SYSTEM DESIGN AND DEVELOPMENT REPORT

Submitted by
Cenex Consult Limited
to
Clean Air Fund

Project Title: Nairobi City-Owned Air Quality Data Management System (AQDMS) & Public Data Portal

DOCUMENT NUMBER	RELEASE/REVISION NUMBER	RELEASE/REVISION DATE
3	V:01	31st October 2025
3	V:02	05 November 2025

Executive Summary

The Nairobi City-Owned Air Quality Data Management System (AQDMS) and Public Data Portal represent a major step forward in Nairobi's environmental governance and digital infrastructure. Developed by Cenex Consult Limited under the Clean Air Fund's Breathe Cities initiative, the system was designed to provide real-time air quality monitoring, data transparency, and actionable insights for both government officials and the general public.

At its core, the AQDMS is a modular, scalable platform built using open-source technologies including Next.js, Node.js, Express.js, and PostgreSQL. It integrates data from low-cost sensors deployed across Nairobi City County, enabling automated data ingestion, validation, and visualization. The system features two tailored interfaces: a County Management Dashboard for strategic planning and policy development, and a Public Data Portal that offers simplified, mobile-friendly access to air quality information.

The development process followed a phased approach, beginning with stakeholder consultations and system design, followed by software and database development, API creation, and iterative testing. Throughout, county ICT staff, environmental experts, and community representatives were actively engaged to ensure the system met both technical standards and user expectations. Key functionalities include role-based access control, threshold-based alerting, interactive maps, and downloadable reports in multiple formats.

Performance testing confirmed the system's ability to handle high-frequency sensor data and deliver responsive dashboards. Security protocols such as JWT authentication and SSL encryption were implemented to safeguard data integrity and user privacy. The API layer supports seamless integration with external platforms, ensuring future interoperability and regional data sharing.

Ultimately, the AQDMS delivers a robust, user-centric solution that empowers Nairobi City County to monitor air quality in real time, inform policy with reliable data, and engage the public in environmental awareness. It lays the groundwork for a healthier, more transparent, and digitally empowered Nairobi.

Acknowledgement

Government ICT and Environment teams, whose technical guidance and institutional support were instrumental throughout the design, development, and deployment of the Nairobi City-Owned Air Quality Data Management System (AQDMS) and Public Data Portal. Their commitment to digital innovation and environmental stewardship laid the foundation for a system that advances public health, transparency, and data-driven governance.

We especially acknowledge the Clean Air Fund for their strategic partnership and unwavering advocacy under the Breathe Cities initiative. Their role in convening stakeholders, facilitating technical dialogue, and championing open data principles has been central to the success of this project. The collaboration with C40 Cities and Bloomberg Philanthropies further enriched the initiative with global insights and policy coherence.

Our gratitude also goes to the environmental experts and community stakeholders who contributed their domain knowledge and local perspectives during system validation and usability testing. Their feedback on sensor calibration, data interpretation, and public-facing features helped shape a platform that is both technically robust and socially inclusive.

We recognize the citizen testers and public users whose engagement during the feedback and optimization phases ensured that the system remains accessible, intuitive, and responsive to community needs. Their participation reflects a growing civic commitment to environmental awareness and digital empowerment.

The shared dedication to a cleaner, healthier Nairobi was evident in the collaborative spirit, technical rigor, and forward-thinking contributions that guided this project from concept to deployment. As the AQDMS continues to evolve, your insights will remain vital in fostering inclusive governance, environmental resilience, and sustainable urban development.

Contents

1. Introduction

1.1. Air Quality Data Management System (AQDMS) – Strategic Overview

The **Air Quality Data Management System (AQDMS)** is a comprehensive digital infrastructure developed to support the Nairobi City County Government in collecting, processing, storing, and disseminating air quality data. It addresses the urgent need for **real-time environmental monitoring**, offering both **public-facing access** to air quality insights and **advanced analytics tools** for county officials to inform policy and decision-making.

This initiative is part of the broader **Breathe Cities program**, led by the **Clean Air Fund**, **C40 Cities**, and **Bloomberg Philanthropies**, which empowers cities globally to reduce air pollution and greenhouse gas emissions. Through a combination of data-driven research, stakeholder engagement, technical policy support, and knowledge sharing, Breathe Cities fosters sustainable urban air quality solutions. In Kenya, the program is actively collaborating with Nairobi City County to develop a localized AQDMS that reflects the city's unique environmental challenges while aligning with both **national and international air quality standards**.

The AQDMS integrates data from **multiple sensor networks**, including **low-cost sensors owned by Nairobi City County Government (NCCG)**, to ensure accurate and timely reporting. The system's architecture features automated data pipelines and modular dashboards that deliver real-time visualizations, trend analysis, and alerts. These dashboards are tailored for two key user groups:

- **County Officials**: For strategic planning, environmental reporting, and policy development.
- **Public Users**: For accessible, transparent, and actionable air quality information.

Together with its **Public Data Portal**, the AQDMS provides a robust, scalable, and user-friendly platform that enhances **government transparency**, **community awareness**, and **data-driven environmental governance** across Nairobi City County.

1.2. User Groups and System Use Cases

The Air Quality Data Management System (AQDMS) is designed to serve two distinct but interconnected user groups: **county officials** and **public users**. Each group interacts with the system through tailored interfaces and functionalities that reflect their unique needs, responsibilities, and levels of technical expertise.

1.2.1. County Officials

Primary Objectives

- Monitor air quality trends across Nairobi City County.
- Generate reports for environmental planning and compliance.
- Inform policy decisions and urban development strategies.
- Respond to pollution alerts and coordinate mitigation efforts.

Key Features for County Use

- **Administrative Dashboard**

For county officials, the AQDMS offers a robust administrative dashboard that delivers real-time sensor data, historical trends, and hotspot mapping. This dashboard was built using Next.js and Express.js, with PostgreSQL powering the backend. Officials can visualize pollution patterns across Nairobi using dynamic maps powered by Leaflet.js, where color-coded indicators reflect current AQI levels. This feature was included to support strategic planning and rapid response to environmental hazards.

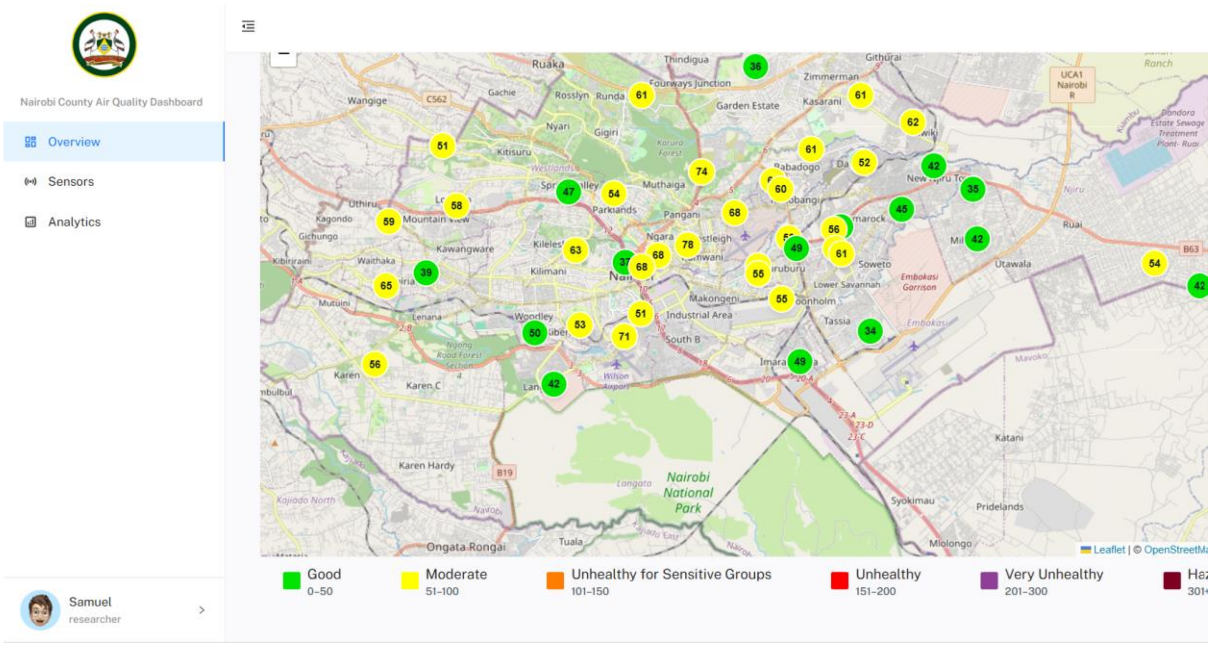
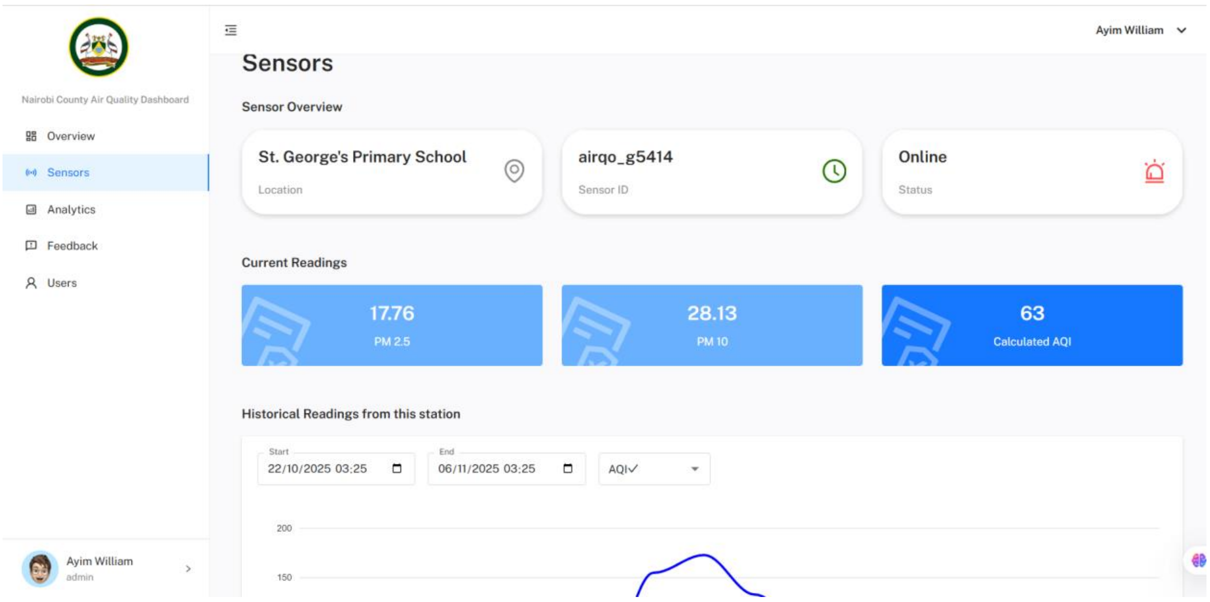


Fig. 1: Snapshot of the administrative dashboard during the testing phase, displaying mapped sensor locations alongside active analytics tabs for data filtering, trend visualization, and alert monitoring.



