PUSL2020 *Software Development Tools and Practices*

Final Project Report 2024/25

Real-time Air Quality Monitoring Dashboard for Colombo

Group Number: Group 60

Faculty of Computing

Members:

ID	Name(As stated in DLE)	Degree Program (SE /CS)
10952727	Pulindu Nadil	SE
10952711	Bedde Damiru	SE
10953054	Liyanage Dasanayake	SE
10952509	Amarasinghe Senevirathne	SE
10952649	Hasiru Jayakody	SE
10953022	S.M.R.H. Senevirathna	SE

Table of Contents

Table of Contents

Group Number: Group 60 1
Faculty of Computing 1

Table of Contents ***2***

Introduction ***3***

Background **3**

Project Objective **3**

Scope **3**

System Screenshots along with the functionalities ***4***

System Overview ***18***

Test Case Design ***20***

Mock Objects ***21***

Unit Tests and Integration Tests ***22***

Functional Test Plans ***23***

Test Strategy & Critical Analysis ***24***

Conclusion ***25***

Work Matrix Breakdown ***26***

Introduction

Background

Air pollution is a growing concern in urban areas worldwide, and Colombo, the commercial capital of Sri Lanka, is no exception. Rapid urbanization, increased vehicular emissions, industrial activities, and improper waste disposal contribute to deteriorating air quality levels. Exposure to air pollutants such as particulate matter (PM2.5 and PM10), nitrogen dioxide (NO2), carbon monoxide (CO), and sulfur dioxide (SO2) poses serious health risks, including respiratory diseases, cardiovascular issues, and long-term environmental damage.

Despite the presence of national air quality monitoring stations, real-time air quality data is often limited, difficult to access, and not well-integrated into a single, user-friendly platform. Citizens, policymakers, and environmental organizations require a reliable system that provides up-to-date air quality information to take timely action. A real-time air quality monitoring system can help bridge this gap by offering live data, visualization tools, and historical trends to inform better decision-making.

Project Objective

The primary objective of this project is to develop a **Real-time Air Quality Monitoring Dashboard for Colombo** that allows users to track air pollution levels through an interactive web-based platform. The system aims to provide real-time air quality index (AQI) data, visualize pollution trends, and enable users to analyze historical air quality patterns.

The system is designed to cater to multiple stakeholders:

- **Monitoring Administrators**: Manage sensor data, user accounts, and system functionalities.
- **Data Providers**: Simulate air quality data inputs for testing and analysis.
- **Public Users**: View real-time and historical air quality data, allowing them to make informed decisions regarding outdoor activities.
- **System Administrators**: Ensure the security and maintenance of the platform.

By implementing this solution, the project aims to enhance awareness, promote proactive measures against air pollution, and support environmental policies in Sri Lanka.

Scope

The project encompasses several key features that ensure comprehensive air quality monitoring and data analysis:

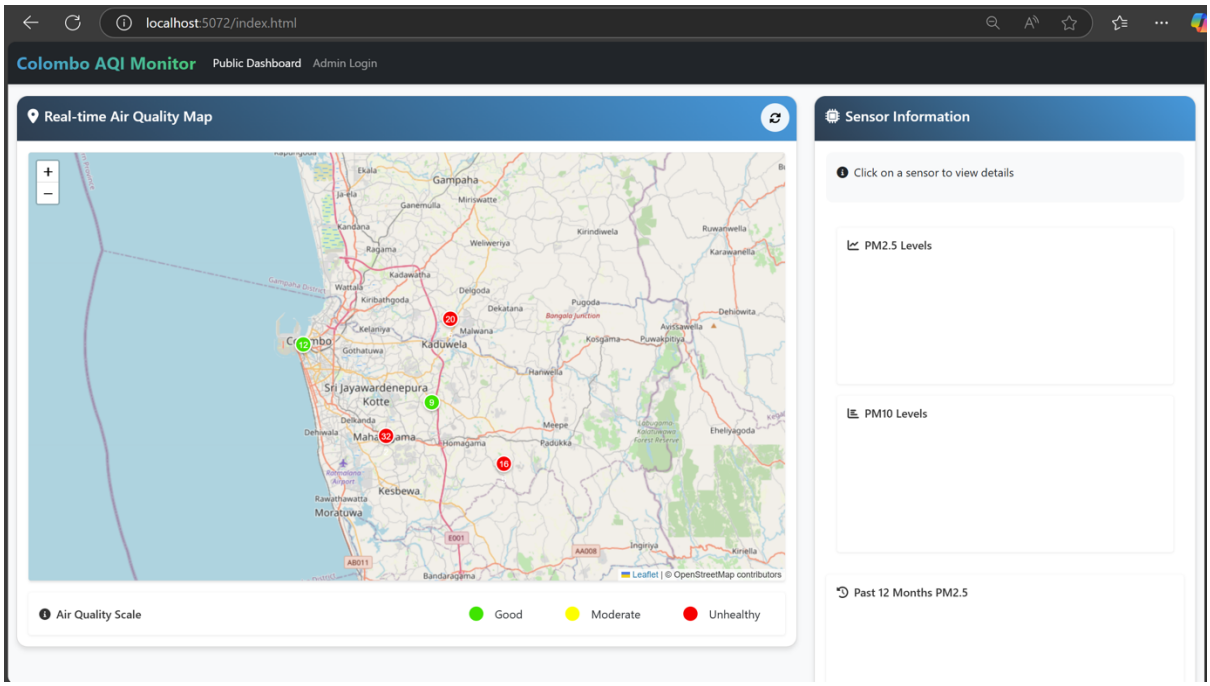
- **Real-time Air Quality Monitoring**: Display of live air quality data using simulated sensors or real-time APIs.
- **Data Visualization**: Integration of Leaflet.js to display AQI readings at different locations.
- **Historical Trends Analysis**: Graphical representation of past air quality data using Chart.js.

- **Role-Based User Access**: Different functionalities for administrators, data providers, and public users.
- **Backend and Database Management**: A robust C# ASP.NET Core Web API to handle data processing and storage in MS SQL Server.
- **User-friendly Interface**: A well-designed frontend using HTML, CSS, and JavaScript for easy navigation and data interpretation.

This project contributes to the advancement of smart city initiatives by leveraging modern web technologies to provide real-time environmental insights. The system serves as an essential tool for citizens, researchers, and government bodies to monitor air pollution and take necessary precautions effectively.

System Screenshots along with the functionalities

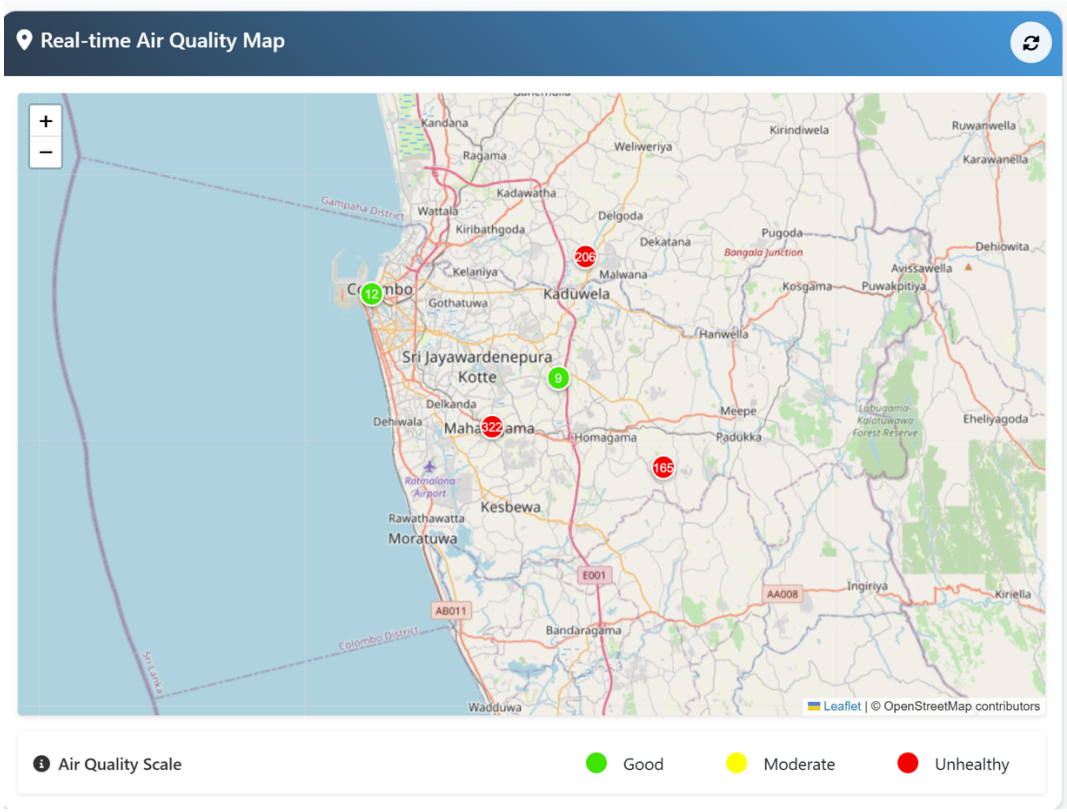
Home Page / index page



The **Home Page** serves as the landing page for all users, providing real-time air quality information for different locations in Colombo. It is accessible to the **public users**, allowing them to view air quality data without requiring authentication.

Real-Time Air Quality Map

The main feature of the home page is an **interactive real-time air quality map** that displays air quality levels at various locations. These locations represent **active sensors**, and the data is dynamically fetched from the backend, ensuring up-to-date air quality monitoring.



