**BSE25-5**

# Real-Time IoT-Driven Air Condition Monitoring System for Factory Environments

## Software Design Document

**TEAM:** Kevin Mugarura B (22/U/6325), Alanda Ambrose (21/U/1392), Kirabo Jelly Rollings (21/U/08437/PS), Musiimenta Cynthia (21/U/05922/PS)

# Table of Contents

# LIST OF TABLES

# List of Figures

# 1 INTRODUCTION

## 1.1 Purpose

This Software Design Document (SDD) describes the architecture and system design of the Real-Time IoT-driven Air Condition Monitoring System for Factory Environments to ensure good health for workers. The solution integrates IoT devices, MySQL database, cloud-based data processing (ThingSpeak), and web-based dashboard with notifications to provide actionable insights into critical air pollutants (e.g., CO, PM2.5, VOCs) and environmental parameters (e.g., temperature, humidity) in industrial settings. By outlining the modules and data structures, this document guides developers, testers, project managers, regulatory bodies, and factory administration teams in understanding how the system is built, deployed, and maintained.

## 1.2 Scope

The system shall continuously monitor air quality in factory environments and alert when pollutant concentrations exceed predefined safety thresholds. Core functionalities include:

- Air Quality Data Collection: IoT devices measuring temperature, humidity, VOCs, CO, PM2.5, and PM10 at regular intervals.
- Cloud-Based Analysis: ThingSpeak integration stores sensor readings and enables broader data processing or visualization if needed.
- Proactive Alerts & Notifications: Buzzer alerts and web-based notifications.
- Role-Based User Management: Control who can view, acknowledge, and configure system parameters (e.g., thresholds).
- Compliance Reporting: Generation of PDF reports to satisfy both internal audits and external regulatory requirements.
- MySQL Database: A relational database that stores user-related data and sensor-aggregated records for analysis.

### 1.2.1 Objectives

- Develop an IoT platform for real-time monitoring of air conditions in factories to enhance worker health and productivity.
- Provide notifications to staff upon detecting hazardous pollutant levels on the web interface and alerts by the buzzers.
- Enhance historical data analysis for person exposure to contaminated air to understand trends and optimize air control mechanisms.
- Produce compliance reports for management and authorities.

### 1.2.2 Benefits

- Worker Health & Productivity: Safe, stable air quality shall correlate with reduced health incidents and higher morale.
- Immediate Hazard Awareness: The system shall reduce response times and help avoid accidents or health risks for workers working in factories.
- Data-Driven Decision-Making: Visualization and trend analysis shall inform better ventilation and filtration strategies.
- Regulatory Compliance: Automated reporting and logging shall simplify audits and compliance checks.

## 1.3 Overview

This SDD is organized as follows:

- Section 1 – Introduction, including purpose, scope, references, and definitions.
- Section 2 – System Overview, summarizing the system's core operations.
- Section 3 – System Architecture, detailing how IoT devices, ThingSpeak, and the web dashboard integrate, plus rationale.
- Section 4 – Data Design, describing the primary data structures, tables, and entities.
- Section 5 – Component Design, explaining both IoT (functional) and web (object-oriented) modules.
- Section 6 – Human Interface Design, presenting UI layouts, screen flows, and user interactions.
- Section 7 – Requirements Matrix, linking functional requirements from the SRS to specific components.
- Section 8 – Appendices containing diagrams, instructions, or references.

## 1.4 Reference Material

- IEEE Std 1016-1998 – Recommended Practice for Software Design Descriptions.
- System Requirements Specification (SRS) – Real-Time IoT-driven Air Quality Monitoring.
- ThingSpeak Official Documentation – Cloud endpoints, authentication, data handling.
- Sensor Datasheets (e.g., MQ-135 for VOC detection, MH-Z19 for $CO_2$).

