SE4G 2024-2025

POLITECNICO
MILANO 1863

Design Document and Test Plan

Aether-Analytics

Bingwen Hao

Hangyun Wang

Yujue Wang

Xinchen Yu

1. Introduction

1.1 Purpose

This project aims to develop a client-server application that supports users in querying and visualizing air quality data from the "Dati Lombardia" open data platform. By integrating and processing real-time or historical air quality measurements, the system is designed to provide insights into pollution distribution, temporal trends, and exposure risks. The target users of the platform include researchers, environmental agencies, public health organizations, and the general public interested in monitoring air quality levels.

The system serves as an analytical and visualization tool that supports evidence-based decision-making and promotes environmental awareness. It facilitates users in exploring sensor-based air quality data at multiple geographic scales and over various time spans.

1.2 Scope

The system consists of three major components:

Database (PostgreSQL + PostGIS): Responsible for storing structured data retrieved from the "Dati Lombardia" datasets. This includes sensor metadata, pollutant concentration levels, timestamps, and geographic coordinates.

Web Server (Flask): Serves as the middleware that provides REST API endpoints for querying the database. It handles data preprocessing, filtering, and packaging results into JSON format for client consumption.

Interactive Dashboard (Jupyter Notebooks): A client-side component that allows users to explore the data through interactive maps and graphs. It supports user-generated visualizations and time-series analysis.

The project scope includes:

Air quality and sensor data retrieval and integration

REST API design and backend implementation

Database schema design and spatial indexing

Visualization and analysis dashboard in Jupyter

Out of scope features include advanced predictive modeling, personalized recommendations, and mobile application development.

1.3 Context and Motivation

Environmental health monitoring is a priority for public authorities and citizens alike. With increasing urbanization and industrial activities, real-time monitoring of air

pollutants is crucial. The "Dati Lombardia" portal offers extensive data on air quality from numerous sensors across the region, yet this data is not easily accessible for non-technical users.
This project addresses this gap by transforming complex raw datasets into comprehensible and actionable insights through an intuitive interface. The motivation behind the system lies in enhancing transparency, promoting data-driven policy decisions, and educating the public about environmental conditions in their surroundings.
Furthermore, the system will support spatial and temporal analysis, empowering stakeholders to compare pollution levels across different administrative areas and track changes over time. This contributes to more informed interventions and improved public communication strategies.

1.4 Definitions and Acronyms

Term / Acronym Definition
API Application Programming Interface. Enables communication between systems.
REST Representational State Transfer. A set of architectural principles for web services.
DB Database, typically referring to PostgreSQL in this context.
PostGIS An extension for PostgreSQL that supports geographic objects and spatial queries.
Jupyter Notebook An open-source web application for creating and sharing documents that contain live code, equations, visualizations, and narrative text.
Dati Lombardia The open data portal of the Lombardy region providing environmental datasets, including air quality sensors.

2. System Architecture

The system architecture of the project follows a multi-tiered design aimed at ensuring scalability, maintainability, and efficient communication among components. The architecture is divided into the following key layers:

2.1 Overview

The system adopts a classic three-tier architecture comprising:

- **Presentation Layer**
  This layer handles the user interface and interaction. It is developed as a web-based front end using modern JavaScript frameworks and provides access for different roles such as administrators, auditors, and general users.
- **Application Layer (Business Logic)**

The core logic resides in this layer. It processes user inputs, enforces rules, manages workflows, and interacts with data services. It is implemented using a modular service-based design (microservices or modular monolith), supporting scalability and fault isolation.

- **Data Layer**
  Responsible for persistent data storage and retrieval. It utilizes relational databases (e.g., MySQL or PostgreSQL) and possibly NoSQL databases for unstructured data. It provides APIs for querying, data analytics, and audit trails.

2.2 Components

The system is composed of the following key components:

- **Authentication & Authorization Module**
  Manages user identity and access control based on roles and permissions. Integrates with LDAP or OAuth2 if required.
- **Workflow Engine**
  Orchestrates business processes and approval flows dynamically, allowing customization per organization needs.
- **Monitoring & Logging Module**
  Provides centralized logging, monitoring, and alerting using tools like Prometheus, ELK Stack, or Grafana.
- **API Gateway**
  Acts as a reverse proxy for routing requests to backend services securely and efficiently, handling concerns such as rate limiting, logging, and API versioning.
- **External Integration Module**
  Interfaces with third-party services such as government databases, email/SMS providers, or external data validation services.

2.3 Deployment

The system supports containerized deployment using Docker, and orchestration via Kubernetes or similar platforms, enabling flexible scaling and high availability. CI/CD pipelines are established to ensure rapid and reliable delivery.