Sensors

Sensor Overview

St. George's Primary School
Location

airqo_g5414
Sensor ID

Online
Status

Current Readings

17.76
PM 2.5

28.13
PM 10

63
Calculated AQI

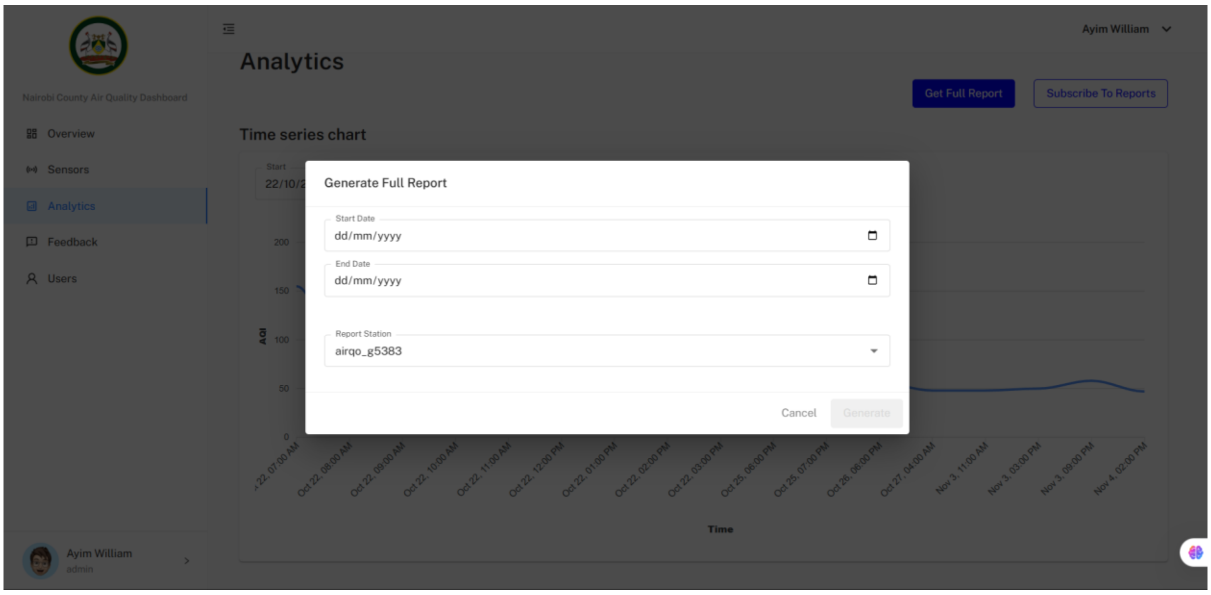
Historical Readings from this station

Start 22/10/2025 03:25
End 06/11/2025 03:25
AQI✓

Fig. 2: This is a snapshot of the sensors section of the dashboard that gives specification information for a chosen sensor, allowing the user to see the past historical trends.

- **Custom Reporting Tools**

To complement the dashboard, custom reporting tools allow officials to export filtered data in PDF, CSV, or JSON formats. These reports are generated through backend endpoints using Puppeteer for PDFs and json2csv for CSVs. This functionality ensures that environmental officers and analysts can prepare documentation for internal use or stakeholder presentations with ease and precision.



Analytics

Generate Full Report

Start Date dd/mm/yyyy

End Date dd/mm/yyyy

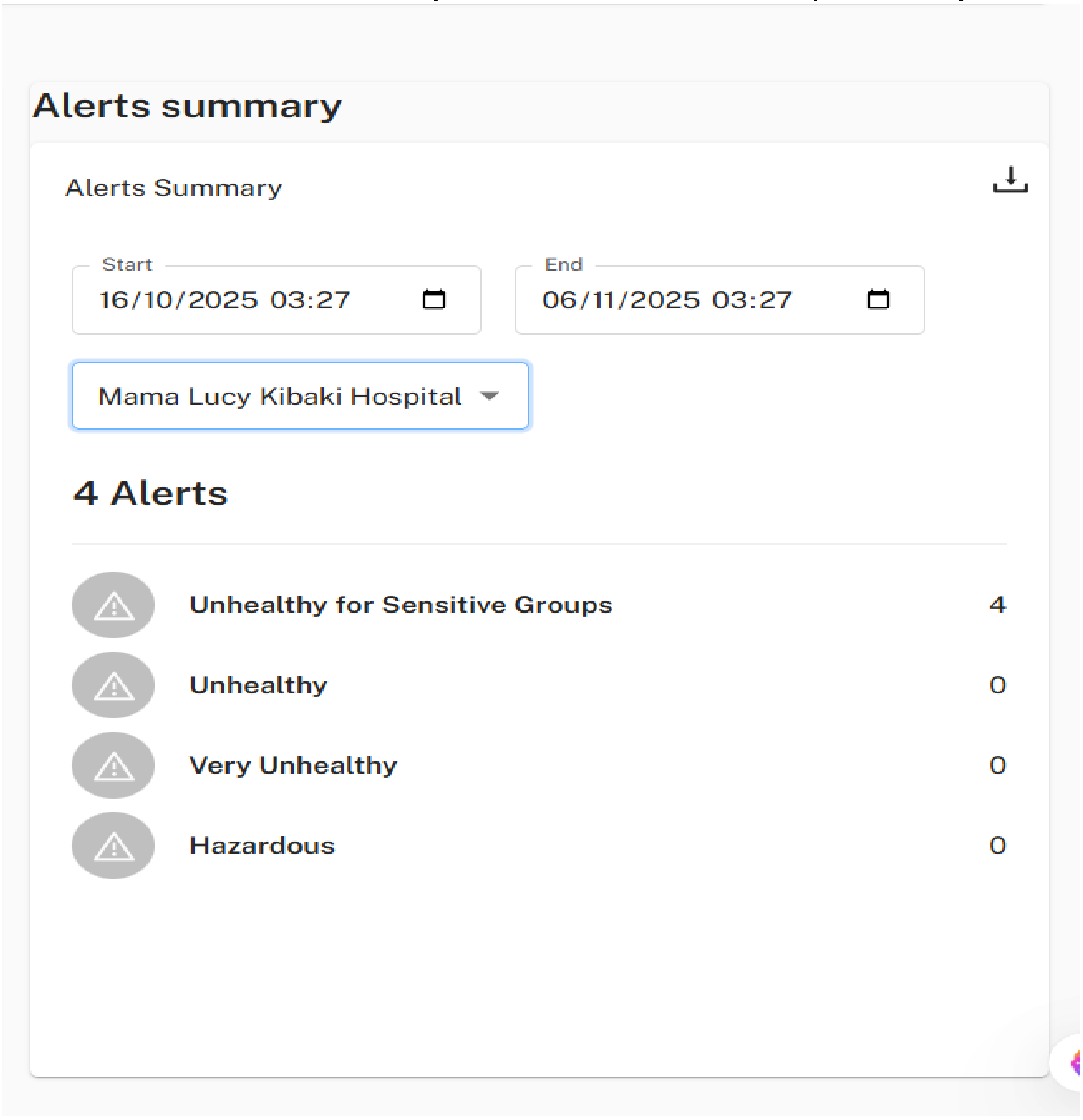
Report Station airqo_g5383

Figure 3: Sensor-Specific Report Generation: This figure illustrates the AQDMS platform's capability to generate comprehensive reports for individual sensors upon user request. The feature enables county analysts and administrators to retrieve historical data, performance metrics, and anomaly logs for any deployed sensor. This

supports targeted diagnostics, maintenance planning, and localized environmental analysis.

- **Alert Management System**:

Another critical feature is the alert management system, which automatically notifies officials when pollution levels exceed predefined thresholds. The system uses a backend algorithm that compares incoming sensor data against WHO and NEMA standards. When a threshold is breached, alerts are logged, queued, and dispatched via email. This mechanism ensures timely interventions and enhances public safety**.**



Alerts summary

Alerts Summary

Start: 16/10/2025 03:27

End: 06/11/2025 03:27

Mama Lucy Kibaki Hospital

4 Alerts

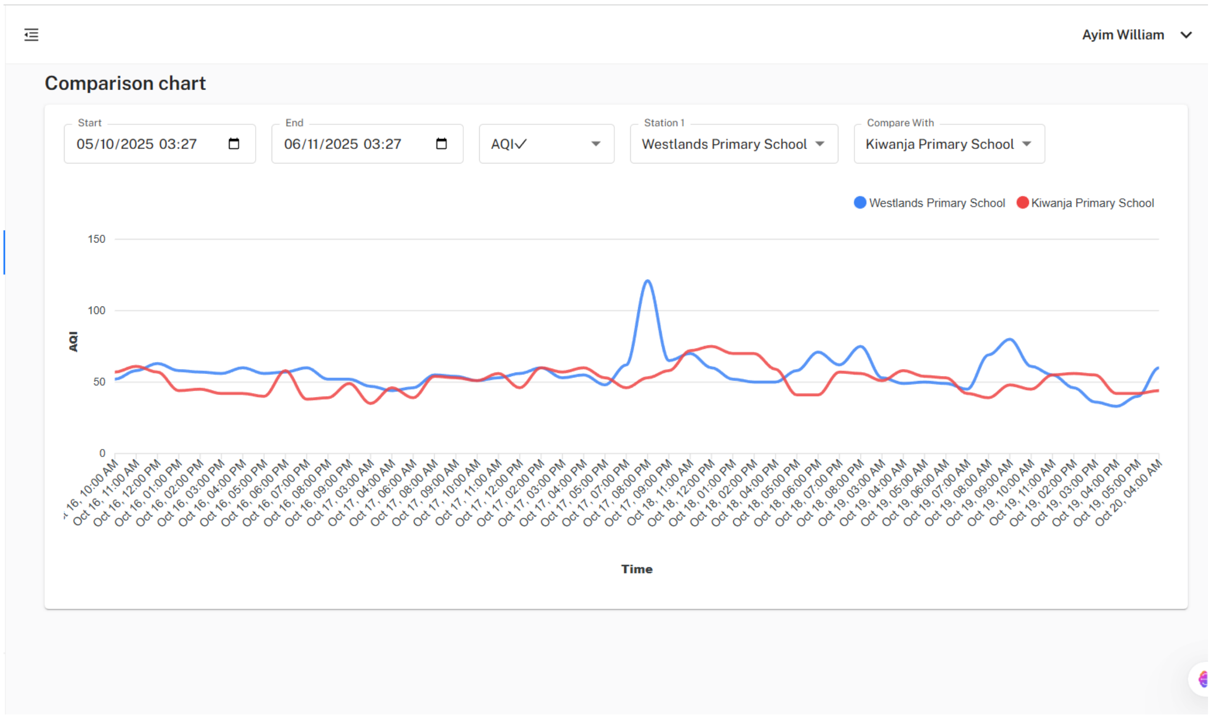
Unhealthy for Sensitive Groups	4
Unhealthy	0
Very Unhealthy	0
Hazardous	0

Figure 4: Sensor-Level Alert History View. This figure showcases the AQDMS platform's ability to filter and display alerts received by a specific sensor over a defined time period. Users—particularly county analysts—can inspect threshold breaches, anomaly flags, and system-generated warnings tied to individual sensors.