## 1.5 Definitions and Acronyms

- VOC: Volatile Organic Compound.
- PM2.5/PM10: Particulate Matter measuring 2.5 / 10 micrometers in diameter.
- CO: Carbon monoxide
- IoT: Internet of Things.
- API: Application Programming Interface
- LED: Light-Emitting Diode
- DHT: Digital Humidity and Temperature.
- GSM: Global System for Communication
- ERD: Entity Relationship Diagram

# 2 SYSTEM OVERVIEW

## 2.1 Overview

The Real-Time IoT-Driven Air Condition Monitoring System shall collect and analyze environmental data to help individuals maintain healthy indoor conditions. Its core objective is to provide a comprehensive device that measures air quality in real-time, not merely for monitoring but subsequent long-term trend analysis. Workers shall use the device in factories to attain the following goals: -

## 2.2 Key Goals

- Integrated Data Collection: The system shall read sensor data (including CO, PM2.5, PM10, VOCs, Methane, temperature, humidity, and other parameters) at regular intervals. This approach will ensure comprehensive coverage of common pollutants and environmental factors that could affect an individual's health in factory spaces.
- Cloud Processing & Storage: After collecting data, the system shall transmit measurements to the ThingSpeak cloud platform using a GSM connection. This enables secure off-site storage and more robust data processing, minimizing local resource usage on the device itself.
- Data Analysis & Reporting (Web Dashboard): A web-based dashboard will consolidate the analytics and reporting components into a single module, enabling both short-term alerts and long-term data analysis. Users will have access to interactive visualizations, real-time notifications, and automated report generation to assess indoor air quality over time.
- Alerts: Should the sensor readings surpass designated safety thresholds, local (buzzer/LED) shall be activated. This prompt feedback loop aims to protect individuals from prolonged exposure to harmful airborne substances, thus mitigating respiratory risks and other health concerns.
- Factory Environment Application: This solution will be adaptable for any industry setting, offering individuals insight into air quality conditions that could impact long-term health.
- Efficient Power Management: The system shall include a low-power mode feature, reducing unnecessary transmissions or continuous polling of sensors when there are no significant changes. This approach enhances battery life, ensuring the system remains active for extended periods without frequent recharging or maintenance.

## 2.3 How It Will Work

- Data Collection: Sensor devices in a room or factory setting will capture key pollutants and environmental factors.
- Local Threshold Checks: If any sensor reading exceeds a configured limit, a buzzer or LED shall warn the user immediately.

- Data Transmission & Cloud Storage: The system shall send sensor data to ThingSpeak at set intervals, ensuring access to the readings from anywhere.
- Web Dashboard Analysis & Reports: Data will be fetched from ThingSpeak and then processed within the Analytics & Reporting module on the web dashboard. Users will examine metrics, generate compliance or health-focused reports, and receive notifications if thresholds are exceeded.
- Health Impact & Long-Term Trends: By enabling trend analysis and historical data visualization, users or managers will make more informed decisions about ventilation improvements, scheduling equipment checks, or adjusting occupancy levels to maintain a healthier environment.

# 3 SYSTEM ARCHITECTURE

## 3.1 Architectural Design

The system shall comprise IoT devices, the ThingSpeak cloud platform, a web dashboard, and MySQL database integration. Each subsystem shall carry distinct responsibilities yet collaborate to form a complete air quality monitoring workflow.

IoT System

- Monitoring: Periodically read pollutant levels for CO, PM2.5, PM10, VOC, Methane, Combustible gasses temperature and humidity.
- Alert: Trigger buzzer/LED whenever readings exceed safety thresholds.
- Power: A 1200mAh Lipo battery to supply power to the GSM communication module.
- Communication: Send data to ThingSpeak (via GSM).
- SD card: Temporary data storage in case of network interruptions.

ThingSpeak (Cloud Platform)

- Data Reception: Collect sensor data in near real-time.
- Security & Scalability: Maintain secure endpoints and can scale to numerous factory zones.

Web Dashboard

- Analysis: Integrate sensor data from the database that is fetched from Thingspeak to compute statistics (min, max, average) and trends.
- Notifications: Issue alert web notifications whenever set thresholds are reached.
- Reporting: Generate PDF compliance or managerial reports based on selected date ranges.
- User Management: Maintain role-based access for managers and maintenance personnel.

MySQL Database Integration

- Data Storage: Preserve user accounts, alerts, and sensor data for deeper queries.
- Compliance Retention: Retain historical data essential for audits or legal purposes.