AQI Color Palette

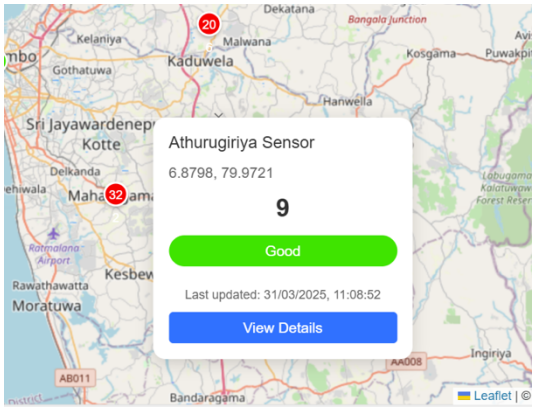
Below the map, a **color-coded AQI legend** is provided to help users interpret the air quality levels:

- **Green** – Good air quality
- **Yellow** – Moderate air quality
- **Red** – Unhealthy air quality

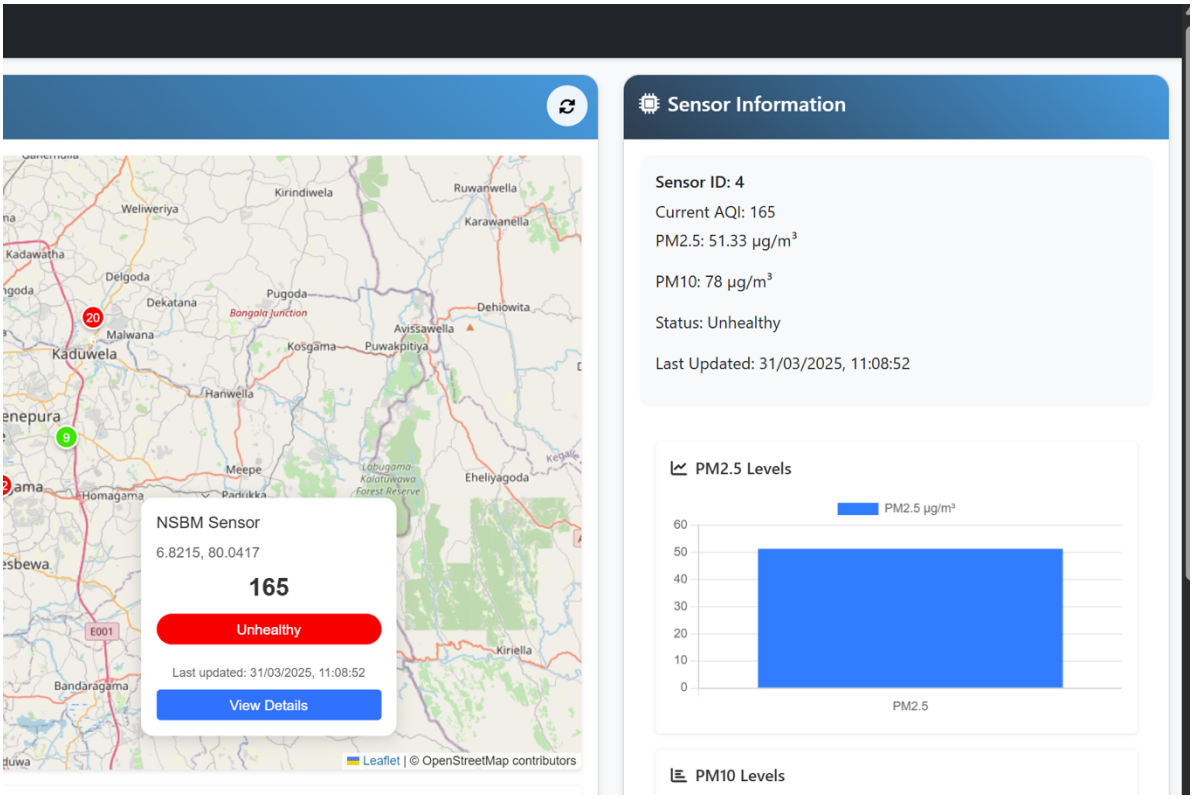
Sensor Data Pop-up

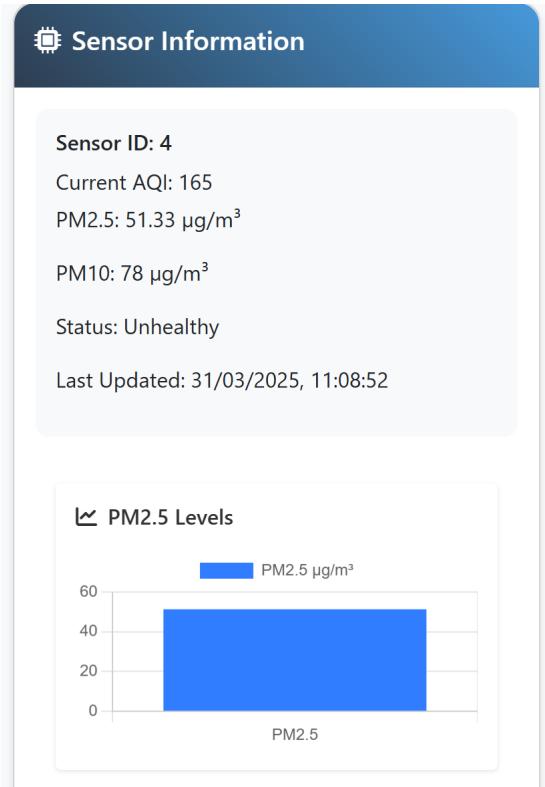
When a user clicks on a specific sensor on the map, a pop-up window appears displaying detailed air quality information for that location. This allows users to gain insights into the air quality conditions in real time.

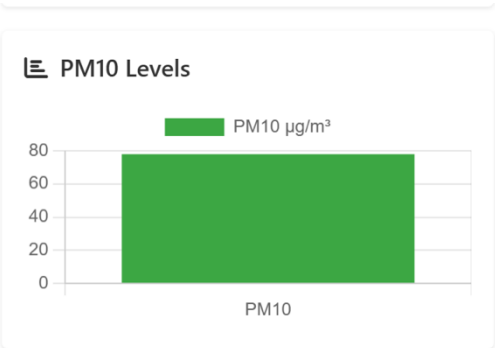
This section of the system ensures that users can easily access and interpret air quality data, promoting better awareness of environmental conditions in Colombo.

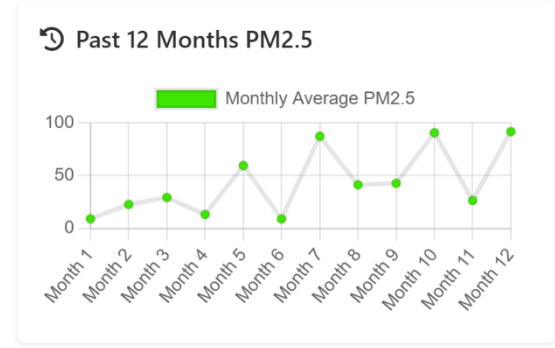


When a user clicks on the **"View Details"** button in the pop-up window, the system dynamically updates the right-hand side section of the page. This section displays **detailed sensor information**, including air quality data measures in the form of interactive charts. The charts visually represent the sensor readings over time, allowing users to analyze fluctuations in air quality at the selected location. This feature enhances data accessibility and provides a more comprehensive understanding of air quality conditions.