This feature supports localized diagnostics, historical pattern analysis, and proactive maintenance planning.

- **Data Filtering & Analysis**

To support deeper analysis, the dashboard includes data filtering tools that allow users to compare stations, timeframes, and pollutant types. These filters are implemented through dynamic queries using Prisma ORM and visualized with Chart.js and D3.js. . This empowers county planners and environmental experts to identify trends and make data-driven decisions.



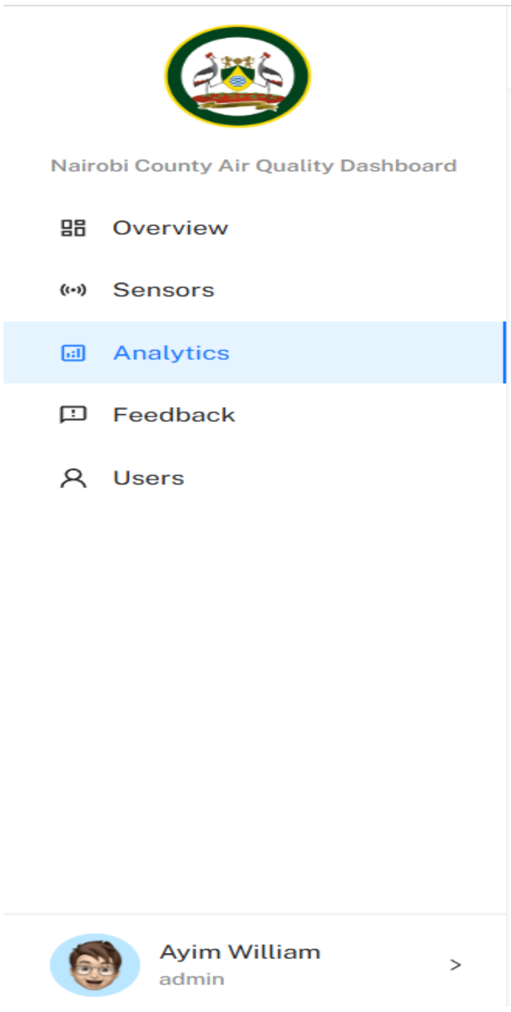
Ayim William
Comparison chart
Start: 05/10/2025 03:27
End: 06/11/2025 03:27
AQI✓
Station 1: Westlands Primary School
Compare With: Kiwanja Primary School
● Westlands Primary School ● Kiwanja Primary School
AQI
Time

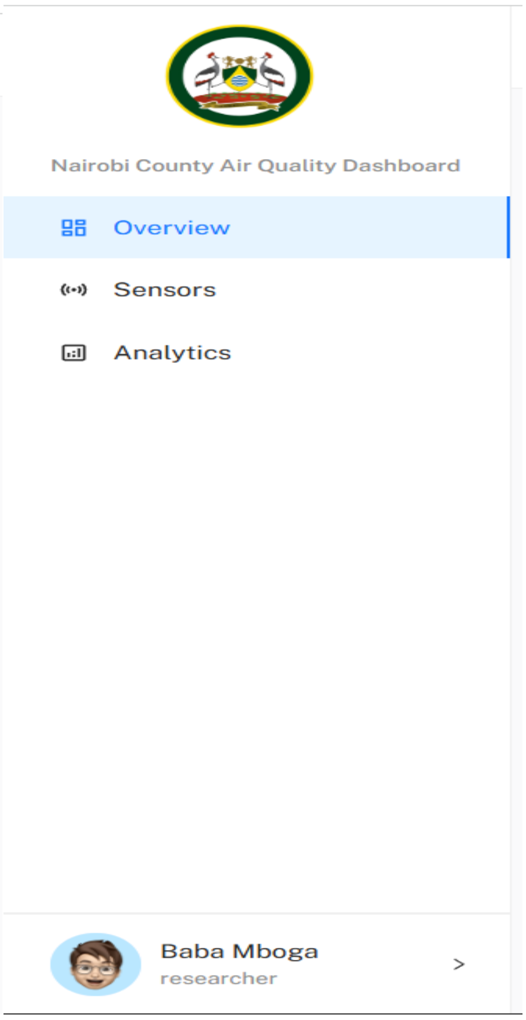
Figure 5: Comparative Sensor Analysis Between Stations

This figure illustrates the AQDMS platform's capability to compare air quality readings between two sensor stations over a defined time period. Users—particularly county analysts and researchers—can select specific sensors and time ranges to visualize differences in pollutant levels, detect anomalies, and assess spatial trends. This feature supports data-driven decision-making for localized interventions and policy development.

- **User Management**

Finally, the user management module ensures secure access control across the system. Using JWT authentication and role-based authorization, the system allows ICT administrators to assign permissions to different user roles—such as analysts, planners, and public health officials—ensuring that sensitive data is only accessible to authorized personnel.





Figures 6 & 7: Tiered Access Levels within the AQDMS Dashboard

These figures illustrate the implementation of Role-Based Access Control (RBAC) within the AQDMS County Dashboard. Users are granted access to specific modules and functionalities based on their assigned roles—such as administrator, analyst, or public viewer. For example, administrators can manage users and configure alerts, while analysts access sensor analytics and reporting tools. Public users are restricted to viewing aggregated data via the Public Data Portal.

This tiered access structure ensures that sensitive operations and data remain protected, while promoting transparency and usability across stakeholder groups.

- **Typical Users**
  - Environmental officers
  - ICT administrators
  - Urban planners

- Public health officials
- Policy advisors

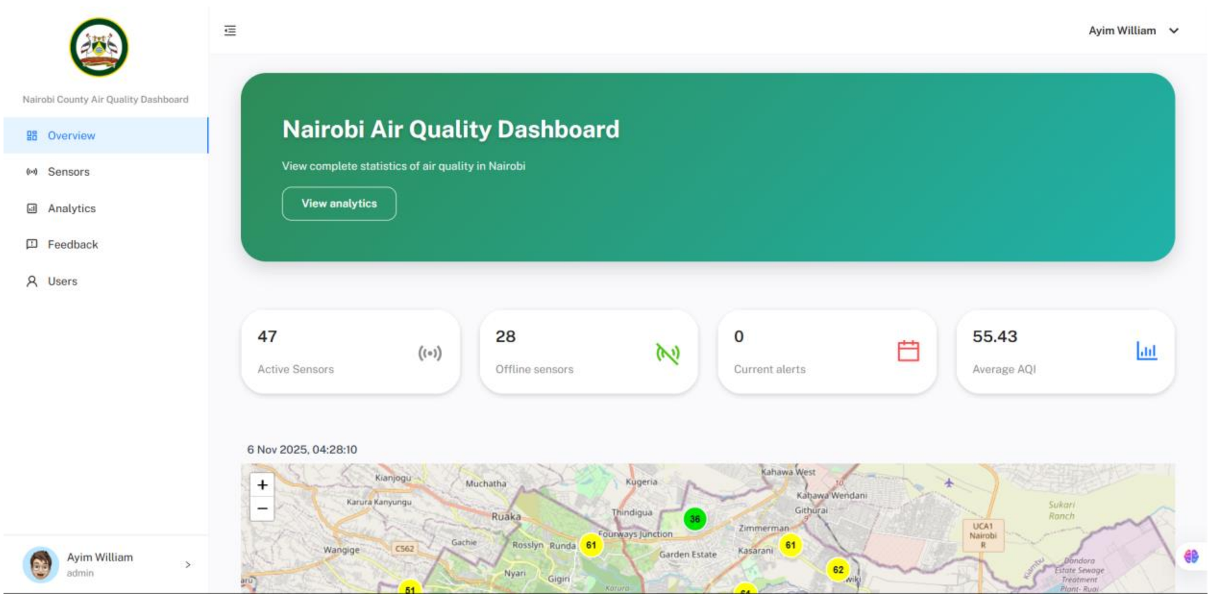


Figure 8: Admin Landing Page of the AQDMS Dashboard

This figure presents the default landing interface displayed to county administrators upon logging into the AQDMS platform. It provides immediate access to key modules such as Overview, Sensors, Analytics, Feedback, and User Management. The layout reinforces role-based access control, ensuring that administrative users can efficiently navigate system functions, monitor air quality data, and manage operational workflows.

.

1.2.2. Public Users

Primary Objectives

- Access real-time air quality information for personal health and safety.
- Understand pollution levels in their neighborhood or workplace.
- Participate in community awareness and advocacy efforts.

Key Features for Public Use

- **Public Data Portal**

For Nairobi's residents, the AQDMS provides a mobile-friendly Public Data Portal designed for accessibility and ease of use. Built with Next.js and HTMX, the portal delivers real-time air quality updates through a clean, responsive interface. This feature was included to democratize access to environmental data and promote health-conscious decision-making.

- **Interactive Maps**

Interactive maps are a centerpiece of the portal, displaying geolocated sensor data with intuitive color-coded AQI indicators. These maps use Leaflet.js to render real-time readings from the city's sensor network, allowing users to understand pollution levels in their neighborhoods or workplaces. Tooltips provide additional context, including pollutant breakdowns and timestamps.