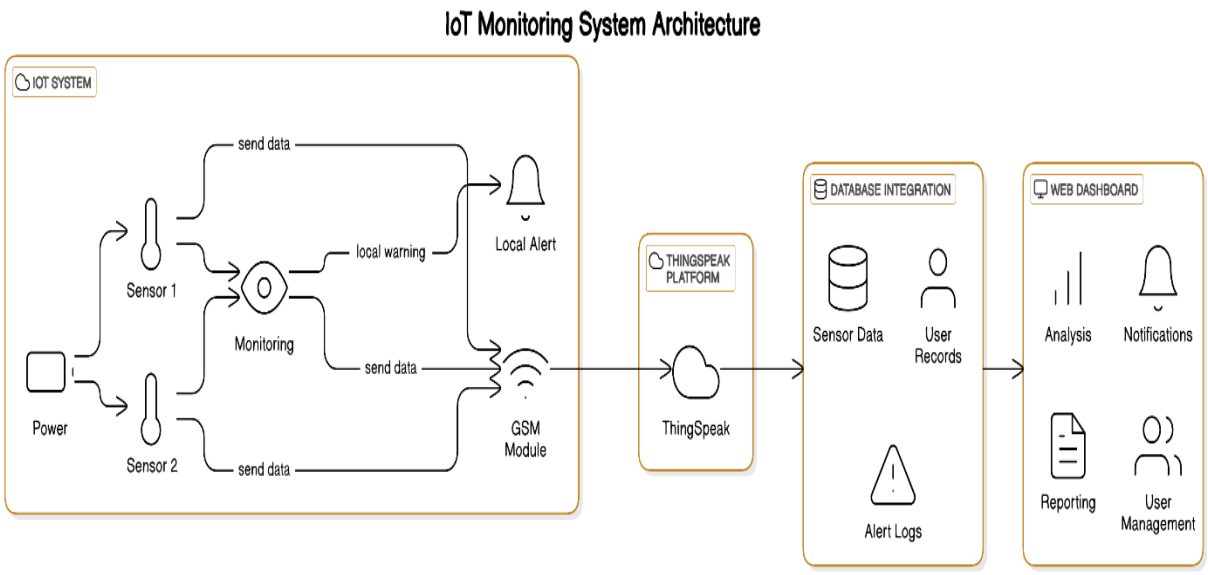


*Figure 3.1.1 System Architectural Diagram*

In Figure 3.1.1, the system shall continuously monitor air quality in factory environments and deliver timely alerts and reports. IoT devices scattered throughout the facility by workers collect data on pollutants (CO, PM2.5, PM10, VOCs) and environmental parameters (temperature, humidity). This data is evaluated locally to trigger immediate alerts, activating buzzers when dangerous levels are detected, and is then transmitted via GSM to the ThingSpeak cloud platform for secure, real-time storage. The web dashboard shall retrieve and analyze this data, displaying real-time trends, sending web notifications, and enabling users to generate detailed compliance reports. A relational database shall store user records, alert logs, and reports, ensuring data redundancy and supporting regulatory audits. This integrated approach will ensure rapid response to hazardous conditions while providing robust historical data for decision-making and compliance.

## 3.2 Decomposition Description

- IoT Subsystem (Functional modules: Monitoring by IoT Devices, Alert, Power, Communication by GSM): Shall provide real-time data collection, local alerting, efficient power usage, and cloud communication via the GSM.
- ThingSpeak (Cloud Platform): Shall act as an intermediate data host, offering secure ingestion of the collected sensor data.
- Database Integration: Shall store vital records for compliance and auditing, ensuring the system can function using a relation MySQL database.
- Web Dashboard (Object-oriented modules: Analysis, Notifications, Reporting, User Management): Shall handle data processing, user interaction with the dashboard, web notifications, and reporting logic for the users of the system.