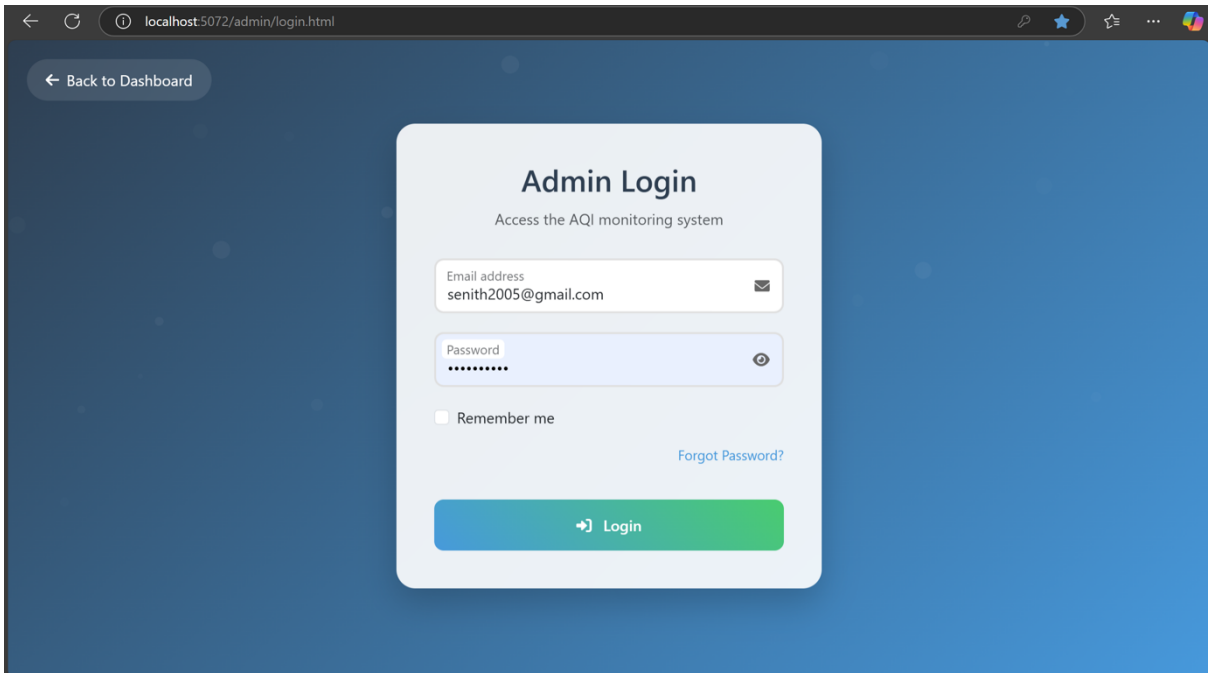




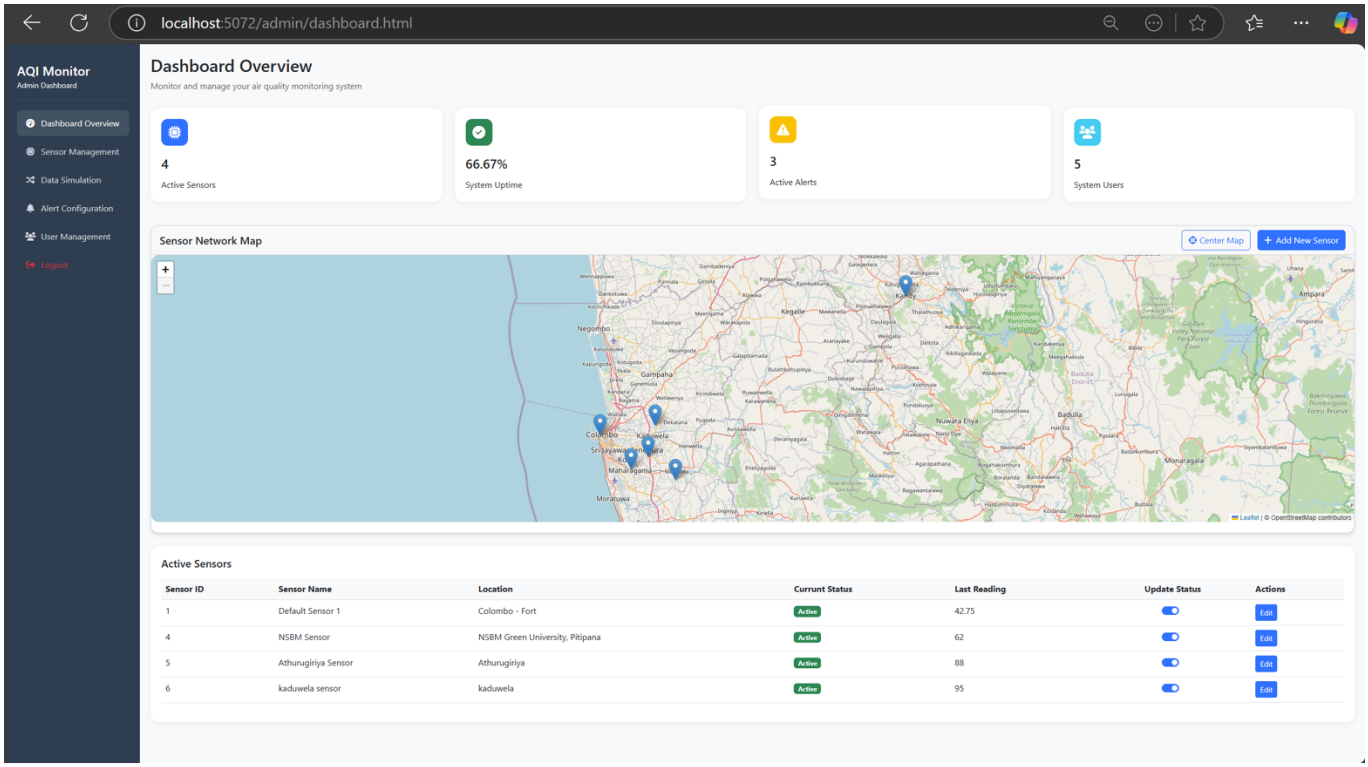


Login Page for Admins

The Admin Login Page allows administrators to securely access the system. To proceed, the administrator must enter a valid email address and password. Upon successful authentication, the administrator is redirected to the Admin Dashboard.



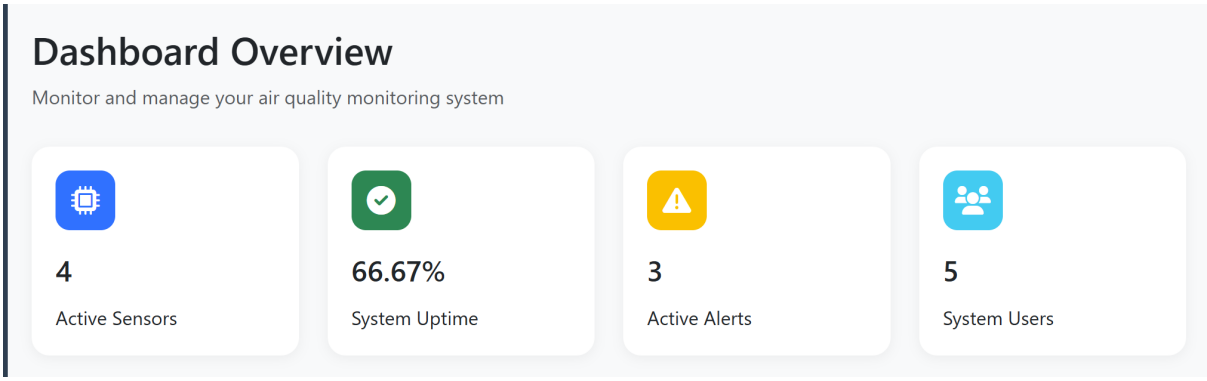
Admin Dashboard

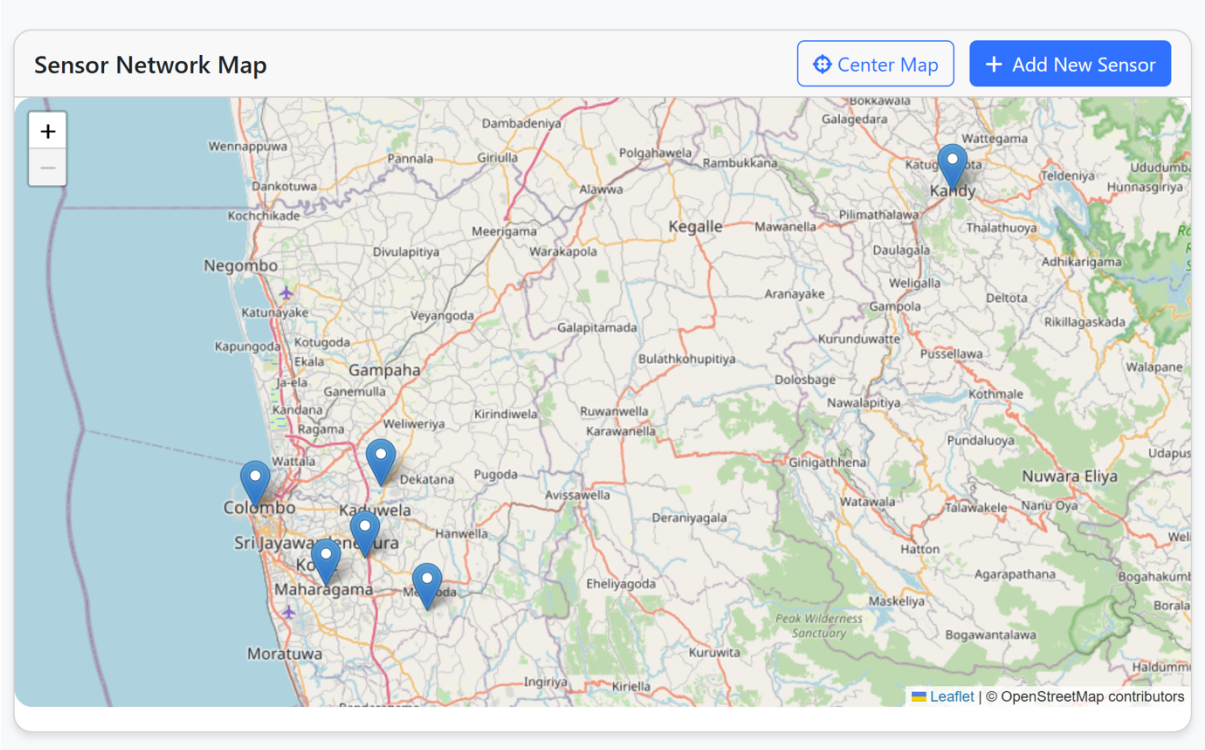


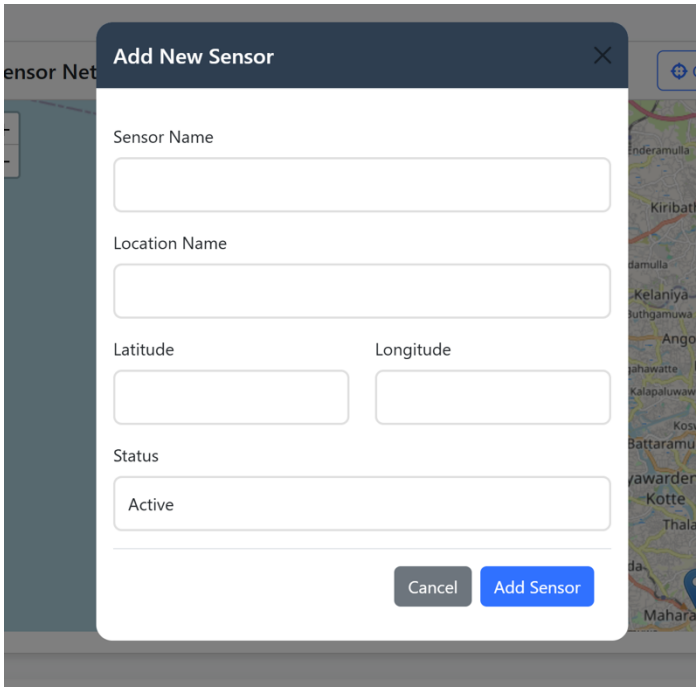
Upon logging in, the administrator is directed to the Admin Dashboard, where they are first presented with the Dashboard Overview section. This section displays key system metrics in real-time, including the number of active sensors, system uptime, the count of active alerts, and the number of currently active system users.

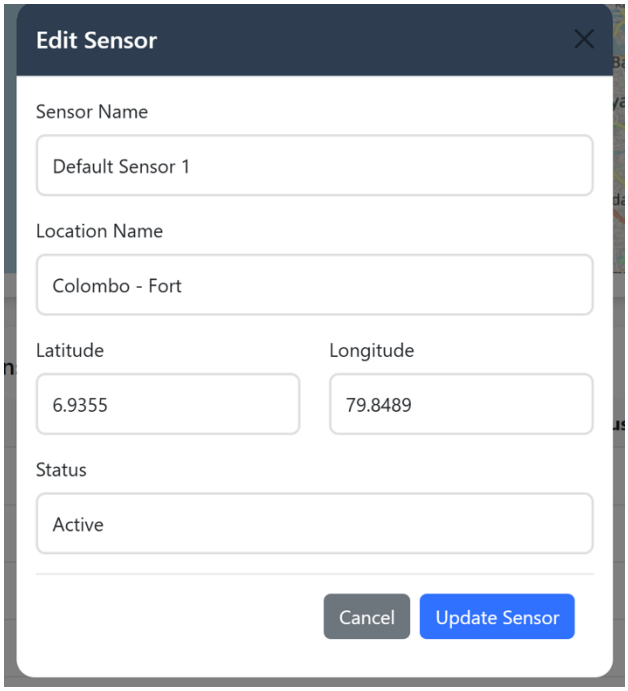
Beneath the overview, an interactive map visualizes all currently active sensors. Administrators also have the option to add new sensors directly from this map interface.