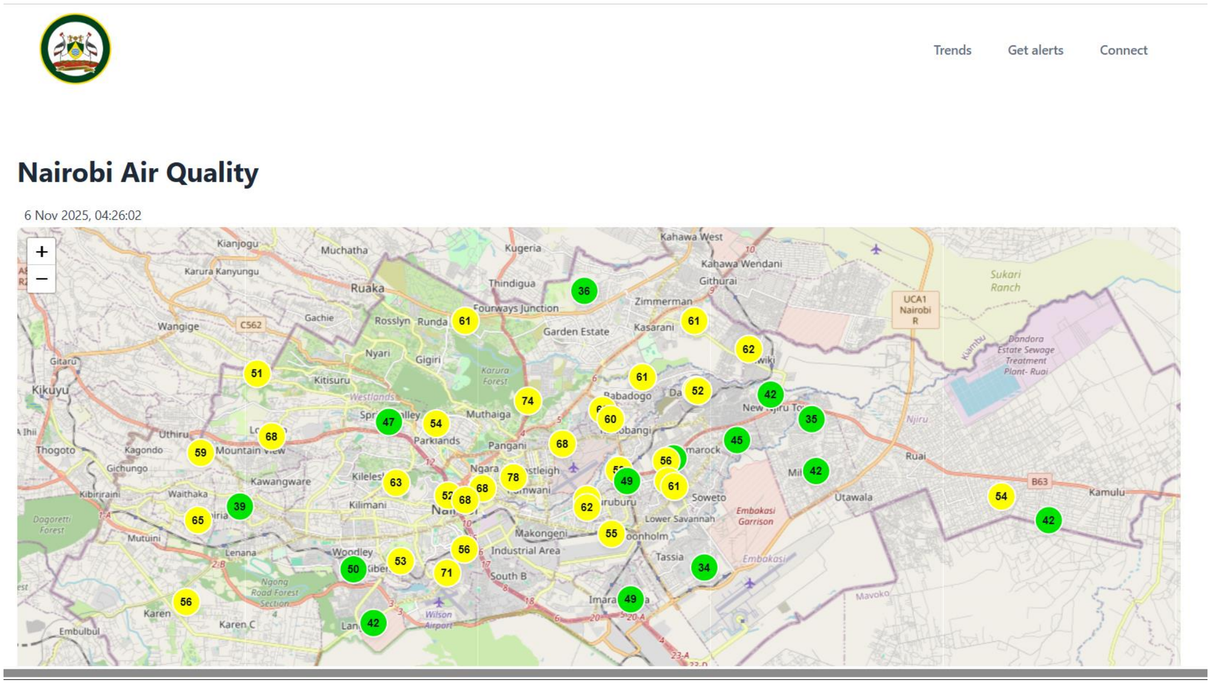
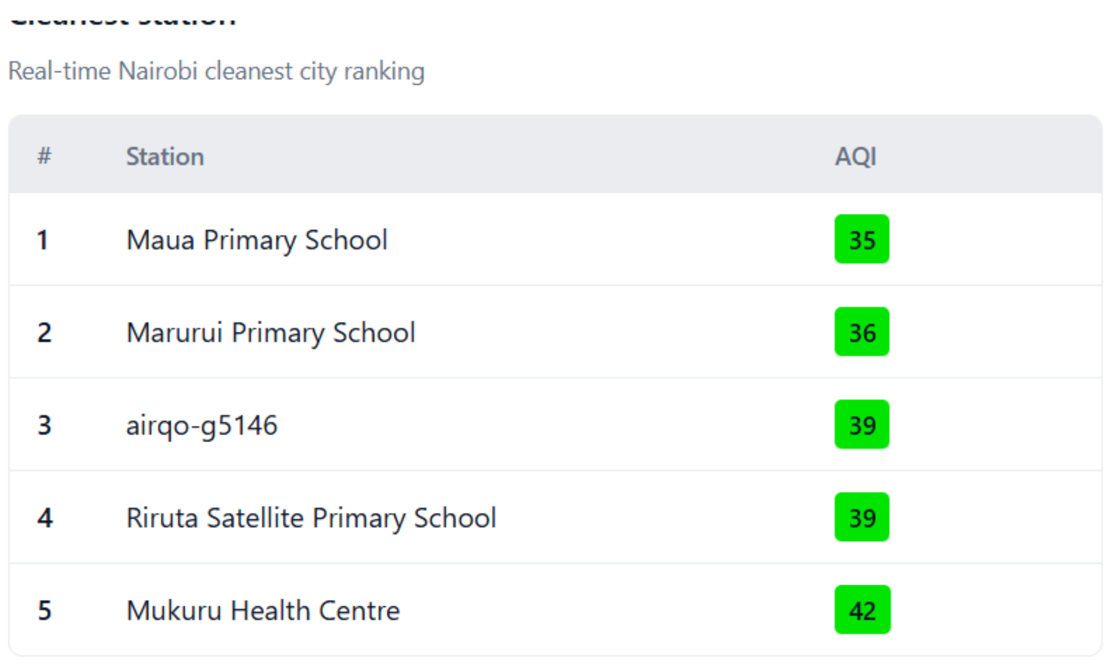


Fig. 9: Interactive sensor map displayed on the public portal, showing active air quality sensors across Nairobi. Users can view real-time metrics—including AQI levels—for specific regions, enabling localized environmental awareness and data exploration.

- **Daily Summaries**

To help users make informed daily choices, the portal offers daily summaries that aggregate hourly data into easy-to-read cards. These summaries include health advisories based on AQI levels, generated using predefined mappings aligned with WHO guidelines. Whether commuting, exercising, or caring for children, users can quickly assess air quality conditions.



[illegible]

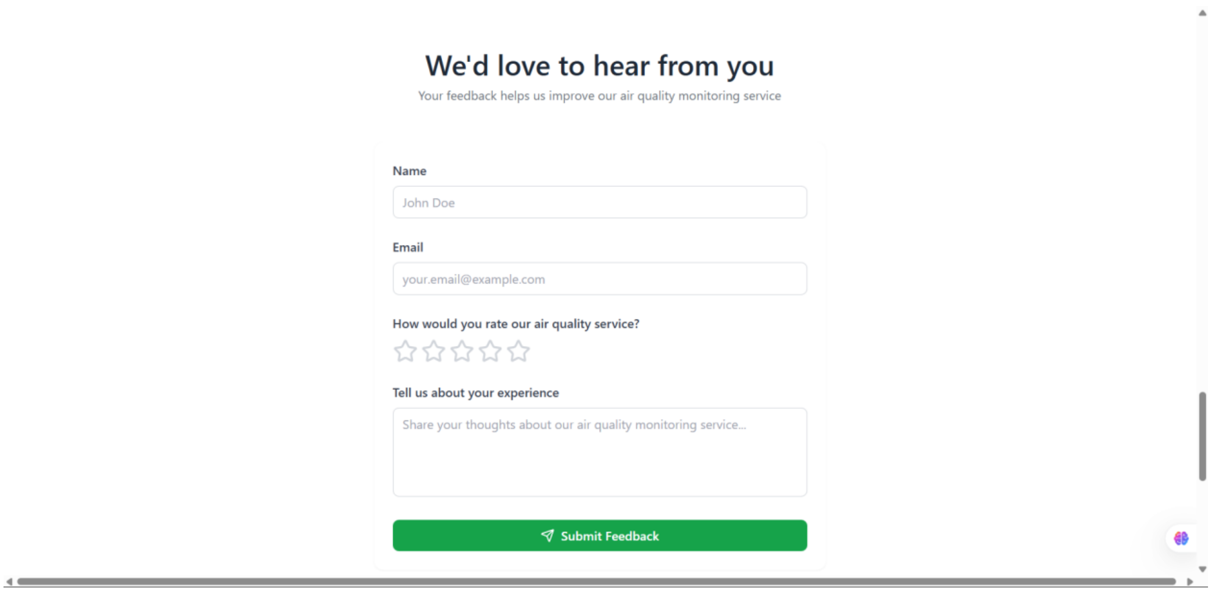
Real-time Nairobi cleanest city ranking

#	Station	AQI
1	Maua Primary School	35
2	Marurui Primary School	36
3	airqo-g5146	39
4	Riruta Satellite Primary School	39
5	Mukuru Health Centre	42

Figure 10: Real-Time Station Ranking Snapshot
This figure displays the AQDMS platform's real-time ranking of air quality monitoring stations across Nairobi. Stations are ranked based on pollutant levels, sensor reliability, or alert frequency within a defined time window. This feature enables county officials and analysts to quickly identify hotspots, prioritize interventions, and communicate environmental risks to the public.

- **Feedback Mechanism**

The portal also includes a feedback mechanism that allows users to submit comments or concerns directly to county officials. Submissions are stored in the system's Feedback table and can be reviewed by administrators through a dedicated dashboard. This feature fosters civic engagement and helps improve system usability based on real-world input.


We'd love to hear from you

Your feedback helps us improve our air quality monitoring service

Name

John Doe

Email

your.email@example.com

How would you rate our air quality service?

☆☆☆☆☆

Tell us about your experience

Share your thoughts about our air quality monitoring service...

Figure 11: Public Feedback Form on the AQDMS Portal
This figure displays the feedback interface available to users on the Nairobi Air Quality Public Data Portal. It allows citizens to submit comments, suggestions, and improvement ideas directly to the County Government. This feature supports civic engagement, promotes transparency, and enables iterative system enhancement based on real-world user input.

- **Educational Resources**:

Lastly, educational resources are embedded throughout the portal to raise awareness about air pollution and its health impacts. These resources include infographics and guides tailored for low-literacy audiences and mobile users, ensuring that environmental education is accessible to all.

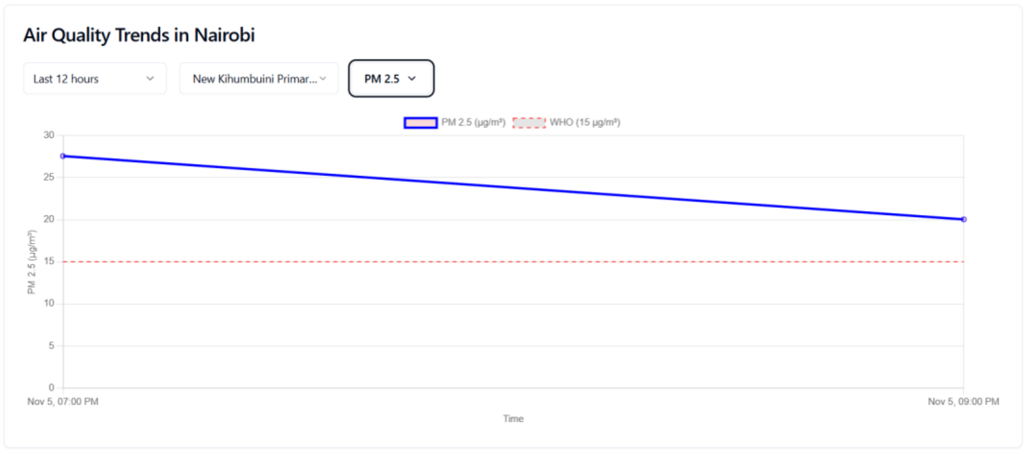


Figure 12: Public Graph Interface for Sensor Data Exploration

This figure highlights the user-friendly graphing feature available on the AQDMS Public Data Portal. Public users can select a specific sensor and time range to visualize air quality trends. The graph includes clearly labeled axes, a dynamic legend, and intuitive navigation controls—enabling citizens to explore environmental data with clarity and confidence. This feature supports public awareness, data literacy, and informed community engagement.