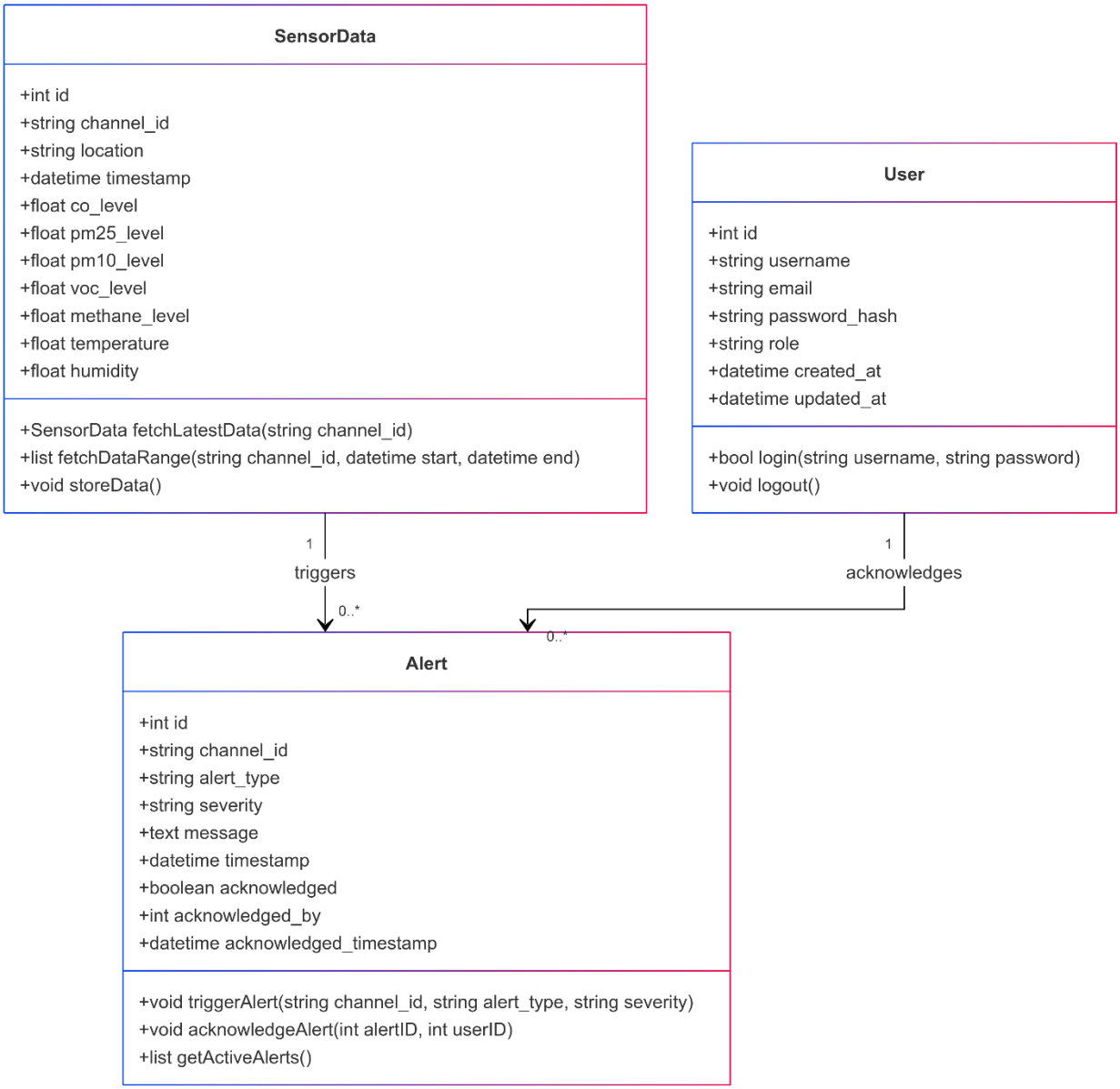


*Figure 3.2.1 Class diagram*

Figure 3.2.1 is a class diagram representing the system's structure by defining its classes, attributes, methods, and relationships between classes. The system includes key classes such as Sensor, Alert, and User. The Sensor class collects data from IoT devices, the Alert class triggers notifications, and the User class manages authentication and access control.