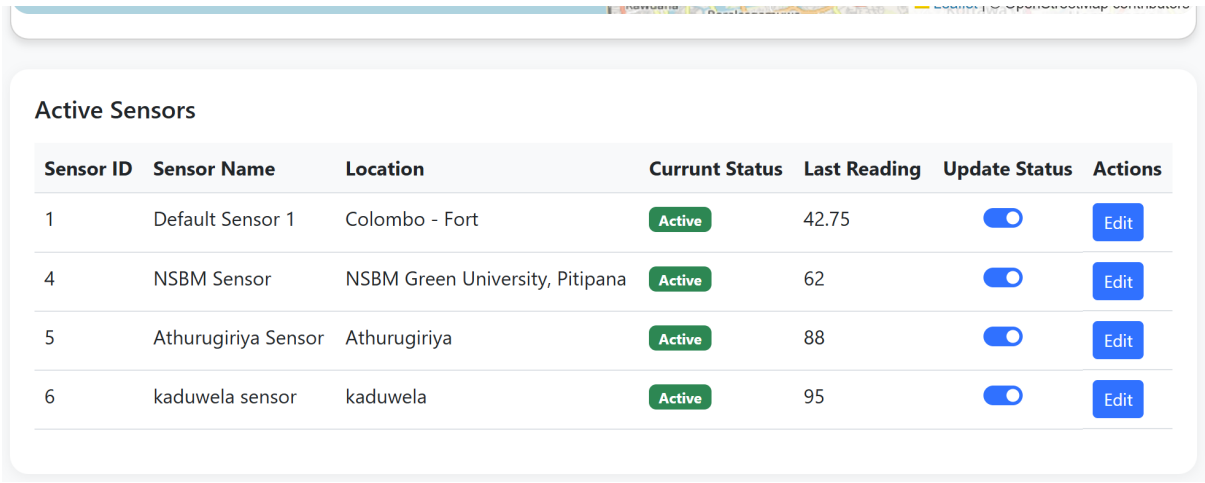
Further down, a detailed list of active sensors is displayed, providing additional information about each sensor. From this list, administrators can update the active status of sensors or edit sensor details as needed.





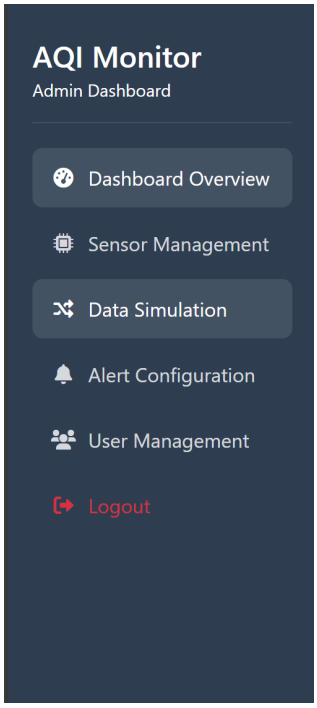






Active Sensors

Sensor ID	Sensor Name	Location	Currunt Status	Last Reading	Update Status	Actions
1	Default Sensor 1	Colombo - Fort	Active	42.75		Edit
4	NSBM Sensor	NSBM Green University, Pitipana	Active	62		Edit
5	Athurugiriya Sensor	Athurugiriya	Active	88		Edit
6	kaduwela sensor	kaduwela	Active	95		Edit

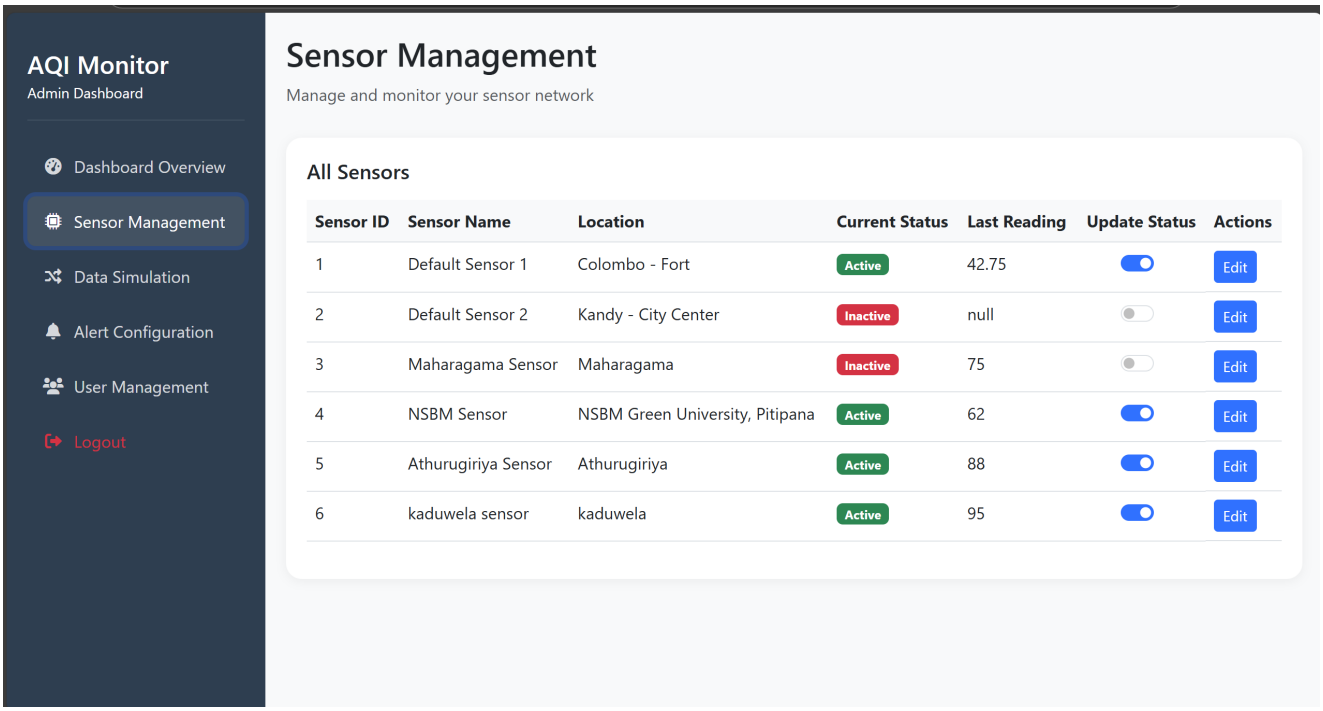


Navigation Panel

A **side navigation panel** is located on the left side of the Admin Dashboard interface, allowing administrators to easily switch between various sections of the system. The available sections include:

- **Dashboard Overview**
- **Sensor Management**
- **Data Simulation**
- **Alert Configuration**
- **User Management**
- **Logout**

Sensor Management

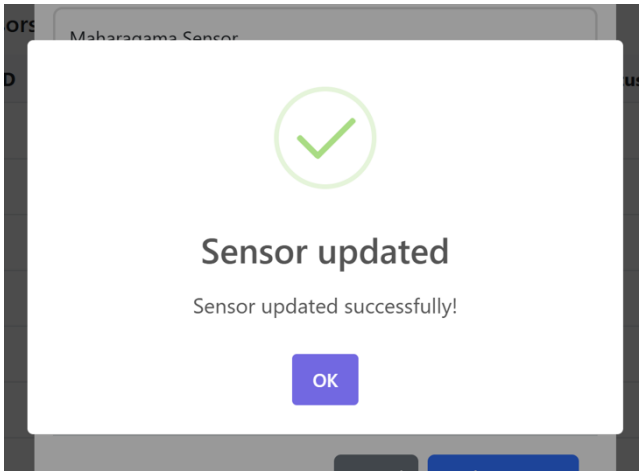


The **Sensor Management** page provides a comprehensive list of all active and inactive sensors within the system. Each sensor entry includes detailed information, enabling administrators to effectively monitor and manage the sensor network.

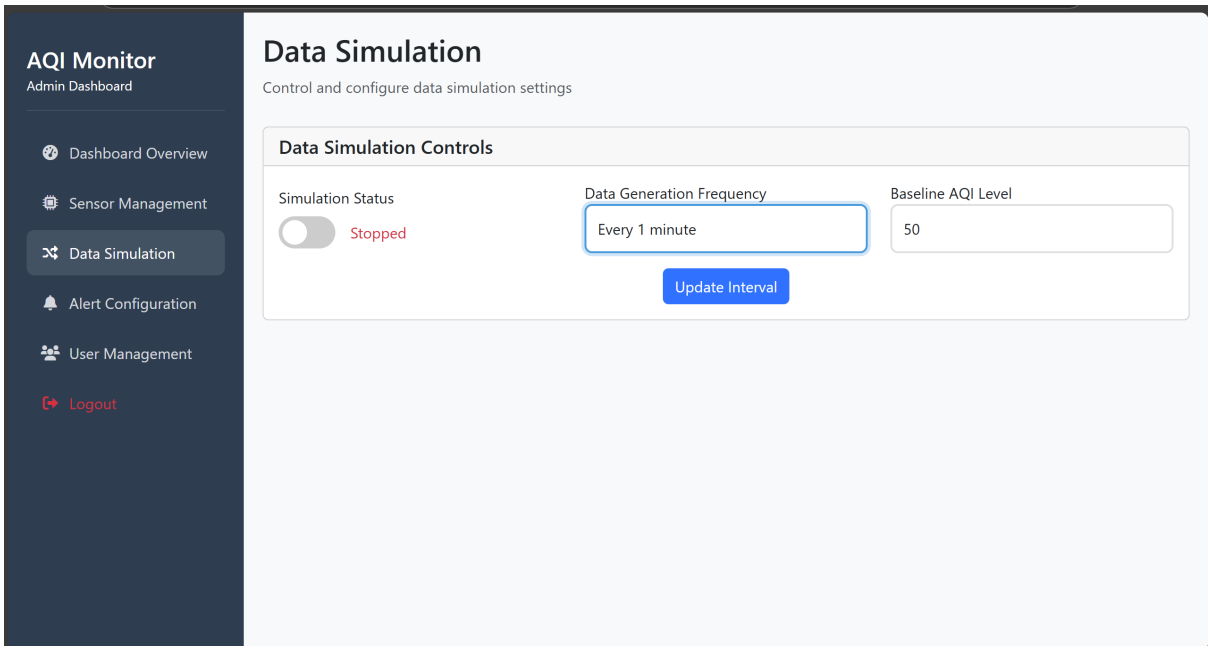
From this page, administrators can:

- Update the status of any sensor (activate or deactivate)
- Edit sensor details as necessary

This feature ensures seamless control and maintenance of the sensor infrastructure.