2.4 Security Design

Security is embedded at each level:

- HTTPS encryption for all communication
- Role-based access control (RBAC)
- Input validation and sanitization

- Secure token-based session management
- Regular audits and vulnerability scans

3. Software Design

3.3 Dashboard Design (Enhanced)

The dashboard is the central user interface designed to allow stakeholders to explore, analyze, and extract insights from the air quality dataset collected by the Dati Lombardia monitoring network. It is implemented using Jupyter Notebooks and leverages Python's data science ecosystem to create an interactive, user-friendly, and visually rich experience.

Objectives

The primary goals of the dashboard are:

To visualize air quality data in both geographic (map-based) and statistical (chart-based) formats.

To allow users to filter, compare, and interpret trends in pollutant levels across different locations and timeframes.

To provide interactive feedback (warnings, alerts) when data is missing, incomplete, or inconsistent.

To enable custom view generation (e.g., exporting plots, saving selections).

Component Architecture

The dashboard is composed of five main interactive components:

Component	Description
Map Viewer	Displays sensor station locations using Folium or IpyLeaflet. Clicking on a marker shows pollutant readings and metadata. Optionally styled with color-coded pollution levels.

Time-Series Plot	Plots pollutant concentration over time using Plotly, Bokeh, or Matplotlib. Updates dynamically based on user input.
Filter Panel	Built with ipywidgets, includes dropdowns, sliders, date-pickers, and parameter toggles for user-defined queries.
Output Panel	Provides inline text feedback including data status, validation messages, and alerts (e.g., “no data available”).
Export/Save Options	(Optional) Save current chart as PNG or export filtered dataset to CSV for external use.

4. Implementation & Testing

This section outlines the planned development process, testing strategy, and quality assurance approach.

4.1 Implementation Plan

The project follows an Agile-inspired workflow organized in weekly sprints. Tasks are tracked via GitHub Projects or Trello.

Sprint	**Main Activities**
Sprint 1	Setup GitHub repo, Flask project, and PostgreSQL DB schema
Sprint 2	Develop initial data ingestion and preprocessing scripts
Sprint 3	Implement backend APIs and test sample queries

Sprint 4	Create initial dashboard prototype (Jupyter Notebook)
Sprint 5	Integrate maps, widgets, and plots into the dashboard
Sprint 6	Final testing, error handling, export features
Sprint 7	Write SRD + Polish all components for final delivery

4.2 Testing Strategy

We apply unit tests, integration tests, and manual system tests to validate the backend and the dashboard.

Unit Testing (for Flask Backend)

Tool: pytest

Scope: Each API endpoint is tested for:

Correct input validation

Response structure (status 200/400/500)

Data format (valid JSON with expected keys)

Endpoint	Method	Description	Test Case
/api/sensors	GET	Returns list of sensors	Test 200 + array of sensor IDs
/api/data?city=X&param=PM10	GET	Get time-series data	Valid params: return 200; Invalid: return 400
/api/summary	GET	Returns average values	Check keys, values not null

Integration Testing

What is tested: Dashboard–Backend communication

Method: Run Jupyter cell → call Flask endpoint → assert result shape & values

Manual System Testing (End-to-End)

Scenario	Action	Expected Result
Select sensor on map	Click marker	Popup shows sensor details
Filter by date & pollutant	Set filters and submit	Chart updates accordingly
City with no data	Select invalid city	Warning: “No data found”
Backend offline	Reload dashboard	Error shown: “Server unavailable”
Export view	Click download	PNG or CSV saved successfully

5. Bibliography

SE4GEO Project Assignment 2025 Document
Dati Lombardia Air Quality Sensor Datasets:
https://www.dati.lombardia.it/Ambiente/Dati-sensori-aria-dal-2018/g2hp-ar79/about_data
https://www.dati.lombardia.it/Ambiente/Stazioni-qualit-dell-aria/ib47-atvt/about_data
Flask Documentation: https://flask.palletsprojects.com/
PostgreSQL Documentation: https://www.postgresql.org/docs/
PostGIS Documentation: https://postgis.net/
Jupyter Project: https://jupyter.org/
Python Libraries:
Pandas: https://pandas.pydata.org/
GeoPandas: https://geopandas.org/
Plotly: https://plotly.com/
Folium: https://python-visualization.github.io/folium/

IpyLeaflet: https://github.com/jupyter-widgets/ipyleaflet
Matplotlib: https://matplotlib.org/
Bokeh: https://docs.bokeh.org/