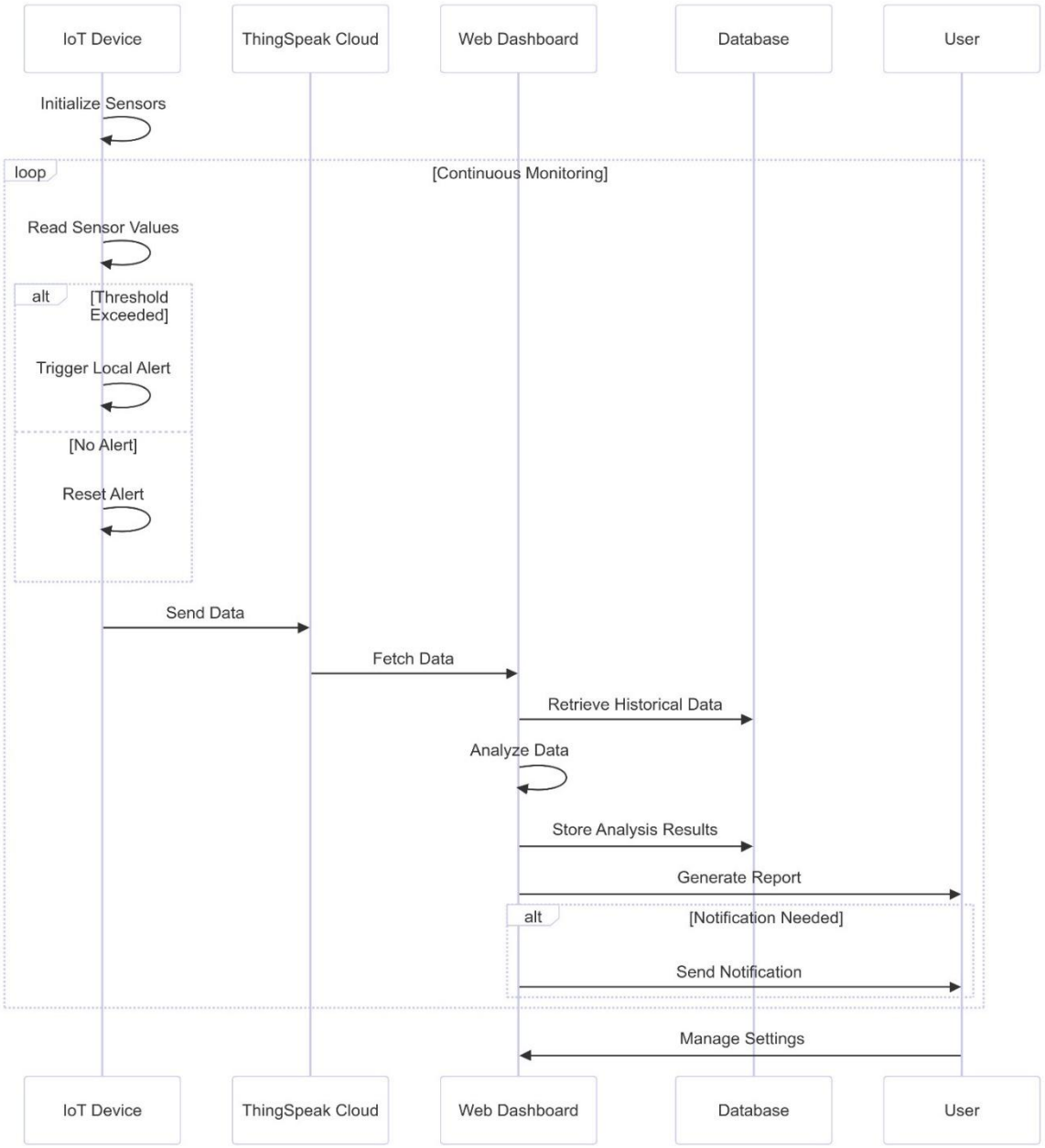


*Figure 3.2.2 Sequence diagram*

Figure 3.2.2 is a sequence diagram that shows the interaction between different system components over time. IoT devices collect factory air quality data and send it to the ThingSpeak cloud. The web dashboard periodically fetches data from ThingSpeak and stores relevant records in the MySQL database. If pollutant levels exceed predefined limits, the system sends notifications to users. The dashboard also analyzes data and allows users to generate compliance reports based on historical trends.