Data Simulation

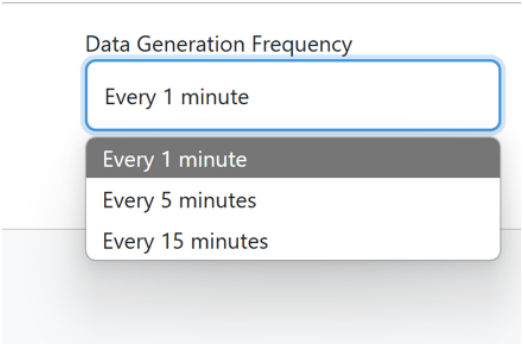


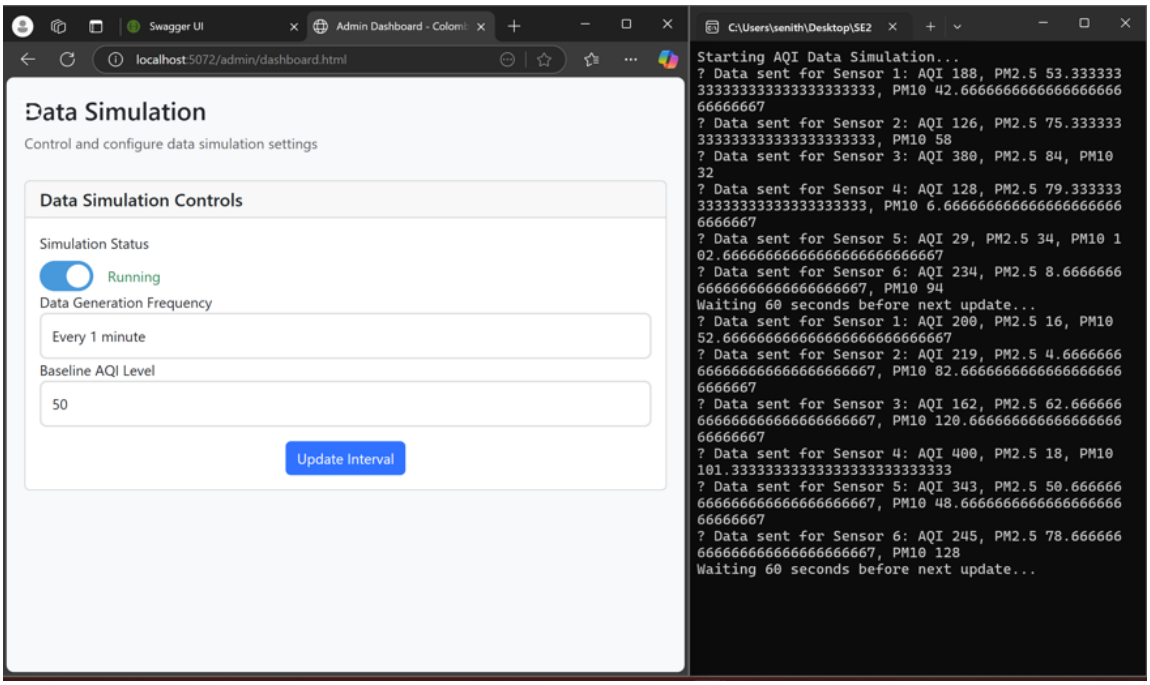
The **Data Simulation** section allows administrators to control the execution of the sensor data simulation application. A dedicated C# console application was developed to generate real-time sensor data, and its operation can be managed directly from this section using a toggle switch to start or stop the simulation.

Additionally, administrators have the ability to:

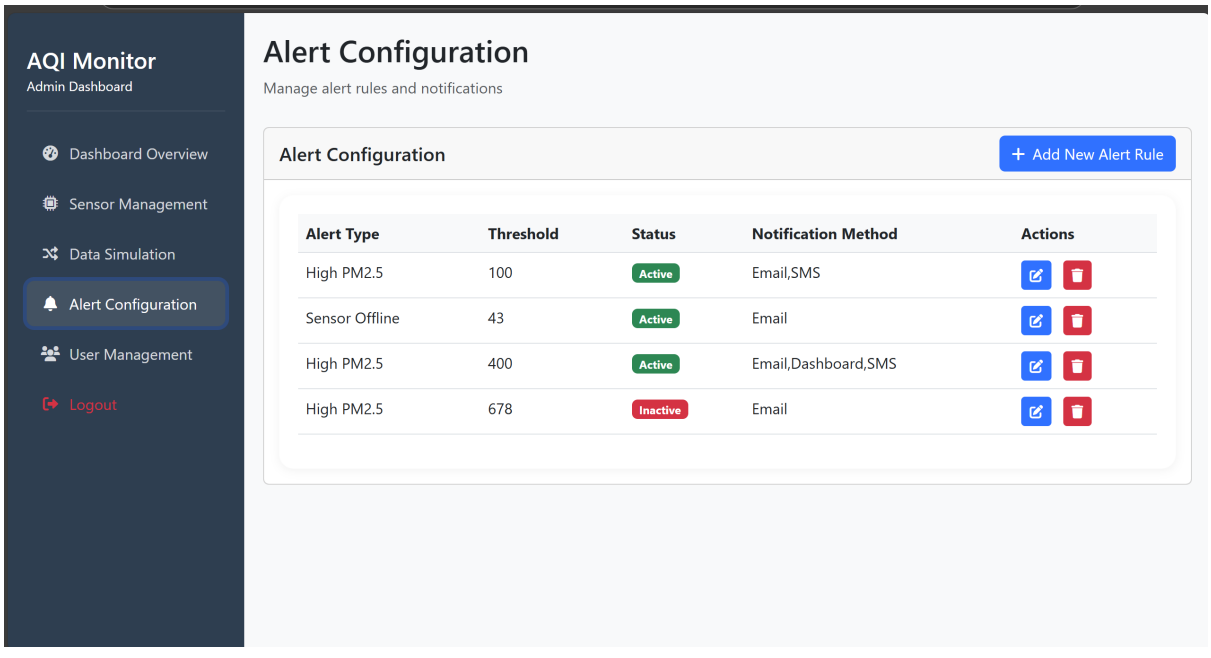
- Adjust the data generation frequency
- Modify the base Air Quality Index (AQI) level

This section provides flexibility in testing and evaluating system performance under different environmental data conditions.





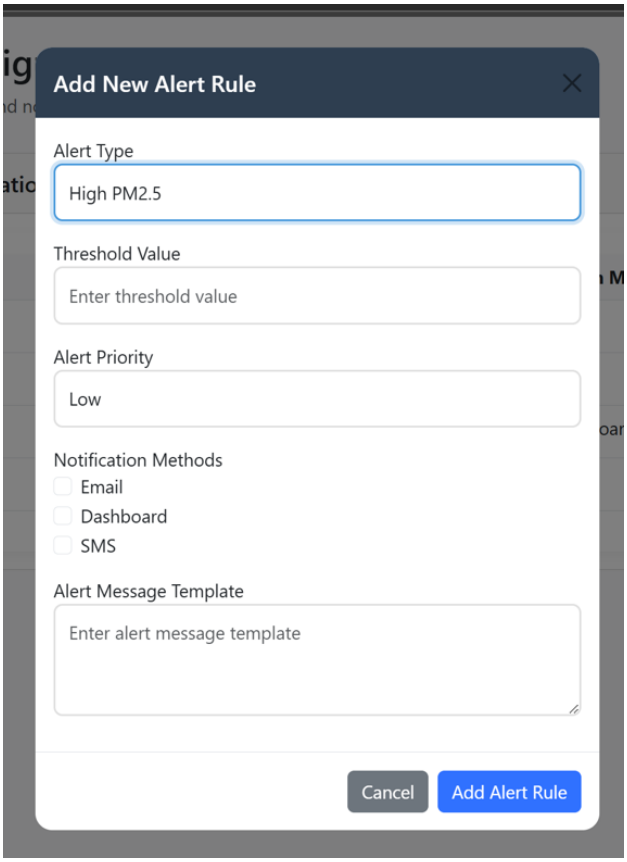
Alert Configuration

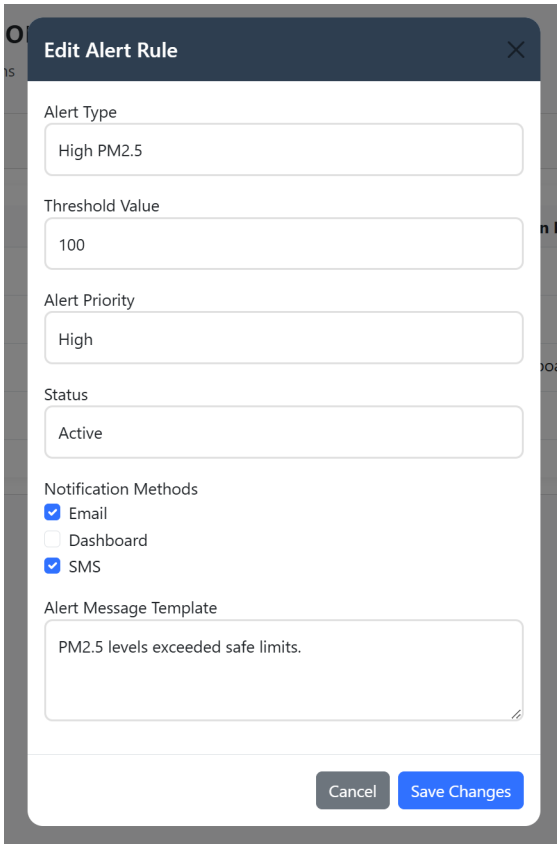


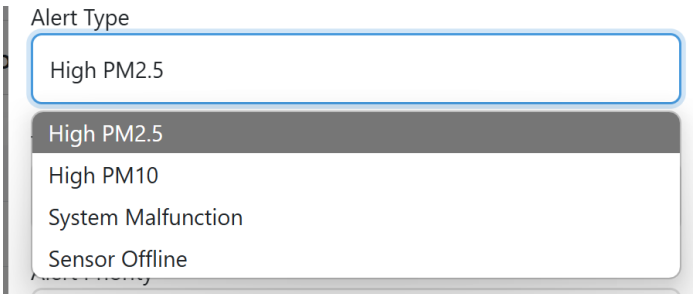
The **Alert Configuration** section enables administrators to manage system alerts effectively. From this section, administrators can:

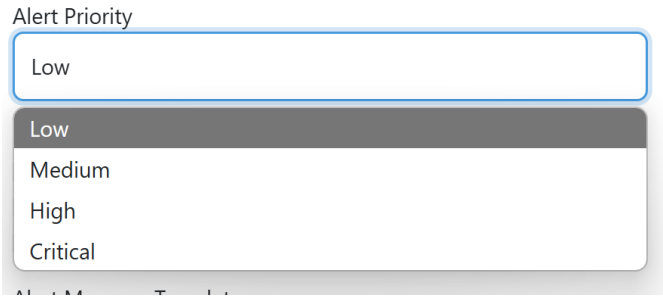
- Add new alert types based on specific conditions or thresholds (e.g., high AQI levels, sensor failure)
- View a list of configured alerts along with their detailed parameters
- Edit existing alerts to update thresholds, notification preferences, or related settings
- Delete alerts that are no longer needed

Each alert can be customized to trigger notifications or system responses, helping ensure proactive monitoring and timely action. This functionality is essential for maintaining system reliability and ensuring public safety in response to environmental changes.