Typical Users

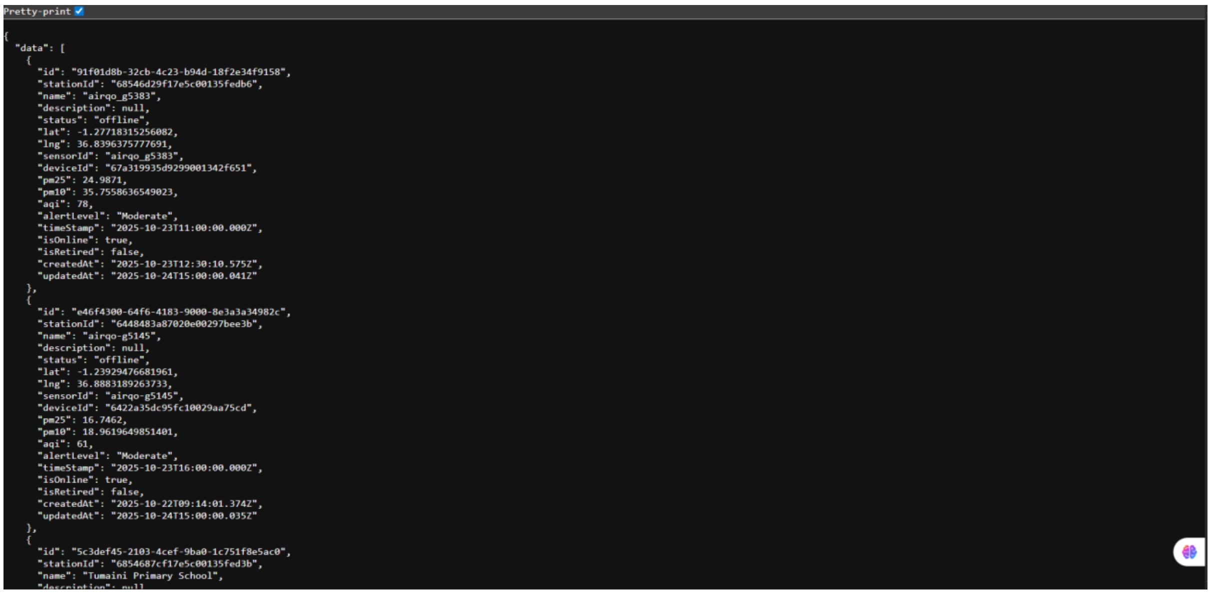
- Nairobi residents
- Commuters and cyclists
- Parents and schools
- Health-conscious individuals
- Environmental advocates

.

1.2.3. Cross-Group Synergy

While the system offers distinct interfaces for each group, it also fosters collaboration and transparency:

- **Shared Data Backbone**: Both dashboards pull from the same validated sensor network and database.



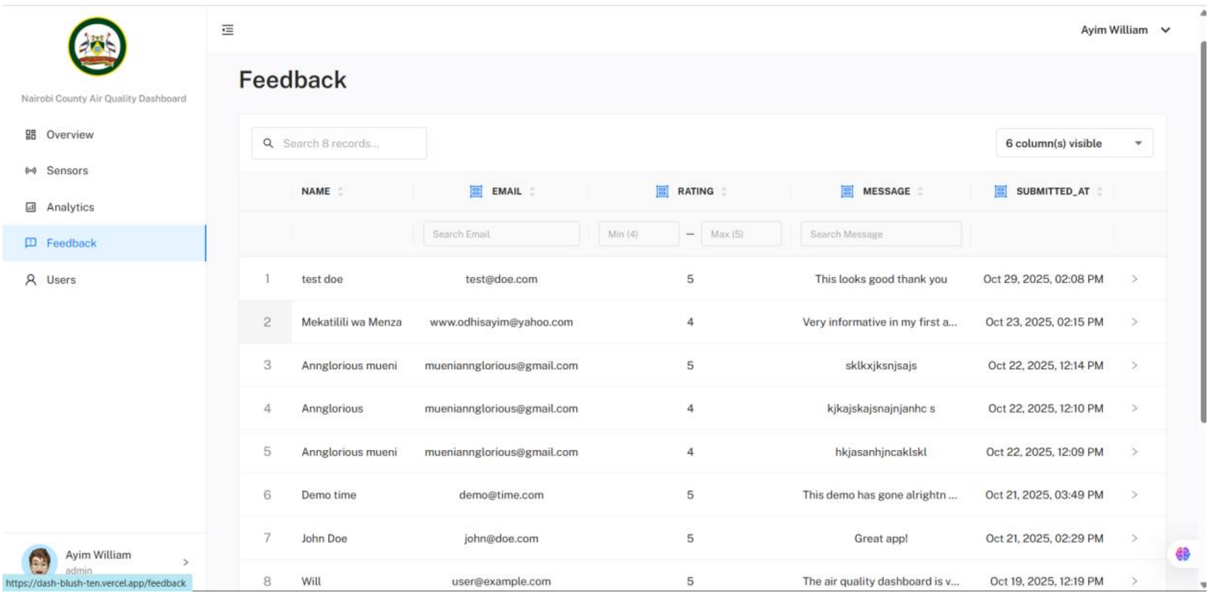
Pretty-print ✓

```
{
  "data": [
    {
      "id": "91f01d8b-32cb-4c23-b94d-18f2e34f9158",
      "stationId": "68546d29f17e5c00135fedb6",
      "name": "airqo_g5383",
      "description": null,
      "status": "offline",
      "lat": -1.27718315256082,
      "lng": 36.8396375777691,
      "sensorId": "airqo_g5383",
      "deviceId": "67a319935d9299001342f651",
      "pm25": 24.9871,
      "pm10": 35.7558636549023,
      "aqi": 78,
      "alertLevel": "Moderate",
      "timeStamp": "2025-10-23T11:00:00.000Z",
      "isOnline": true,
      "isRetired": false,
      "createdAt": "2025-10-23T12:30:10.575Z",
      "updatedAt": "2025-10-24T15:00:00.041Z"
    },
    {
      "id": "e46f4300-64f6-4183-9000-8e3a3a34982c",
      "stationId": "6448483a87020e00297bee3b",
      "name": "airqo-g5145",
      "description": null,
      "status": "offline",
      "lat": -1.23929476681961,
      "lng": 36.8883189263733,
      "sensorId": "airqo-g5145",
      "deviceId": "6422a35dc95fc10029aa75cd",
      "pm25": 16.7462,
      "pm10": 18.9619649851401,
      "aqi": 61,
      "alertLevel": "Moderate",
      "timeStamp": "2025-10-23T16:00:00.000Z",
      "isOnline": true,
      "isRetired": false,
      "createdAt": "2025-10-22T09:14:01.374Z",
      "updatedAt": "2025-10-24T15:00:00.035Z"
    },
    {
      "id": "5c3def45-2103-4cef-9ba0-1c751f8e5ac0",
      "stationId": "6854687cf17e5c00135fed3b",
      "name": "Tumaini Primary School",
```

Figure 13: Sample Data Snapshot Shared Across Portal and Dashboard

This figure presents a snapshot of the air quality data ingested and visualized by both the AQDMS Public Data Portal and County Dashboard. The shared dataset includes timestamped sensor readings, location metadata, and pollutant concentrations (e.g., PM2.5, PM10). This unified data stream ensures consistency across public-facing *insights and administrative analytics, supporting transparency, operational planning, and real-time environmental monitoring.*

- **Public Feedback Loop**: Insights from public users help county officials refine policies and improve system usability.



	NAME	EMAIL	RATING	MESSAGE	SUBMITTED_AT
1	test doe	test@doe.com	5	This looks good thank you	Oct 29, 2025, 02:08 PM
2	Mekatilili wa Menza	www.odhisayim@yahoo.com	4	Very informative in my first a...	Oct 23, 2025, 02:15 PM
3	Annglorious mueni	mueniannglorious@gmail.com	5	sklkxjksnjsajs	Oct 22, 2025, 12:14 PM
4	Annglorious	mueniannglorious@gmail.com	4	kjkajskajsnajnjanhc s	Oct 22, 2025, 12:10 PM
5	Annglorious mueni	mueniannglorious@gmail.com	4	hkjasanhjncaklskl	Oct 22, 2025, 12:09 PM
6	Demo time	demo@time.com	5	This demo has gone alrightn ...	Oct 21, 2025, 03:49 PM
7	John Doe	john@doe.com	5	Great app!	Oct 21, 2025, 02:29 PM
8	Will	user@example.com	5	The air quality dashboard is v...	Oct 19, 2025, 12:19 PM

Figure 14: County Dashboard Feedback Section
This figure shows the feedback interface accessible to county officials within the AQDMS administrative dashboard. It aggregates user-submitted comments and suggestions from the Public Data Portal, allowing officials to review, categorize, and respond to public input. This feature strengthens civic engagement, supports iterative system improvements, and ensures that environmental governance remains responsive to community needs.

- **Transparency Tools**: Open access to selected datasets builds trust and encourages civic engagement.

1.3. Project Goals

The development of the Air Quality Data Management System (AQDMS) is guided by a set of strategic goals that align with Nairobi City County's environmental priorities and the broader objectives of the Breathe Nairobi initiative. These goals ensure the system is technically robust, socially impactful, and future-ready.

1.3.1. Scalable and Modular Architecture

- Design a system that can seamlessly accommodate future expansion, including additional sensor networks, new data types, and evolving user needs.

- Implement modular components for data ingestion, processing, visualisation, and alerting to support independent upgrades and maintenance.
- Ensure compatibility with both on-premise and cloud-based deployments for long-term flexibility.

1.3.2. Data Accuracy, Reliability, and Accessibility

- Integrate validated sensor networks and automated data pipelines to minimize errors and ensure consistent data quality.
- Apply rigorous testing protocols to verify ingestion speed, query responsiveness, and dashboard performance.
- Provide secure, role-based access to ensure appropriate data visibility for different user groups.

1.3.3. Public Awareness and Government Transparency

- Empower Nairobi residents with real-time, easy-to-understand air quality information through a public-facing data portal.
- Support evidence-based decision-making by county officials with advanced analytics and reporting tools.
- Foster civic engagement and trust by making environmental data openly available and actionable.

1.3.4. Alignment with National and International Standards

- Ensure system design and data protocols comply with Kenya's environmental regulations and global air quality frameworks.
- Facilitate interoperability with regional and international platforms for broader data sharing and benchmarking.

1.4. Stakeholder Roles and Responsibilities

The successful development and deployment of the Air Quality Data Management System (AQDMS) relied on close collaboration among multiple stakeholder groups. Each played a distinct and complementary role in ensuring the system's technical robustness, environmental accuracy, and public usability.

1.4.1. Nairobi City County ICT Team

Role: Oversight, Infrastructure, and Deployment

- ➢ Facilitated integration with county digital infrastructure and ensured compliance with ICT policies.
- ➢ Oversaw system deployment, hosting, and long-term maintenance planning.

- Provided strategic direction and governance throughout the project lifecycle.

System Developers

Role: Software Architecture, API Design, and Implementation

- Designed and implemented the modular system architecture, including frontend dashboards, backend services, and database schemas.
- Developed RESTful APIs to enable real-time data ingestion, retrieval, and integration with external systems.
- Conducted iterative testing and optimization to ensure performance, scalability, and security.

1.4.2. Environmental Experts

Role: Data Validation and Domain Guidance

- Provided technical input on air quality indicators, sensor calibration, and data interpretation.
- Validated the accuracy and reliability of sensor data and analytical outputs.
- Ensured alignment with national environmental standards and international best practices.

1.4.3. Public Users

Role: Usability Testing and Community Feedback

- Participated in user testing sessions to evaluate the clarity, accessibility, and relevance of the public data portal.
- Offered feedback on dashboard design, data presentation, and user experience.
- Helped shape the system's public-facing features to ensure inclusivity and community engagement.

1.5. Project Timeline Overview

The development of the Air Quality Data Management System (AQDMS) followed a structured, phased approach to ensure clarity, stakeholder alignment, and technical excellence. Each phase was designed to build upon the previous, with iterative feedback loops and milestone reviews to guide progress.

