register" button. 2. Enters username, email, password, and selects user type (general/expert). 3. Clicks "Submit". 4. System validates and stores the data. 5. Redirects to login screen.
Exit Condition	User account is successfully created and ready for login.
Exception Handling	1. The username or email already exists. 2. The password format does not meet security requirements. 3. Registration fails due to network issues.

Use Case U2: Login

User	Registered User
Condition	1. The user has already registered. 2. Login form is displayed.
Flow of Events	1. User enters email or username and password. 2. Clicks “Login”. 3. System authenticates credentials. 4. Redirects to dashboard upon success.
Exit Condition	User is authenticated and session is active.
Exception Handling	1. Incorrect username or password. 2. User not found. 3. Timeout or network error during login.

Use Case3: Explore Dashboard	
User	Logged-in User
Condition	1. The user has successfully logged in. 2. Dashboard is loaded.
Flow of Events	1. Dashboard loads default data view. 2. User views map, graphs, or statistics. 3. Can navigate across different sections.
Exit Condition	User interacts with the dashboard or proceeds to filter or profile.
Exception Handling	1. Data loading fails. 2. Map does not render properly. 3. Graph components crash or freeze.

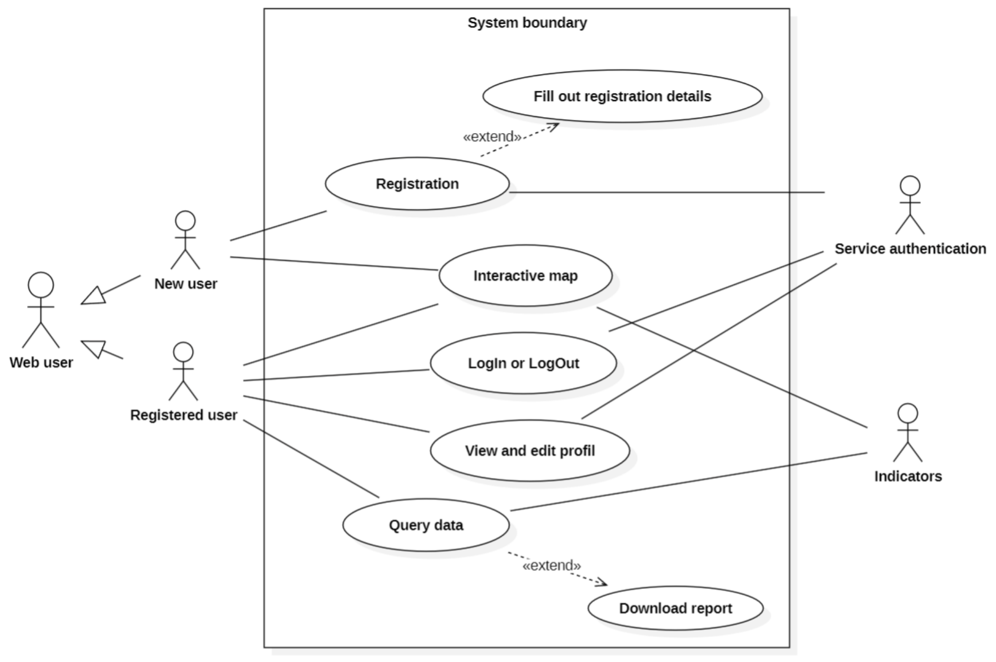
Use Case4: Filter Data by City, Pollutant, and Time	
User	Logged-in User
Condition	The dashboard is loaded and data is available.
Flow of Events	1. User selects a city. 2. Chooses a pollutant type. 3. Selects a time range. 4. System updates the visualization accordingly.
Exit Condition	Filtered data is correctly shown on the dashboard.
Exception Handling	1. No data found for the selected filter. 2. Invalid or conflicting time interval. 3. Visualization fails to update.

Use Case5: View and Edit Profile	
User	Logged-in User
Condition	The user is logged in and accesses “Profile” page.
Flow of Events	1. User clicks “Profile” → “Edit Profile”. 2. Edits name, email, phone, password, or user type. 3. Clicks “Save changes”. 4. System validates and updates the information.
Exit Condition	User information is successfully updated.
Exception Handling	1. Invalid email format. 2. Database update failure. 3. Insufficient permission to modify certain fields.

Use Case6: Export Report	
User	Logged-in User
Condition	The user is on the dashboard with filtered data shown.
Flow of Events	1. User clicks “Export”. 2. Chooses format (e.g., image or PDF). 3. System generates and downloads the report.
Exit Condition	The report is downloaded to user’s device.
Exception Handling	1. No data to export. 2. File generation failed or resulted in a corrupted file. 3. Export blocked due to permission or system error.