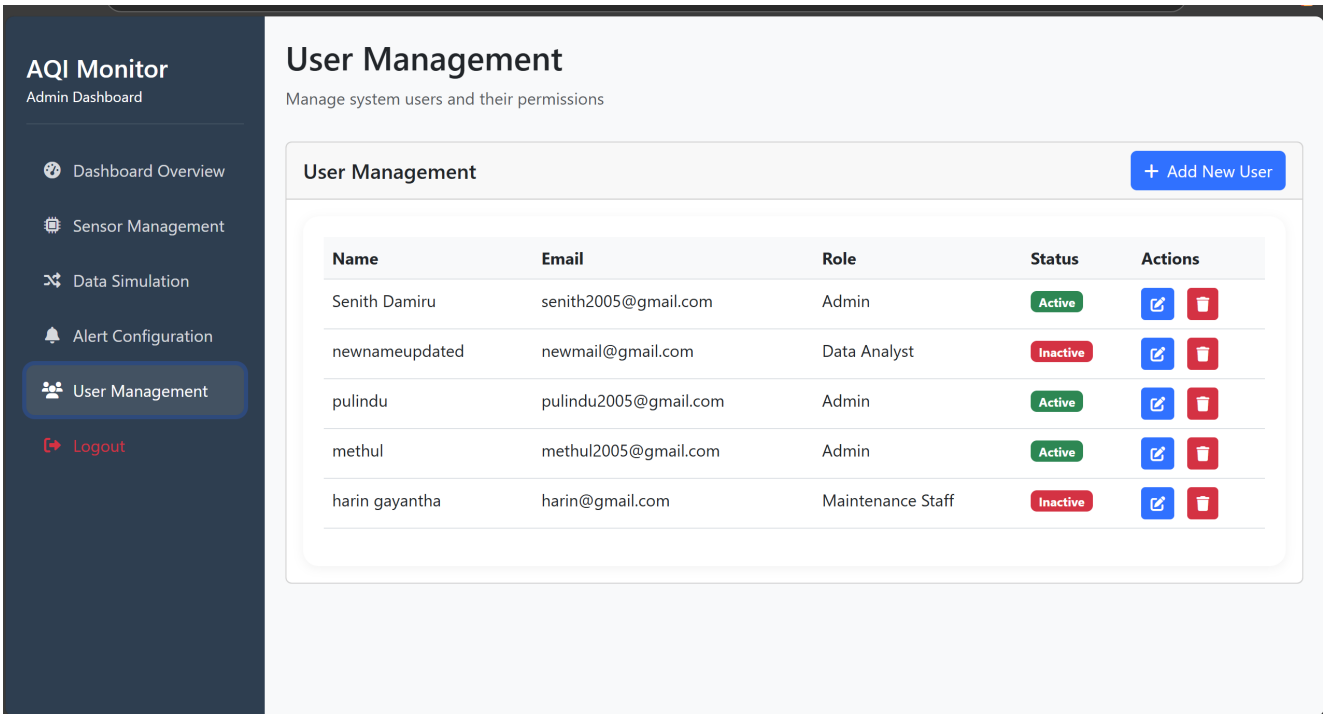


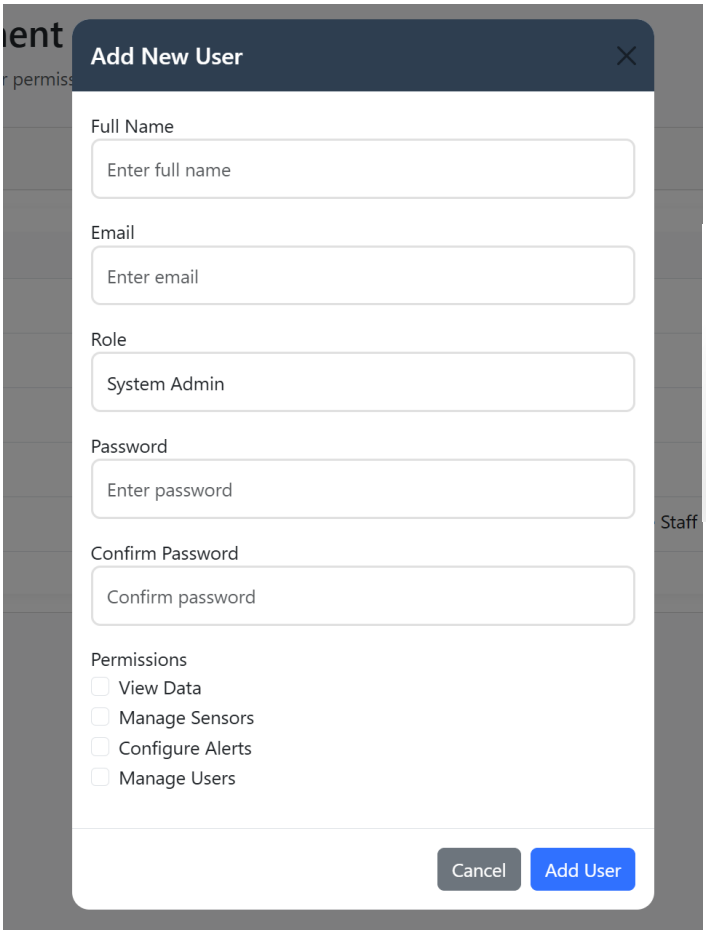


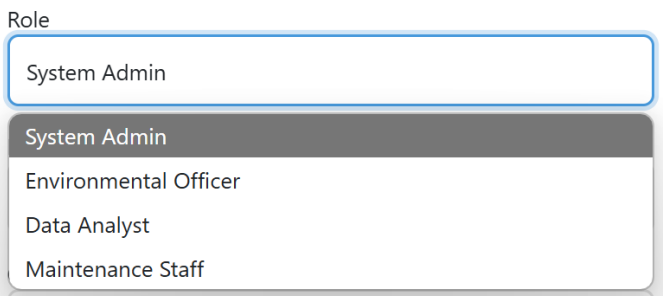


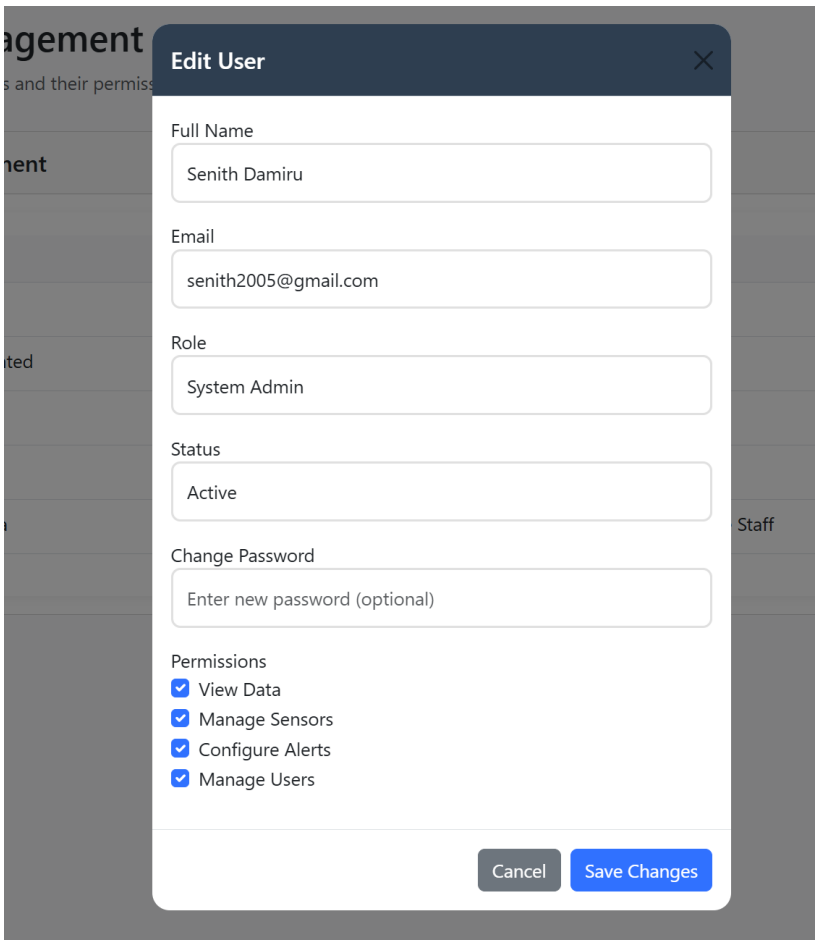


User Management







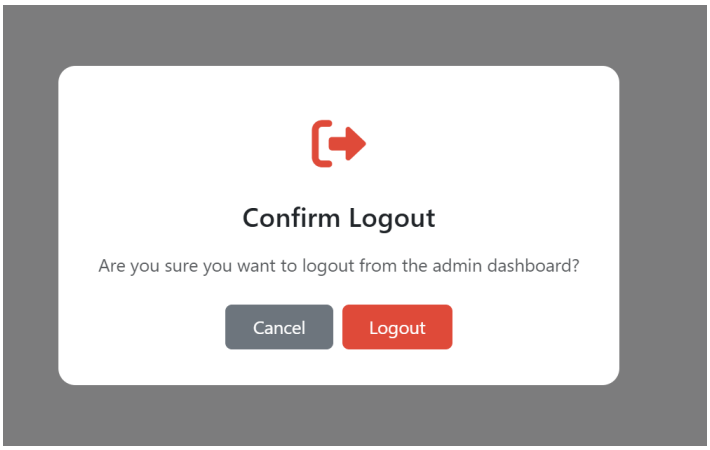


The **User Management** section allows administrators to efficiently manage system users, specifically other admin accounts. From this section, administrators can:

- Add new administrators to the system
- Edit existing admin details, such as name, email, role, change password, or status
- Delete admin accounts that are no longer required

The interface also provides a comprehensive list of all registered administrators, displaying key details including name, email, role, and account status (active/inactive).

This feature ensures proper control over administrative access, helps maintain system security, and supports role-based responsibilities within the platform.