Phase	**Description**	**Duration**	**Key Activities**

Phase 1	Requirements Gathering & Design	—	Conducted stakeholder consultations, defined system specifications, mapped user needs, and drafted architectural blueprints.
Phase 2	Software & Database Development	4 weeks	Developed frontend and backend modules, configured PostgreSQL database, implemented schema models, and established secure data pipelines.
Phase 3	API Development & Testing	6 weeks	Built RESTful APIs, integrated sensor data ingestion and dashboard retrieval, conducted unit and integration testing, and validated data exchange protocols.
Phase 4	Feedback & Optimization	3 weeks	Engaged stakeholders for usability testing, refined UI/UX elements, optimized system performance, and resolved identified issues.
Phase 5	Deployment & Documentation	2 months	Finalized system deployment, prepared user manuals and technical documentation, and conducted training sessions for county ICT staff.

2. Software Development

2.1. Deliverable Focus

Writing and testing software code to implement system functionalities, develop interfaces, and refine through stakeholder feedback. It focused on delivering a robust, modular and user-centric system.

2.2. Objectives

➢ Build modular software components for data ingestion, processing, visualisation, and alerting.

➢ Develop intuitive interfaces for both public and administrative users.

➢ Conduct iterative testing with stakeholders to ensure usability, performance, and relevance.

➢ Maintain full alignment with Nairobi City County ICT standards for security, interoperability and scalability.

It was structured into modular, iterative phases emphasising user needs, the project requirements document, data security and scalability.

2.3. Development Activities

The development of the Nairobi City-Owned Air Quality Data Management System (AQDMS) followed a structured, iterative process anchored in user-centered design, modular architecture, and stakeholder collaboration. The process was divided into four key phases: **Planning**, **Implementation**, **Testing**, and **Deployment**, culminating in a fully containerized system ready for distribution and operational handover.

➢ Planning: From Vision to Blueprint

The planning phase laid the foundation for AQDMS by translating stakeholder needs into actionable system requirements. Collaborative workshops and design sprints were conducted with Nairobi City County ICT staff, environmental officers, and Clean Air Fund representatives to define user stories and map out core functionalities.

Tools such as **Trello** were used for sprint planning and task tracking, while **Figma** supported wireframe design and UI prototyping. User personas were developed to represent county officials, public users, and analysts, guiding interface decisions and feature prioritization.

Key design decisions included:

- A **dual-interface model**: one dashboard for county officials and one portal for public users.
- A **modular architecture** to support future expansion and independent upgrades.
- **Role-based access control** to ensure secure and appropriate data visibility.

User stories were framed around real-world scenarios, such as:

- "As an environmental officer, I want to receive alerts when pollution spikes so I can coordinate a response."
- "As a Nairobi resident, I want to check air quality in my neighborhood before commuting."

These narratives shaped the system's logic, flow, and visual hierarchy

This further led to the below activities:

- ➢ Gathered and finalised user stories and acceptance criteria with NCCG environmental officers and ICT staff while prioritising features into MVP.
- ➢ Defined a high-level system architecture

- ➢ Repository and branching strategy inclusive of creating a mono-repo with a well-defined git branching strategy.
- ➢ Development environment setup inclusive of .env and local environment conventions & dockerisation.
- ➢ CI/CD pipeline baseline defined.

➢ **Implementation: Building the System Layer by Layer**

The implementation phase focused on translating design blueprints into functional software. AQDMS was developed using a modular architecture, with each layer built independently to ensure scalability and maintainability.

Key technologies included:

- o **Frontend**: Built with **Next.js**, using HTML, CSS, and HTMX for responsive, mobile-friendly interfaces.
- o **Backend**: Developed using **Node.js** and **Express.js**, with RESTful APIs powering data ingestion, retrieval, and alert logic.
- o **Database**: **PostgreSQL** served as the core engine, managed via **Prisma ORM** for schema modeling and migrations.
- o **Authentication**: **JWT tokens** enabled secure, role-based access across dashboards and APIs.

Core modules included:

- o **Sensor Integration Scripts**: interfaced with NCCG-owned sensors to ingest real-time data.
- o **Alert Engine**: monitored pollutant thresholds and triggered notifications.
- o **Visualization Layer**: rendered interactive maps and trend graphs using Leaflet.js and Chart.js..

Each module was version-controlled via **GitHub**, with branching strategies aligned to feature sets and sprint cycles.

In a nutshell the following activities took place:

- **a.** Project scaffolding inclusive of repository initialisation.
- **b.** Database models & Prisma ORM defined.
- **c.** Core API endpoints established while connecting the setup sensors to the API.
- **d.** Business logic & services setup.
- **e.** Authentication and role based access control
- **f.** Background jobs and backend testing.
- **g.** Design & components following the wireframes.
- **h.** Implement responsive UI components that consume api endpoints
- **i.** Intergration and local testing

API Development & Integration:

j. Finalise API contract with Swagger spec documentationfor all endpoints, including request/response schemas, auth requirements and error codes.
k. Finalisation of sensor ingestion endpoints
l. Data validation and normalisation with an implementation pipeline:
    i. Step A: parse and standardise
    ii. Step B: apply validation per every sensor reading.
    iii. Step C: run anomaly detection
    iv. Step D: mark quality flags and persist
m. Alert engine and queues persisted
n. Caching and performance
o. Rate limiting for sensor ingestion & user tokens and security
p. Documentation & end-to-end tests that simulate sensor ingestion → validation → alert generation.

The AQDMS was architected using a **modular design approach**, enabling independent development and maintenance of the following layers:

- **Data Collection Layer**: Interfaces with low-cost environmental sensors to ingest real-time air quality data.
- **Data Processing Layer**: Cleans, validates, and stores sensor data using automated pipelines and schema-driven logic.
- **API Layer:** Ensure interoperability with dashboard, portal, mobile apps and third-party platforms.
- **Visualisation Layer**: Powers interactive dashboards for both county officials and public users, offering real-time filtering, trend analysis, and report generation.

Key development milestones included:

➢ Building responsive dashboards tailored to the needs of both technical and non-technical users.

➢ Optimising backend performance to ensure **low-latency data flow** from sensors to dashboards.

➢ Implementing **role-based access control** and secure API endpoints to protect data integrity.

➢ **Testing & Optimisation: Validating Performance and Reliability**

Testing was conducted throughout development to ensure robustness, accuracy, and usability. The team employed a layered testing strategy and matrix:

- **Unit Testing**: Implemented using **Jest** to validate individual functions, such as data parsers and alert triggers.
- **Integration Testing**: Conducted via **Postman** and **Swagger**, simulating real-time sensor inputs and dashboard queries.
- **User Acceptance Testing (UAT)**: Involved County staff and community testers evaluating system responsiveness, clarity, and accessibility across devices.
- **Load/Stress tests:** Involved simulation of high ingestion rates testing the portal, dashboard and backend for scenarios with high traffic of data in and requests being made.

Test scenarios included:

- Simulated high-frequency sensor streams to assess ingestion stability.
- Role-based access validation to ensure correct permissions.
- Dashboard rendering tests across mobile and desktop environments.

Bug tracking and resolution were managed via **GitHub Issues**, with feedback loops integrated into sprint retrospectives. Performance tuning was based off on provided feedback and identified bugs. Security hardening was also priotised with secure .env file handling, enforced HTTPs & organisation-validated SSL certs via Nginx.

Health checks were also set for the containers with configured logs retention and alerts channel

➢ Deployment: Going Live with Confidence

The final phase involved packaging and deploying AQDMS as a fully containerized system. Each component—Portal, Dashboard, and Backend—was independently containerized using Docker, ensuring consistent environments across development, testing, and production.

Deployment highlights:

- **Distribution**: The complete system was bundled into zipped folders and shared via Google Drive, allowing stakeholders to download and deploy the system with ease.
- **Build Process**: Each folder includes a dedicated Dockerfile. Users navigate into each project directory and run docker build to compile the containers.
- **Runtime Execution**: Containers are launched using docker run or orchestrated via docker-compose up --build -d for full-system deployment.

- **Hosting**: The system is deployed on Nairobi City County’s on-premise servers, with optional cloud migration pathways.
- **CI/CD Pipeline**: Configured using **GitHub Actions** for automated builds, tests, and deployments.
- **Environment Configuration**: .env files and secure deployment scripts ensure proper handling of credentials and performance tuning.
- **Versioning**: Semantic versioning (e.g., v1.0.0) was adopted to track releases and updates.

2.4. Tools & Technologies

Component	**Technologies Used**
Frontend	Next.js, HTML, CSS, HTMX
Backend	Node.js, Express.js
Database	PostgreSQL
Version Control	Git, GitHub
Testing & Validation	Jest (unit testing), Postman (API testing)

These technologies were selected for their performance, scalability, and compatibility with open-source standards.

2.5. Key Logic and Algorithms

The AQDMS platform incorporates several core algorithms and backend logic flows that ensure data accuracy, system responsiveness, and operational reliability. These components are designed to handle real-time environmental data, trigger alerts, and support dynamic filtering across user interfaces. Below is a breakdown of the most critical logic modules:

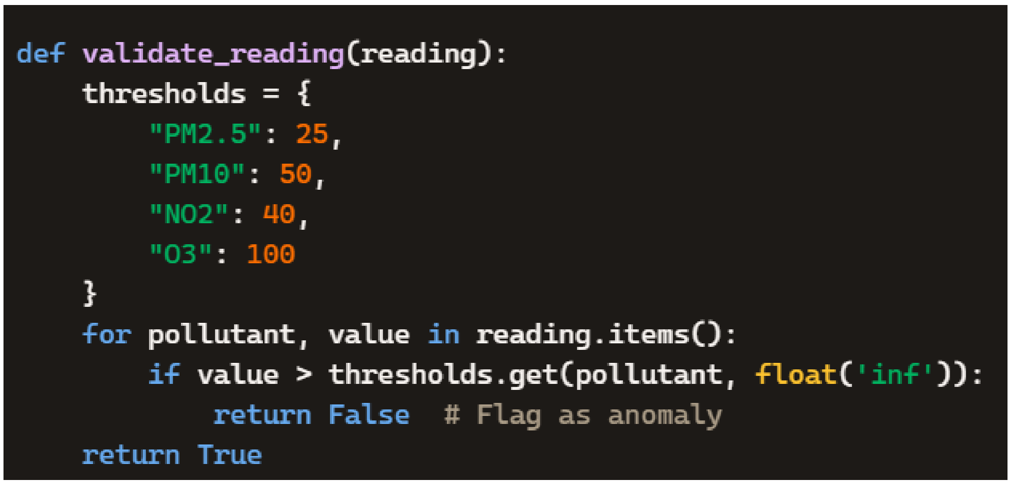
- **Data Validation Algorithm:** Applies WHO/NEMA thresholds to detect anomalies in sensor data.

Purpose: To ensure that incoming sensor readings are accurate, consistent, and within acceptable environmental thresholds.