Use Case7: Log Out	
User	Logged-in User
Condition	User is logged in.
Flow of Events	1. User clicks “Log Out”. 2. Session is ended. 3. System redirects to login screen.
Exit Condition	User session is terminated and access is reset.
Exception Handling	1. Session termination interrupted by network error. 2. Page redirection fails. 3. Session not cleared properly.

3.2 Diagram Representation



3.3 Functional Requirements

3.3.1 Course Data Collection

Description:

The system shall collect and parse course-related data from university-provided sources such as APIs or structured files (e.g., JSON, XML, CSV). The data includes course name, code, credits, level (Bachelor/Master), descriptions, and learning outcomes.

Rationale:

Ensures the system has access to the most up-to-date and comprehensive information needed for visualization, prerequisite checking, and plan building.

Priority: High

Acceptance Criteria:

• All available course data is stored correctly.

• Data is displayed accurately in the user interface.

• In case of format discrepancies, errors are logged and flagged.

3.3 Programme Tree Visualization

Description:

The system shall generate a tree-structured visual representation of academic programmes. Nodes represent individual courses, while edges indicate prerequisite relationships.

Rationale:

Helps users better understand course dependencies and their academic progress.

Priority: High

Acceptance Criteria:

• The tree view dynamically displays all courses.

• Courses are grouped by academic year or period.

• Relationships (e.g., prerequisites) are clearly shown.

3.3.3 Programme Reorganization

Description:

Users shall be able to interactively add, remove, or replace courses in the programme tree structure. The system should allow temporary inconsistency in prerequisites during editing, but will prompt users to fix it before final submission.

Rationale:

Allows users to customize their study plan based on personal goals or past performance.

Priority: High

Acceptance Criteria:

• Modified trees reflect user changes in real-time.

• Visual indicators flag courses with unmet prerequisites.

3.3.4 Prerequisite Validation

Description:

During and after reorganization, the system shall automatically validate each course’s prerequisites and highlight any violations.

Rationale:

Ensures the integrity of the academic path and prevents students from creating invalid plans.

Priority: High

Acceptance Criteria:

• A validation engine checks each course.

• Errors are flagged with warnings or colored indicators.

3.3.5 Filtering and Highlighting

Description:

Users shall filter and highlight courses in the programme tree based on attributes such as academic year, main area (e.g., DVA, ELA), and learning outcomes (e.g., teamwork, report writing).

Rationale:

Improves navigation and helps users focus on relevant information.

Priority: Medium

Acceptance Criteria:

• Multiple filters can be combined.

• Highlighted courses are visually distinguishable (e.g., by color or border style).

3.3.6 User Management

Description:

The system shall support user registration, authentication, password recovery, and credential updates.

Rationale:

Secure access is necessary for saving personalized study plans and managing data.

Priority: Medium

Acceptance Criteria:

• Input fields are validated.

• Unique email and strong password required.

• Authentication follows standard security protocols.

3.3.7 Unregistered User Capabilities

Description:

Unregistered users shall have limited access to the system. They can browse programmes, build temporary study plans, validate them, and export them locally (e.g., as JSON).

Rationale:

Lowering the entry barrier encourages broader use while protecting data.

Priority: Low

Acceptance Criteria:

• No registration is required.

• Plans are session-based and not stored.

• Export contains all plan details.

3.4 Non-functional Requirements

3.4.1 Security – Data Protection

Description:

User data shall be protected using encryption-at-rest and secure communication protocols (TLS/HTTPS). Access control and authentication mechanisms must be enforced.

Priority: High

Acceptance Criteria:

• All communication is encrypted.

• Sensitive data is stored securely in the database.

3.4.2 Privacy Compliance

Description:

The system must comply with the EU General Data Protection Regulation (GDPR), ensuring informed consent, data minimization, and right to deletion.

Priority: High

Acceptance Criteria:

• Users are informed about data usage.

• Users can request data deletion.

• No unnecessary personal data is collected.

3.4.3 System Maintainability

Description:

The system shall be designed using a modular architecture (e.g., microservices or component-based front-end), facilitating independent development and upgrades.

Priority: High

Acceptance Criteria:

• Components can be updated independently.

• Clear documentation is maintained.

• System downtime during maintenance is minimal.

3.4.4 Platform Compatibility

Description:

The application must function properly across modern browsers (Chrome, Firefox, Edge).

Priority: Medium

Acceptance Criteria:

• All features work without layout or functionality issues across browsers.

• Mobile compatibility is a future goal.

3.4.5 Performance and Scalability

Description:

The system shall be able to scale horizontally and vertically to serve more users and handle more data.

Priority: Medium

Acceptance Criteria:

• Response time remains under 2 seconds under typical load.

• Load testing confirms system stability.

3.4.6 Password Policy

Description:

Passwords must meet complexity requirements (e.g., min 8 characters, alphanumeric and symbol mix) and be hashed using industry-standard algorithms.