Logout Functionality

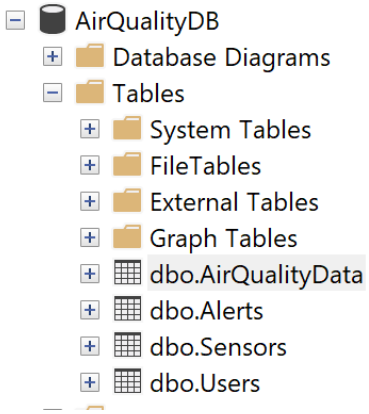
The Admin Dashboard includes a Logout option to securely end the current session. When an administrator chooses to log out, the system terminates the session and redirects the user back to the Admin Login Page.

This functionality ensures secure access control by preventing unauthorized use of the system after an administrator has finished their tasks.

MS SQL Database

As previously mentioned, Microsoft SQL Server was used as the database management system, and the database was designed and managed using SQL Server Management Studio (SSMS).

Below are screenshots showcasing the database structure, including the organization of tables and their respective fields. These visuals provide an overview of how data is stored and managed within the system, supporting core functionalities such as sensor management, alert configuration, user administration, and more.



100 %

Results Messages

	UserID	Name	Email	PasswordHash	Role	Status
1	3	Senith Damiru	senith2005@gmail.com	$2a$11$L085oU7X2lrIpsPy36IZU.UMav0a0vswlo9CBe4cKVpo...	Admin	Active
2	5	newnameupdated	newmail@gmail.com	$2a$11$czM81deBeTSPSdwO.XV7ru/96r3MG0iduc9tpaQUfS...	Data Analyst	Inactive
3	6	pulindu	pulindu2005@gmail.com	$2a$11$PunopAJIhiTTrjSvIq.9PuGss8GGFFJ6r6pWu59U1I/5...	Admin	Active
4	7	methul	methul2005@gmail.com	$2a$11$00kODsn2BU9IloZfmsGWPeta7ndXlBpeXHJpqIXM1c...	Admin	Active
5	8	harin gayantha	harin@gmail.com	$2a$11$IwCsMszd.FC9Zgo4GESOfevDucJ.C2SHFjSYrVITr.Q...	Maintenance Staff	Inactive

100 %

Results Messages

	SensorID	LocationName	Latitude	Longitude	Status	LastReading	SensorName	ActivatedAt	DeactivatedAt
1	1	Colombo – Fort	6.935500	79.848900	Active	42.75	Default Sensor 1	2025-03-30 15:14:25.620	NULL
2	2	Kandy – City Center	7.290600	80.633600	Inactive	NULL	Default Sensor 2	2025-03-30 15:07:04.773	2025-03-30 15:15:13.087
3	3	Maharagama	6.848300	79.928500	Active	75.00	Maharagama Sensor	2025-03-30 15:14:46.080	2025-03-30 17:38:32.870
4	4	NSBM Green University, Pitipana	6.821500	80.041700	Active	62.00	NSBM Sensor	2024-03-30 08:30:00.000	2025-03-30 14:05:39.687
5	5	Athurugiriya	6.879800	79.972100	Active	88.00	Athurugiriya Sensor	2024-03-30 09:00:00.000	NULL
6	6	kaduwela	6.960000	79.990000	Active	95.00	kaduwela sensor	2025-03-30 15:16:54.937	NULL

100 %

Results Messages

	AQIDataID	SensorID	AQI	PM25	PM10	Timestamp
1	1	1	200	16.00	52.67	2025-04-01 06:29:36.370
2	2	2	219	4.67	82.67	2025-04-01 06:29:36.580
3	3	3	162	62.67	120.67	2025-04-01 06:29:36.607
4	4	4	400	18.00	101.33	2025-04-01 06:29:36.633
5	5	5	343	50.67	48.67	2025-04-01 06:29:36.660
6	6	6	245	78.67	128.00	2025-04-01 06:29:36.690

100 %

Results Messages

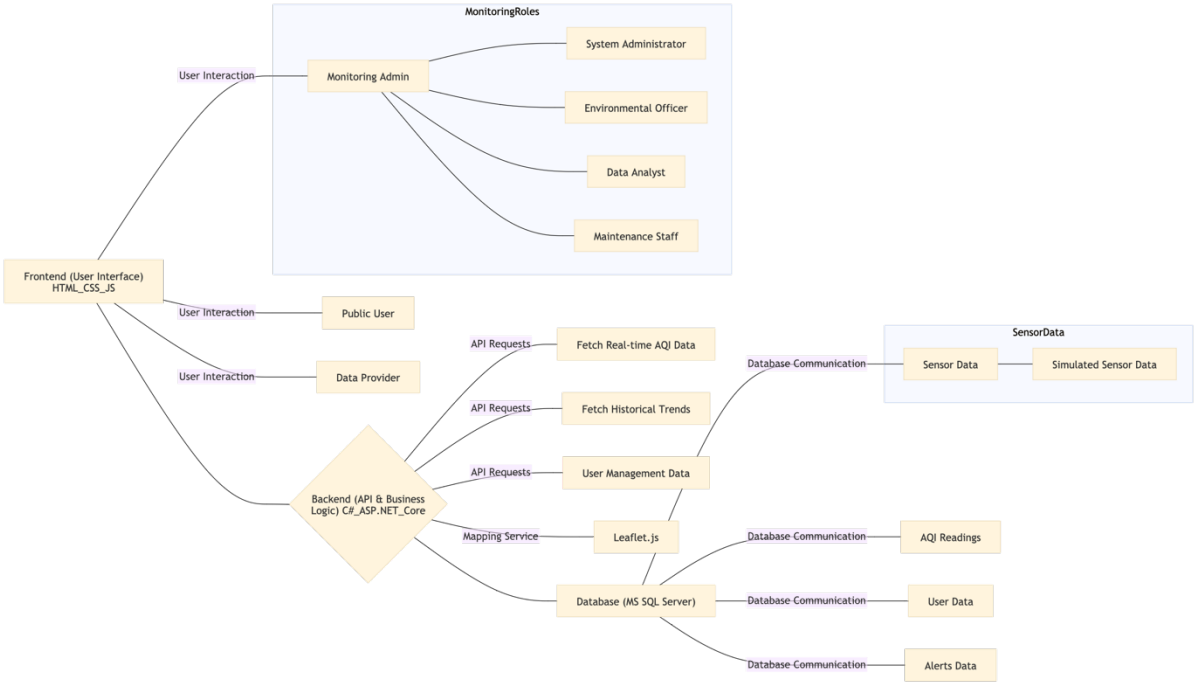
	AlertID	AlertType	AQIThreshold	Priority	NotificationMethods	AlertMessage	CreatedAt	Status
1	1	High PM2.5	100.00	High	Email,SMS	PM2.5 levels exceeded safe limits.	2025-03-28 19:46:56.660	Active
2	2	Sensor Offline	43.00	Critical	Email	Sensor has been offline for more than 30 minutes.	2025-03-30 12:47:58.543	Active
3	3	High PM2.5	400.00	Low	Email,Dashboard,SMS	fbd gfnf	2025-03-30 17:58:30.413	Active
4	6	High PM2.5	678.00	Low	Email	new alert	2025-03-30 17:49:24.013	Inactive

System Overview

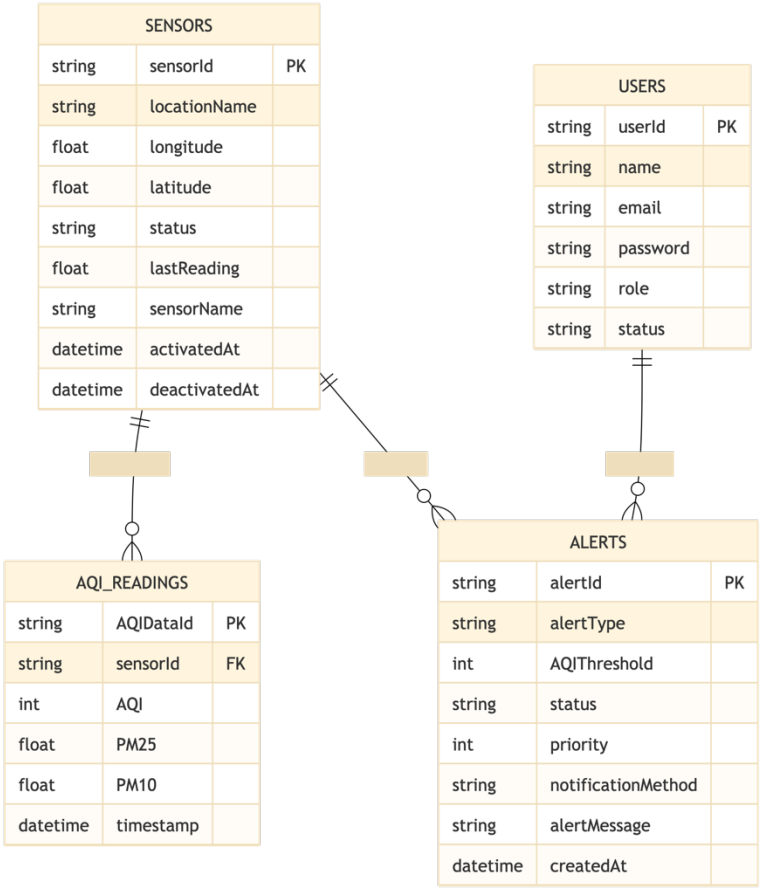
The **Real-time Air Quality Monitoring Dashboard** follows a client-server architecture with multiple interacting components:

1. **Sensors (Simulated Data Generator)**
   - A separate **C# console application** generates random air quality data.
   - The console app sends the data to the **ASP.NET Core Web API**, mimicking simulated sensor readings.
2. **Backend API (ASP.NET Core)**
   - Handles data processing, storage, and retrieval.
   - Provides RESTful API endpoints for the frontend to fetch data.
   - Stores sensor data, user details, and historical trends in the database.
3. **Database (MS SQL Server)**
   - Stores air quality readings, user roles, and other application data.
   - The backend fetches and updates data from the database when required.
4. **Frontend (HTML, CSS, JavaScript)**
   - Provides a user-friendly interface for viewing real-time and historical AQI data.
   - Sends API requests to fetch air quality data and displays it in graphical form.
   - Integrates mapping APIs for geographic data visualization.
5. **Mapping API (Leaflet.js)**
   - Displays AQI levels on a map with markers.
   - Shows real-time air quality data for different locations.
6. **User Roles & Interaction**
   - **Public Users**: View AQI data and trends.
   - **System Admin**: Manage sensors, alerts, and users.