Why it matters: Sensor data can be noisy or erratic due to hardware faults, environmental interference, or transmission errors. Validating this data before storage and visualization is essential for public trust and policy reliability.
How it works:

- Each incoming reading is checked against WHO and NEMA threshold values for pollutants like PM2.5, PM10, NO_2, and O_3.
- Anomalies (e.g., sudden spikes or out-of-range values) are flagged and excluded from visualization.
- Validated readings are stored in the SensorReadings table and passed to the dashboard/API layer.

Pseudocode:



```
def validate_reading(reading):
    thresholds = {
        "PM2.5": 25,
        "PM10": 50,
        "NO2": 40,
        "O3": 100
    }
    for pollutant, value in reading.items():
        if value > thresholds.get(pollutant, float('inf')):
            return False  # Flag as anomaly
    return True
```

Fig 15. Pseudocode illustrating the data validation logic used during ingestion. Only readings that meet predefined pollutant thresholds are stored in the SensorReadings table and made available to the dashboard and API layer.

- **Alert Engine Logic:**

Purpose: To notify county officials when pollution levels exceed safe limits, enabling timely interventions.

Why it matters: Automated alerts reduce response time and improve public health outcomes by enabling proactive mitigation.

How it works:

- When a validated reading crosses a pollutant threshold, the system triggers an alert.
- The alert is logged in the AlertLog table, queued in AlertQueue, and dispatched to registered users in AlertUsers.
- Retry logic ensures delivery even during network interruptions.

Code snippet:

```
Every 10 minutes:
  1. Get pending or failed alerts (max 10)
  2. For each alert:
      a. Mark as 'processing'
      b. Fetch sensor and station details
      c. Check thresholds against user preferences
      d. Generate report (Excel)
      e. Send email notification
      f. Log alert
      g. Mark as 'done' or 'failed'
```

Fig 16: This is pseudo-code that simplifies the alert generation logic for both the dashboard and portal platforms.

- **Data Ingestion Logic:** Ensures smooth data flow from sensors to the API layer using asynchronous operations.

Purpose: To manage real-time data flow from sensors to the backend API and database.
Why it matters: Smooth ingestion ensures that dashboards reflect current conditions and that historical data remains complete and queryable.
How it works:

- Sensor data is pushed to the backend via RESTful endpoints.
- Asynchronous operations (using async/await in Node.js) prevent blocking and allow concurrent ingestion.
- Validated data is stored in PostgreSQL and indexed for fast retrieval.

Pseudocode:

```
BEGIN DataIngestionProcess

  LOG "Starting data ingestion..."

  TRY
    // Step 1: Receive new sensor data
    RECEIVE sensorPayload FROM external source (API, IoT gateway, or CSV
file)

    // Step 2: Validate incoming payload
    IF sensorPayload IS EMPTY OR MISSING critical fields THEN
        LOG "Invalid payload - skipping entry"
        RETURN
    END IF

    VALIDATE sensorPayload fields:
        - sensorId EXISTS
        - pm25, pm10, co2, temperature, humidity ARE numeric
        - timestamp IS valid ISO format

    // Step 3: Confirm sensor registration
    FETCH stationRecord FROM Station WHERE sensorId ==
sensorPayload.sensorId

    IF stationRecord DOES NOT EXIST THEN
        LOG "Unknown sensor ID - ignoring reading"
        RETURN
    END IF

    // Step 4: Compute AQI and categorize alert level
    CALCULATE aqiValue = ComputeAQI(sensorPayload.pm25,
sensorPayload.pm10)
    SET alertLevel = DetermineAlertLevel(aqiValue)

    // Step 5: Store the reading
    INSERT INTO SensorReadings TABLE:
        sensorId = sensorPayload.sensorId
        stationId = stationRecord.id
        pm25 = sensorPayload.pm25
        pm10 = sensorPayload.pm10
        aqi = aqiValue
        alertLevel = alertLevel
        timeStamp = sensorPayload.timestamp
        status = "processed"

    // Step 6: Check for alert conditions
    IF alertLevel IS ABOVE threshold THEN
        CREATE alertJob IN alertQueue:
            readingId = newlyInsertedReading.id
            sensorId = sensorPayload.sensorId
            stationId = stationRecord.id
            status = "pending"
            timeStamp = sensorPayload.timestamp
    END IF

    // Step 7: Update last activity timestamp for station
    UPDATE Station SET lastActive = CURRENT_TIMESTAMP WHERE id ==
stationRecord.id
```

```
    LOG "Ingestion complete for sensorId: " + sensorPayload.sensorId

  CATCH error
    LOG "Data ingestion failed: " + error.message
    STORE error details IN IngestionLogs TABLE
  END TRY

END DataIngestionProcess
```

Fig 17: Pseudocode illustrating controlled data ingestion from environmental sensors into the PostgreSQL database. It ensures that only validated readings are asynchronously processed and stored, maintaining system integrity and real-time responsiveness.

➢ Filtering and Search Logic

Purpose: To allow users to explore air quality data by station, timeframe, and pollutant type.

Why it matters: Dynamic filtering supports hotspot analysis, historical comparisons, and personalized data views.

How it works:

- Filters are applied via frontend dropdowns and passed as query parameters to the backend.
- Prisma ORM constructs SQL queries based on selected filters.
- Results are rendered using Chart.js and Leaflet.js for visual clarity.

Example Query:

```
const readings = await prisma.sensorReadings.findMany({
  where: {
    stationId: selectedStation,
    timestamp: {
      gte: startDate,
      lte: endDate
    }
  },
  orderBy: { timestamp: 'asc' }
});
```

Fig. 17: Pseudocode demonstrating filtered retrieval of sensor readings from the PostgreSQL database using Prisma ORM. The query selects data by station ID and timestamp range, ordered chronologically to support time-series analysis and dashboard visualisation.

- **Scheduling and Retry Logic**

Purpose: To ensure periodic tasks (e.g., alert dispatch, backup, summary generation) run reliably.

Why it matters: Scheduled jobs maintain system health and ensure the timely delivery of critical information.

How it works:

- Background jobs are scheduled using cron expressions or job queues (e.g., BullMQ).
- Retry logic is built into alert dispatch and ingestion flows to handle transient failures.

Diagram: Alert Dispatch Flow

```
Sensor → API → Validation → AlertQueue → DispatchJob → Email/SMS
                       ↘ AlertLog
```

2.6. Stakeholder Engagement

- **Live Demos:** Showcased working prototypes to gather real-time feedback.
- **Usability Walkthroughs:** Tested accessibility across literacy levels and devices.
- **Joint Steering Committee:** Ensured cross-sectoral alignment and policy coherence.

2.7. Outcomes

The software development phase was successfully delivered:

- A **stable and responsive system** capable of handling real-time data ingestion and visualisation.
- A **user-friendly interface** that meets the needs of both county officials and the general public.
- A **scalable architecture** that supports future expansion, including additional sensors, new data types, and external integrations.
- A system ready for **deployment and operational handover**, with full documentation and training support.

2.8. Challenges and Resolutions

Throughout the development of the Nairobi City-Owned Air Quality Data Management System (AQDMS), the team encountered several technical and operational challenges. These were systematically addressed through adaptive design strategies, stakeholder engagement, and iterative refinement of system components.

◇ Sensor Data Variability

Challenge:
Early sensor readings exhibited inconsistent values caused by environmental noise, hardware precision limits, and network transmission losses. This variability affected data aggregation accuracy and risked misleading trends in the public portal.

Resolution:
The backend team implemented a **normalized database schema** and introduced **data validation and smoothing logic**. Each incoming reading is now processed through statistical anomaly detection and range validation before storage. The system uses rolling median filters and quality flags to mark unreliable data.

Trade-offs / Limitations:
Although filtering improves accuracy, it introduces minor latency (a few seconds delay) before readings are published. This was considered acceptable to prioritise data integrity and public confidence.

◇ Server Downtime and Resource Bottlenecks

Challenge:
During initial load testing, backend services experienced intermittent downtime due to inefficient resource utilisation in the ingestion pipeline. The synchronous data processing model caused blocked requests when multiple sensors transmitted simultaneously.

Resolution:
The team redesigned the ingestion workflow using **asynchronous queue handling** and optimised database write operations through **batch inserts**. Docker resource limits and PostgreSQL connection pools were fine-tuned to prevent container overutilization. These changes stabilised API uptime during stress tests and doubled ingestion throughput.

Trade-offs / Limitations:
To maintain high concurrency, non-critical tasks (e.g., generating summary reports) were moved to scheduled background jobs, slightly delaying real-time analytics but ensuring overall system responsiveness.

◇ Limited Sensor Data During Development

Challenge:
Midway through development, access to live sensor data was delayed due to procurement and deployment timelines. This hindered real-world testing and calibration of alert logic.

Resolution:
The team developed **sensor data simulators** that mimicked live readings, enabling full validation of the ingestion, alerting, and visualization modules. These simulations followed realistic diurnal air quality patterns based on historical open data sources. This approach allowed uninterrupted development despite hardware delays.

Trade-offs / Limitations:
While simulations provided reliable testing conditions, they could not replicate unpredictable field issues such as hardware drift or communication losses. Final calibration was deferred to pilot deployment once real sensors came online.

- ### Security and Access Control

Challenge:
Managing secure access for multiple roles—county administrators, environmental analysts, and public users—required robust authentication and fine-grained authorization controls.

Resolution:
The team implemented **JWT-based authentication** integrated with **role-based access control (RBAC)** in the API layer. Access levels were clearly defined in the database schema, ensuring that administrative functions (e.g., data editing, report generation) were restricted to authorized accounts while keeping public data accessible through the portal.

Trade-offs / Limitations:
The RBAC configuration adds moderate complexity to user management and token refresh logic, but the benefit of maintaining compliance with data governance standards outweighs the added setup effort.

- ### Infrastructure Integration and Deployment Coordination

Challenge:
Deploying the containerised system within NCCG’s infrastructure introduced compatibility issues with existing network policies and SSL requirements. Some Docker services initially failed to start under restricted ports and SELinux configurations.

Resolution:
Joint troubleshooting sessions were conducted between the development and NCCG ICT teams. The reverse proxy (Nginx) was reconfigured to align with internal port mappings, and Let’s Encrypt SSL certificates were automated using Certbot. Detailed deployment documentation and runbooks were also created to standardise future updates.

Trade-offs/Limitations:
Strict firewall rules occasionally delay container image updates, requiring manual approval from network administrators. However, this enhances security and audit compliance across deployments.

3. Build and Compile Process

The Nairobi Air Quality Data Management System (AQDMS) has been fully containerized to simplify deployment, ensure consistency across environments, and support modular scalability. The system comprises three core components—Portal, Dashboard, and Backend—each packaged with its own Dockerfile for independent setup and execution.

3.1. Step-by-Step Deployment Instructions

3.1.1. Download and Extract the System Files

Begin by accessing the zipped project files via the provided

Google Drive link:
https://drive.google.com/drive/folders/1QAqYxE2D9widaU5kXiV5__50DqOnsIMK?usp=drive_link

Extract the contents to your preferred working directory. You will find three main folders:

- portal/ – the public-facing interface
- dashboard/ – the administrative and monitoring interface
- backend/ – the API and data management service

Each folder contains its own Dockerfile and environment configuration template.