Priority: High

Acceptance Criteria:

• Weak passwords are rejected.

• Passwords are never stored in plaintext.

• Optional periodic password renewal.

3.5 Technological Requirements

3.5.1 Data Integration

Description:

The system shall support importing academic data through structured APIs or uploaded files. Initial implementation is based on MDU programme data.

Priority: High

Acceptance Criteria:

• Data sources are configurable.

• Invalid data is logged and handled gracefully.

3.5.2 Web-Based Architecture

Description:

The system shall follow a client-server model using modern frameworks (e.g., React for frontend, Node.js/Python for backend).

Priority: High

Acceptance Criteria:

• Frontend and backend are decoupled.

• API endpoints follow REST or GraphQL standards.

3.5.3 Database Requirements

Description:

The database (e.g., PostgreSQL, MongoDB) must support indexing, querying, and encrypted storage of course, programme, and user data.

Priority: High

Acceptance Criteria:

• All data is queryable via indexed fields.

• Encryption-at-rest is enabled for sensitive data.

3.5.4 Security Protocols

Description:

Authentication and authorization should be implemented using OAuth 2.0 or similar frameworks. All client-server communication must occur over HTTPS.

Priority: High

Acceptance Criteria:

• Session tokens are securely handled.

• Unauthorized access is blocked.

3.5.5 Hosting and Deployment

Description:

The system shall be deployed on cloud infrastructure supporting autoscaling, containerization (Docker), and CI/CD pipelines (e.g., GitHub Actions, Jenkins).

Priority: Medium

Acceptance Criteria:

• Deployments are automatic after successful testing.

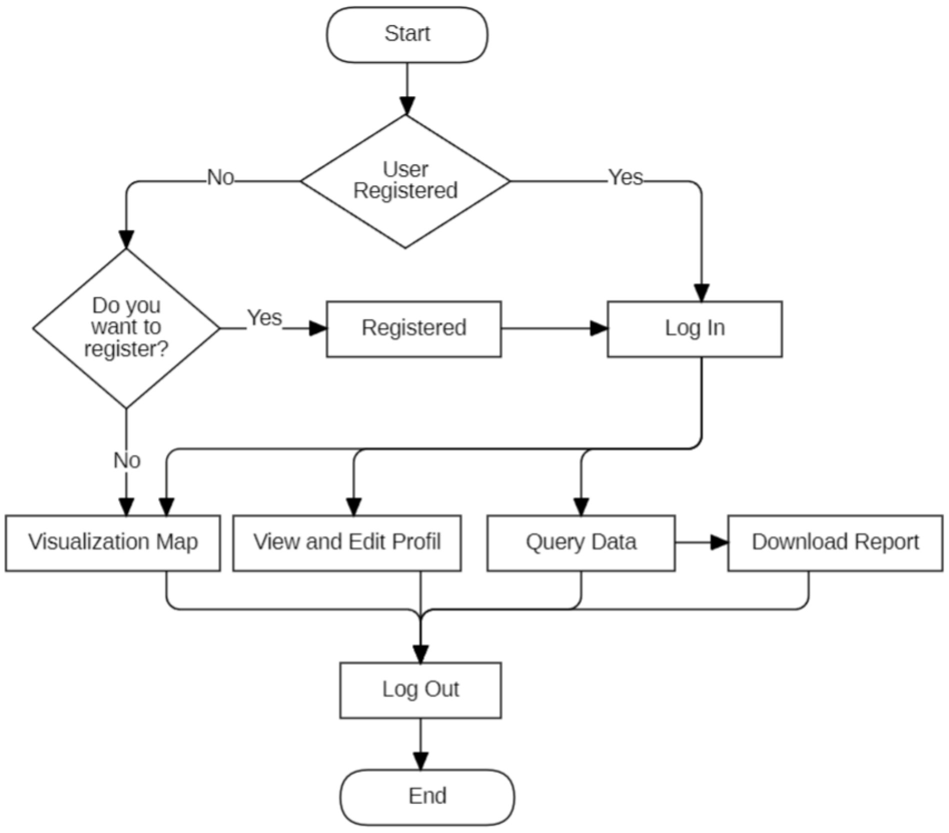
• The system scales based on traffic load.

4 System Model

The process begins with users having the option to either register or proceed without registering. If already registered, users can log in; otherwise, they can continue only with visualization map, they can get an empty map without any data or information. The map with data and information is only open to registered users.

For those who already log in, they can visit their profil page and edit personal information like username, e-mail and so on.

For query data function, users must log in, and then they can query data by city, pollutant, and time. If they want, they can generate reports and download reports for study and analysis.


Start
User Registered
No
Yes
Do you want to register?
Yes
Registered
Log In
No
Visualization Map
View and Edit Profil
Query Data
Download Report
Log Out
End