System Architecture Diagram



ER Diagram for the Database



Test Case Design

To ensure the quality and reliability of the Real-Time Air Quality Monitoring Dashboard, a comprehensive set of test cases was designed, covering both backend and frontend components of the system. The test cases focused on **core functional areas** including authentication, sensor data processing, alert generation, administrative operations, and user interface behavior.

Tested Features and Components:

- **Admin Login/Logout**
- **Sensor Management (Add, Update, Activate/Deactivate)**
- **Live Data Simulation and Display**
- **Alert Configuration and Triggering**
- **Map Visualization of Sensor Locations**
- **User Management**

Test Design Approach:

Test cases were designed using a mix of **equivalence partitioning**, **boundary value analysis**, and **decision table testing**to cover various valid and invalid inputs and system states. This ensured both typical and edge-case scenarios were tested.

Examples of Test Scenarios:

Test Case ID	Description	Input	Expected Output
TC01	Login with valid admin credentials	Valid email and password	Redirect to Admin Dashboard
TC02	Login with invalid credentials	Incorrect password	Display error message
TC03	Add new sensor with valid details	Name, Location, Status = Active	Sensor appears in sensor list
TC04	Deactivate an active sensor	Status toggle to off	Sensor marked as inactive
TC05	Simulate data with AQI = 305	AQI = 305	Alert triggered for hazardous air quality
TC06	Edit alert threshold to AQI > 200	Threshold = 200	System triggers alerts only above AQI 200
TC07	View sensor location on map	Click map view	Map displays sensor markers at correct coordinates
TC08	Add new admin user	Name, Email, Role	New admin appears in user list
TC09	Delete an alert configuration	Select alert, click delete	Alert removed from the list
TC10	Stop data simulation	Toggle simulation OFF	No new data updates appear

This test case design helped verify that all critical workflows were functioning correctly under both expected and unexpected conditions.

Mock Objects

In the testing phase of the Real-Time Air Quality Monitoring Dashboard, mock objects were strategically used to simulate external dependencies and to isolate components for more effective testing, especially within the backend system developed using **C# ASP.NET Core Web API**.

Simulated Sensor Data (Mock Service)

Instead of relying on real-time sensor hardware, a **custom C# console application** was developed to act as a **mock data generator**. This application simulated realistic air quality data and sent it to the backend API at configurable intervals. The console app mimicked the behavior of actual sensors by sending parameters such as AQI values, timestamps, and sensor IDs to the system's endpoints. This mock service allowed the team to:

- Test how the system handles **continuous data updates**
- Validate **alert triggering mechanisms** based on AQI thresholds
- Simulate **high-load conditions** by increasing data frequency

Mocking in Unit Tests

For backend unit testing, **mock objects** were used to simulate:

- **Database interactions** (e.g., retrieving or saving sensor and user data)
- **Authentication services**, where mock tokens and users were created to test protected endpoints

Tools such as **Moq** and **xUnit** were used to create mock repositories and service layers, allowing isolated unit testing without relying on the actual database or live services. This ensured:

- **Faster test execution**
- **Higher reliability** by avoiding side effects from database states
- Easier detection of bugs in business logic

Frontend Integration

Although **Leaflet.js** was used for real-time map visualization, mock sensor data (from the simulation app) was used during frontend testing to ensure that sensor markers appeared correctly and updated in real time based on backend responses.

Unit Tests and Integration Tests

During the development of the Real-Time Air Quality Monitoring Dashboard, testing was primarily conducted manually by executing and validating each feature in real-time using the actual system components.

Testing Approach

Rather than setting up a separate automated test project, we adopted an **incremental and integrated testing approach**, where features were tested immediately after implementation. This method allowed rapid feedback, continuous validation, and faster debugging throughout development.

Tools Used for Testing

- **Postman**: Used to manually test API endpoints, simulate different request types (GET, POST, PUT, DELETE), and verify correct responses from the backend.
- **Browser Console & Dev Tools**: Used to inspect network requests, console logs, and map behavior (via Leaflet.js) on the frontend.
- **Live Sensor Simulation**: Our custom C# console application was used to generate and send live AQI data to the system, enabling realistic testing of:
    - Real-time sensor updates
    - Alert triggering
    - Map visualizations and dashboard updates

How Tests Were Run

1. **API Testing**: After developing a backend feature, it was immediately tested using Postman to confirm expected behavior, status codes, and data integrity.
2. **Frontend Validation**: Once the backend functionality was confirmed, the corresponding frontend interface was checked for correct rendering, data binding, and user interaction.
3. **System Integration Testing**: The full system, including simulated sensor data and admin functionalities, was run locally to ensure that components worked seamlessly together.

While no automated test framework was used, this hands-on approach ensured thorough coverage of the core system workflows and allowed developers to fix issues instantly as they were identified.

Functional Test Plans

To ensure that the system met all critical functional requirements, we conducted a series of **manual functional tests** that validated end-to-end workflows across all major features of the dashboard. These tests were designed to confirm that each component of the system behaves as expected under normal operating conditions.

Key Functional Requirements Tested:

1. **Admin Login and Logout**
2. **Dashboard Overview Visualization**
3. **Sensor Management (Add, Edit, Activate/Deactivate)**
4. **Sensor Location Visualization on Map**
5. **Simulated AQI Data Handling**
6. **Alert Configuration (Add, Edit, Delete Alerts)**
7. **User Management (Add, Edit, Delete Admins)**
8. **Logout Functionality and Session Management**

Functional Test Cases and Scenarios:

Feature	Test Case Description	Expected Outcome
Login	Admin logs in with correct credentials	Redirected to dashboard
Login	Admin logs in with incorrect password	Error message shown
Logout	Click logout from dashboard	Redirected to login page
Sensor Management	Add a new sensor with valid inputs	Sensor appears in the list
Sensor Management	Edit a sensor's location/status	Sensor details updated successfully
Sensor Management	Deactivate a sensor	Sensor status changes to inactive
Map View	View sensor map on dashboard	Map loads with all active sensors shown
Data Simulation	Start simulation using toggle	AQI values begin updating in real time
Data Simulation	Adjust frequency and base AQI	Data changes according to new settings
Alert Config	Add a new AQI alert (e.g., AQI > 200)	Alert appears in the list
Alert Config	Edit an existing alert threshold	Alert threshold updates correctly
Alert Config	Delete an alert	Alert is removed from the system
User Management	Add a new admin user	New user appears in user list
User Management	Edit an existing admin's role/status	User details update successfully
User Management	Delete an admin user	User is removed from list

Test Methodology:

All functional test cases were **executed manually** by interacting with the system through:

- The **web interface** for admins
- **Postman** to validate API responses directly
- The **sensor data simulation console app** to emulate real-time input

This approach allowed us to validate full workflows and simulate real user interactions, ensuring the reliability of both frontend and backend components.

Test Strategy & Critical Analysis

Test Strategy Overview

Our test strategy was primarily focused on **manual functional and integration testing**, given the nature of the system, which involves real-time data simulation, user interaction, and API-driven components. The goal was to validate key workflows across the entire system, from sensor data ingestion to visualization and alert handling.

We prioritized **integration testing** over traditional unit testing because:

- The system's core functionality depends on **real-time AQI data updates** from a simulated C# console app.
- Features like alerts, sensor updates, and dashboard summaries rely heavily on **interconnected components** (API ↔ database ↔ frontend).

Why This Approach Was Chosen

- The system was under active development with evolving requirements.
- Manual and integrated testing allowed us to **quickly verify changes** and **observe behavior live**.
- It ensured **accurate validation** of the end-to-end flow, especially for visual elements like maps and sensor lists.

What We Focused On

- **End-to-end feature workflows**: Such as login/logout, sensor creation, and alert handling.
- **Real-time data handling**: Verifying that simulated data is received and displayed correctly.
- **User role management**: Ensuring only admins could perform certain actions.
- **System feedback**: UI messages, validation messages, and real-time updates on the dashboard.

Challenges Faced

- **No automated test suite**: All tests had to be manually executed, which limited test repetition and increased time spent on regression testing.
- **Real-time simulation complexity**: Ensuring timing and data frequency from the simulation app matched frontend expectations.
- **Data consistency**: Simulated values sometimes caused unexpected visual behavior (e.g., overlapping map markers or alert spikes).
- **No formal mocking setup**: We tested against the actual database and services, making it harder to isolate issues.

Areas Thoroughly Tested

- Functional behavior of all admin features (sensor management, alert creation, login/logout).
- Real-time updates and interactions between the simulation app and the dashboard.
- Visual behavior of the Leaflet.js map, sensor markers, and dynamic data updates.

Areas Not Fully Tested

- **Edge cases**: Such as extremely high-frequency data simulation or invalid/malformed sensor data.
- **Security testing**: Authentication was tested for correct login, but deeper penetration or role-based access control tests were not conducted.
- **Performance testing**: Load testing or stress testing for high numbers of sensors/data points was not included due to time and tooling constraints.

Conclusion

In conclusion, the testing process for the Real-Time Air Quality Monitoring Dashboard was focused on ensuring the robustness and reliability of key system functionalities. The approach, primarily centered around manual functional testing and integration testing, was well-suited to validate the interconnected components of the system, particularly given the real-time data simulation and live API interactions.

We thoroughly tested critical workflows such as login/logout, sensor management, alert configurations, and user management, ensuring that each feature worked end-to-end. The system's ability to handle simulated real-time data, coupled with the dynamic visualization on the dashboard, was central to our validation efforts.

Despite the absence of automated testing tools, we employed practical, hands-on testing using Postman and the C# console app for data simulation to ensure seamless interaction between the backend and frontend. While we did not test edge cases or conduct performance testing due to time constraints, the system’s core functionalities were thoroughly validated under normal operating conditions.

The testing strategy was effective in identifying and resolving critical issues early, providing confidence that the system will meet user expectations in real-world deployment. Moving forward, implementing automated testing and further stress testing will be beneficial for long-term maintenance and scalability.

Work Matrix Breakdown

ID	Name(As stated in DLE)	Tasks Carried Out
10952727	Pulindu Nadil	Database Design & Integration Alert Configuration
10952711	Bedde Damiru	Sensor Data Simulation App User Management Data Simulation Dashboard Overview Sensor network map Documentation & Report Writing
10953054	Liyanage Dasanayake	Login Page Frontend & Backend Alert Configuration
10952509	Amarasinghe Senevirathne	Public Dashboard frontend– Realtime air quality map Sensor Pop-up info Sensor Charts
10952649	Hasiru Jayakody	Public Dashboard backend – Realtime air quality map Sensor Pop-up info Sensor Charts
10953022	S.M.R.H. Senevirathna	Sensor Management Dashboard Overview live status fields & Sensors List