3.1.2. Environment Setup

To ensure stable performance during development, testing and production deployment of the Air Quality Data Management System(AQDMS) the following minimum hardware and software specifications are recommended:

Component	Specification	Notes
Operating system	Ubuntu 22.04 LTS/Windows 10 (WSL2 enabled)/macOS 13+	Linux recommended for compatibility with Docker
Processor (CPU)	Intel Core i5 (4 cores) or AMD equivalent	Required for running Node.js, Next.js, Docker and PostgreSQL concurrently
Memory (RAM)	8 GB	Minimum for running containers smoothly (backend + frontend + database)
Storage	30 GB free SSD space	For local Docker images, database and build caches

Network	Stable internet connection	Required for pulling Docker images and packages as well as downloading everything require for environment setup
Graphics	Integrated graphics sufficient	Frontend visualisation tools do not require GPU acceleration

Before building the containers, ensure the following prerequisites are installed:

- Docker Engine and Docker Compose
  Install Docker Engine:

```
# Update system packages
sudo apt update -y
sudo apt install ca-certificates curl gnupg lsb-release -y

# Add Docker's official GPG key
sudo mkdir -p /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --
dearmor -o /etc/apt/keyrings/docker.gpg

# Set up the Docker repository
echo \
  "deb [arch=$(dpkg --print-architecture) signed-
by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/ubuntu $(lsb_release -cs) stable" |
\
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Install Docker Engine, CLI, and container runtime
sudo apt update -y
sudo apt install docker-ce docker-ce-cli containerd.io docker-buildx-
plugin docker-compose-plugin -y

# Verify installation
docker --version
```

Install Docker Compose:

```
# Download the latest stable release of Docker Compose
sudo curl -L
"https://github.com/docker/compose/releases/latest/download/docker-
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

# Apply executable permissions
sudo chmod +x /usr/local/bin/docker-compose

# Verify installation
docker-compose --version
```

- Git (optional, for version tracking)

```
# Update package lists
sudo apt update -y

# Install Git
sudo apt install git -y

# Verify installation
git --version
```

- PostgreSQL (if running outside Docker)

```
# Update package index and install PostgreSQL
sudo apt update -y
sudo apt install postgresql postgresql-contrib -y

# Start and enable PostgreSQL service
sudo systemctl start postgresql
sudo systemctl enable postgresql

# Confirm installation
psql --version
```

For each component, configure environment variables by copying the template:

```
cp .env.example .env
```

Update .env with:

- DATABASE_URL for PostgreSQL connection
- JWT_SECRET for authentication
- SMTP credentials for alert dispatch
- API base URLs and deployment flags

3.1.3. Build the Containers

Navigate into each project folder and run the Docker build command:

Portal

```
cd portal
docker build -t aqdms-portal
```

Dashboard

```
cd dashboard
docker build -t aqdms-dashboard
```

Backend

```
cd backend
docker build -t aqdms-backend
```

3.1.4. Run the System Locally or on a Server

Local Deployment (Development or Testing)

```
docker run -p 3000:3000 aqdms-portal
docker run -p 3001:3001 aqdms-dashboard
docker run -p 4000:4000 aqdms-backend
```

Production Deployment (Docker Compose Recommended) If a docker-compose.yml file is provided, you can orchestrate all services together:

```
docker-compose up --build -d
```

4. Database Setup

4.1. Deliverable Focus

Configuring a secure, scalable database system with stakeholder-tested performance and integration.

4.2. Objectives

- ➢ Establish a high-performance, secure data storage system.
- ➢ Ensure seamless integration with AQDMS modules and future cloud migration readiness.
- ➢ Validate database performance through stakeholder testing.

4.3. Configuration

- ➢ **Deployment:** On-premise PostgreSQL database co-located with backend services for low-latency performance.
- ➢ **Schema Design:** Modular schema includes models for Users, Stations, Sensor Readings, Reports, Feedback, and Alerts.
- ➢ ORM **Layer:** Prisma ORM used for schema management, migrations, and secure backend interactions.

4.4. Security Measures

To ensure data integrity, privacy, and compliance with Nairobi City County ICT standards, the following security protocols were implemented:

- ➢ **Role-Based Authentication**: Granular access control based on user roles (e.g., admin, analyst, public viewer).
- ➢ **SSL/TLS Encryption**: Secures data in transit between the database, backend, and dashboards.
- ➢ **Automated Backups**: Regular snapshots stored securely to support disaster recovery.
- ➢ **Access Monitoring**: Real-time logging and audit trails for database access and modification events.

These measures collectively safeguard sensitive environmental data and ensure system resilience.

4.5. Performance Testing

Comprehensive performance testing was conducted to validate the database's ability to handle real-time operations:

- **Simulated Sensor Streams**: High-frequency data ingestion tests confirmed the system's ability to process and store readings without bottlenecks.
- **Query Responsiveness**: Benchmarked retrieval times across various dashboard filters and report generation scenarios.
- **End-to-End Flow Validation**: Ensured smooth data transmission from sensors to API endpoints and visual dashboards.

The results demonstrated that the database setup meets the required standards for speed, reliability, and scalability under operational conditions.

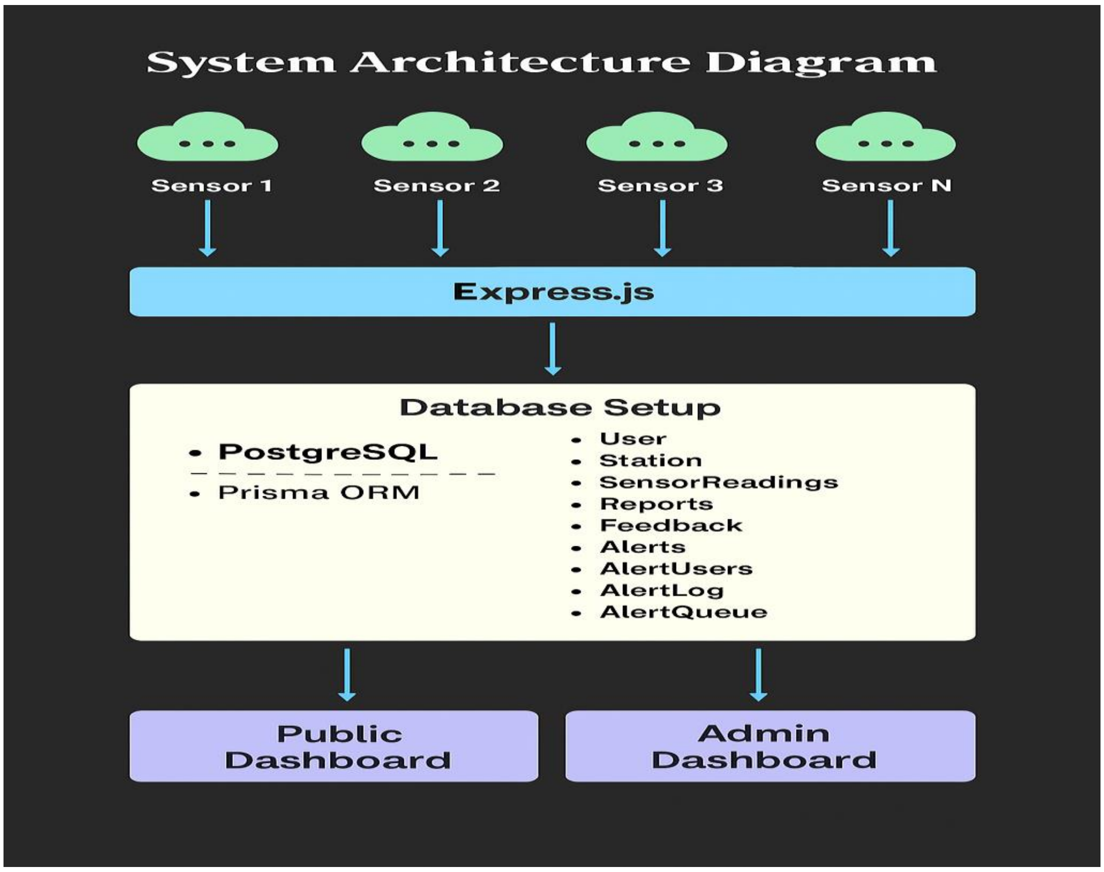


Fig. 18: System architecture diagram for the Nairobi Air Quality Data Management System (AQDMS): visually maps out the full data flow from sensor networks to dashboards and external integrations

5. API Development

5.1. Deliverable Focus

Building APIs for data integration and external system access.

5.2. Objectives

The API layer of the Air Quality Data Management System (AQDMS) was developed to:

- ➢ Facilitate real-time, secure, and structured data exchange between system components.
- ➢ Enable seamless integration with internal dashboards, external platforms, and sensor networks.
- ➢ Support scalable interoperability for future expansion, including regional and international data-sharing initiatives.

5.3. Design Overview

- ➢ RESTful architecture using Node.js (Express.js)
- ➢ Asynchronous operations for efficient real-time data flow
- ➢ Interactive documentation via Swagger and Postman
- ➢ Flexible output formats: JSON, CSV, PDF

5.4. Integration Strategy

The API was designed to serve as the central conduit for data movement across the AQDMS ecosystem. Integration capabilities include:

- **Sensor Data Ingestion**: Real-time input from low-cost air quality sensors deployed across Nairobi.
- **Dashboard Retrieval**: Secure access to processed data for both public and administrative dashboards.
- **Token-Based Authentication**: Ensures secure access control for internal users, external partners, and automated systems.
- **External System Compatibility**: Supports integration with national environmental databases, international air quality platforms, and third-party research tools.

This strategy ensures the system remains interoperable, extensible, and aligned with global data-sharing standards.

5.5. Testing & Validation

Robust testing protocols were implemented to ensure API reliability, accuracy, and resilience:

- **Ingestion & Retrieval Tests**: Simulated high-frequency sensor inputs and dashboard queries to validate throughput and response times.
- **Error Handling & Retry Logic**: Built-in mechanisms to detect and recover from sensor network disruptions, ensuring data continuity.
- **Stakeholder Validation**: County ICT staff, environmental experts, and community testers reviewed API outputs for clarity, consistency, and usability.

These efforts confirmed that the API layer meets operational demands and supports the broader goals of transparency, accessibility, and data integrity.

6. Conclusion

The System Design and Development phase resulted in a robust, modular, and scalable AQDMS platform. The system achieves its objective of delivering real-time air quality insights for Nairobi City County Government while enhancing public access and awareness. Continuous stakeholder engagement ensured usability, security, and compliance with environmental data standards. The AQDMS stands ready for full deployment and long-term operational use.

7. Appendices

Appendix 1: Public-Facing Data Portal Live portal built with Next.js and HTMX, showcasing real-time air quality data and public insights. https://air-quality-portal.vercel.app

Appendix 2: County Admin Dashboard Role-based dashboard for county officials with analytics, reporting tools, and alert management. https://air-dashboard-eight.vercel.app/overview