Air Quality Monitoring System with Community Validation and Predictive Insights

CS32_4

Divyagnan Reddy (2201130)
Satya Pranav (2201213)
Pranay Satvik Reddy (2201221)
Vasireddy Sriniketh (2201226)

1.Introduction

Air pollution is a growing global issue, and accurate monitoring is essential for public health. The Air Quality Monitoring System (AQMS) enhances existing sources like WAQI by allowing users to flag inaccurate data and enabling trusted contributors called providers to submit real-time sensor readings or validate using different sources . These providers earn credibility scores , ensuring data quality. AQMS also offers personalized alerts based on user-defined health thresholds. By combining official data with community input, AQMS delivers reliable, localized, and engaging air quality insights.

2. Problem Statement

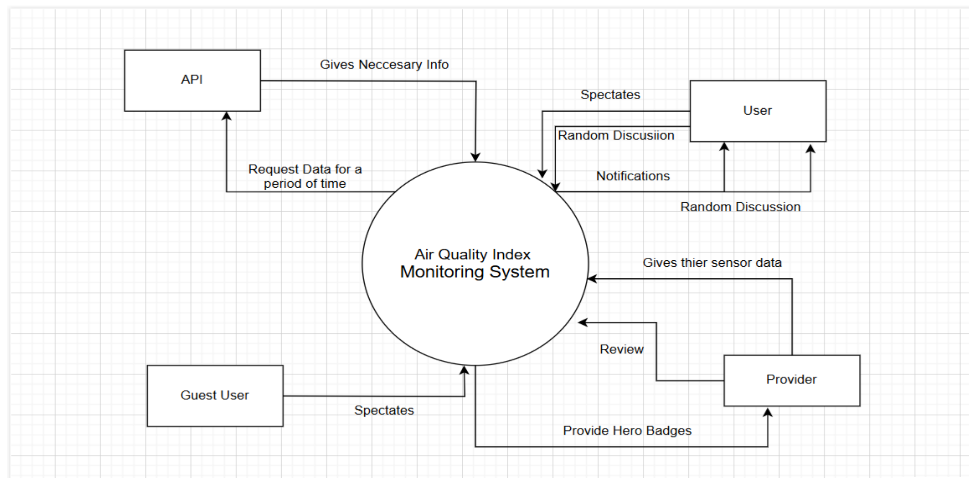
Air pollution has emerged as a critical threat to public health, especially in urban areas where industrial activities and vehicular emissions continue to rise. Existing air quality monitoring tools often rely on centralized data sources with limited flexibility, delayed updates, and lack of personalization.

3. Motivation

The AQMS project aims to deliver a more robust, real-time, and community-driven solution for tracking air quality. By leveraging modern microservices architecture, machine learning techniques, and public participation, AQMS offers enhanced accuracy, responsiveness, and user engagement.

4.Architecture and Business Logic

The AQMS employs a microservices architecture, dividing the system into independent FastAPI-based services for authentication, data ingestion, and notifications. Its scalability allows high-demand components like the Machine Learning (ML) Service to scale independently during peak usage. Each microservice exposes RESTful endpoints for synchronous tasks and uses asyncio for asynchronous loops to schedule operations, such as fetching AQI data every 20 minutes. A PostgreSQL database, accessed via SQLAlchemy ORM, ensures efficient queries for tables like users, measurements, and predictions. For instance, a request for Tokyo's AQI triggers the Measurements service to fetch validated data, while the Notifications service checks thresholds for alerts.



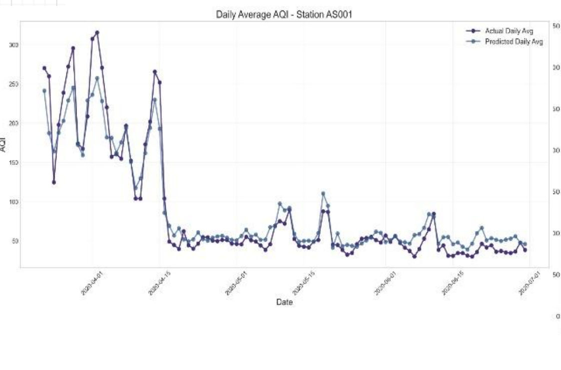
Validation logic ensures data reliability. User inputs, like usernames, are checked for uniqueness, while AQI values are flagged if unrealistic (e.g., PM2.5 > 1000 µg/m³). Contributions are rejected if duplicated, such as a user submitting "PM10 = -5."

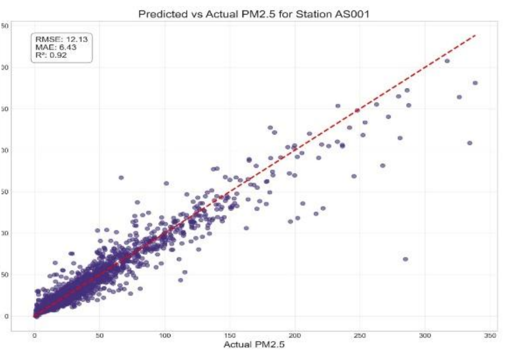
Data transformation converts raw data into JSON (e.g., { "station": "London", "PM2.5": 20 }) for API responses and GeoJSON for maps, with hourly AQI aggregated into daily trends for charts. Rule enforcement restricts contribution approvals to admins, triggers alerts based on user thresholds. Security measures include bcrypt password hashing, rate limiting, and audit logs for admin actions, safeguarding the system.

The Agile Model enabled incremental delivery and continuous user feedback, allowing core features like AQI fetching to launch early and refining tools like custom thresholds based on testing.

5.Novelty

ML: The AQMS employs **XGBoost**, a gradient-boosting algorithm, for accurate 48-hour air quality forecasts. Unlike basic models like linear regression, XGBoost captures complex environmental patterns, predicting pollutant levels (e.g., PM2.5, NO_2). Trained weekly on historical data from the measurements table, it generates hourly and daily predictions stored in the predictions table. Computational efficiency enables real-time forecasts, empowering users to plan activities or take health precautions, such as avoiding outdoor exercise during pollution spikes.





Contributor verification mechanism: enhances data reliability by combining API-sourced data with community input. To address discrepancies in AQI readings across platforms, verified contributors, vetted via Google Forms with gold standard questions, submit corrections (e.g., adjusting an outdated NO_2 reading in Delhi). Admins cross-check submissions, ensuring accuracy and transparency. This hybrid approach improves data trust, engages communities, and scales coverage to regions with limited API access.

Custom preferences: redefine personalization, allowing users to select global monitoring stations, set pollutant thresholds (e.g., PM2.5 > 35 µg/m³), track multiple locations. Stored in the user_preferences table and processed asynchronously, this feature supports diverse use cases, from parents monitoring school air quality to runners tracking ozone levels. It transforms passive data consumption into active engagement.

6.Tech Stack

Backend: FastAPI with Pydantic for APIs, APScheduler for tasks.
Database: PostgreSQL with SQLAlchemy and Alembic
Frontend: React.js with React Router, CSS Modules, and Browser Notifications.
Communication: CORS, Fetch API, and JWT for secure interaction.

7.Conclusion

The Air Quality Monitoring System (AQMS) is a powerful tool that combines smart technology with user-friendly design to track and improve air quality. It uses XGBoost-based predictions to help users make informed decisions and relies on community feedback to ensure data accuracy. Built on a scalable microservices architecture, AQMS can grow and perform well even with many users. It provides timely, personalized air quality alerts to support public health and environmental awareness.