## 3.3 Design Rationale

- Modular IoT Functionality: Separating sensor monitoring, alerting, power management, and communication simplifies firmware development, enhances reliability, and optimizes energy usage.
- Scalable Web Dashboard: An object-oriented design for the web interface improves maintainability and scalability, ensuring clear separation of concerns for data analysis, notifications, reporting, and user management.
- Cloud Leverage via ThingSpeak: Utilizing ThingSpeak minimizes the need for custom cloud infrastructure while providing secure, real-time data ingestion and built-in analytics capabilities.
- Robust Data Redundancy: Integrating a local database ensures data redundancy, supports efficient querying, and meets compliance requirements by maintaining comprehensive audit trails and historical records.

# 4 DATA DESIGN

## 4.1 Data Description

The system shall employ a relational MySQL database to manage three primary categories of information:

- User Credentials & Roles: Ensures secure login and enforces role-based access (e.g., manager, worker).
- Sensor Readings: Captures time-series data from IoT devices (CO, PM2.5, PM10, VOCs, Methane, temperature, humidity), accompanied by timestamps and channel identifiers.
- Alerts: Records threshold breaches or system notifications, including who acknowledged each alert.

Although ThingSpeak shall store raw sensor readings in near real-time, the MySQL database shall mirror or selectively archive critical records to provide redundancy, enable faster queries, and support advanced trend analysis or compliance audits.

Integration and Relationships

- User (table 4.2.4) has a one-to-many relationship with Alerts (Alerts table) since a single user may acknowledge multiple alerts.
- SensorData (table 4.2.5) references the channel ID to map each reading to a specific sensor device and also the location.
- Alerts reference both SensorData (indirectly, via the channel ID) and the User (AckowledgedBy foreign key).
- Additional tables (not shown or not mandatory) may be introduced if the system requires advanced grouping of sensor channels, custom threshold settings per user, or more detailed logging.

## 4.2 Data Dictionary

Below are two sets of entities. The first set provides a concise summary of each entity, while the second set expands on the attributes, data types, constraints, indexing considerations, and relationships.

### 4.2.1 Summary of Tables and Descriptions

User

*Table 4.2.1 User Attribute Descriptions*

| Field | Description |
|---|---|
| UserID | Unique identifier for each user. |
| Username | User's login name. |
| Email | User's email address. |
| PasswordHash | Securely hashed password using bcrypt. |
| Role | User role (e.g., Manager, Worker). |
| CreatedAt | Timestamp of user record creation. |
| UpdatedAt | Timestamp of the last update to the user record. |

Table 4.2.1 stores user-related information, including authentication credentials, roles, and timestamps for account creation and updates.

SensorData

*Table 4.2.2 SensorData Attribute Descriptions*

| Field | Description |
|---|---|
| DataID | Unique ID for each sensor reading. |
| ChannelID | Identifier for the channel. |
| Location | The location where the sensor device is placed. |
| Timestamp | When the reading was taken. |
| CO | Carbon Monoxide (ppm). |
| PM2_5 | Particulate Matter 2.5µm. |
| PM10 | Particulate Matter 10µm. |

| | |
|---|---|
| VOC | Volatile Organic Compounds (ppm). |
| Methane | Methane (ppm). |
| Humidity | Humidity as a percentage (%). |
| Temperature | Temperature in °C or °F. |

Table 4.2.2 records real-time environmental sensor readings, including pollutants, humidity, and temperature, with timestamps and location tracking.

Alerts

*Table 4.2.3 Alerts Attribute Descriptions*

| **Field** | **Description** |
|---|---|
| AlertID | Unique identifier for each alert. |
| ChannelID | Sensor channel that triggered the alert. |
| AlertType | Type of alert (e.g., ThresholdBreach). |
| Severity | Severity level (Low, Medium, High). |
| Message | Detailed description of the alert condition. |
| Timestamp | When the alert was generated. |
| Acknowledged | Whether the alert has been marked as resolved. |
| AcknowledgedBy | The user acknowledged the alert. |
| AcknowledgedTimestamp | When the alert was acknowledged. |

Table 4.2.3 logs system alerts triggered when sensor readings exceed predefined thresholds. It tracks alert severity, acknowledgment status, and user actions.

### 4.2.2 Detailed Attribute Definitions with Constraints

Below is a more comprehensive breakdown, highlighting data type details, indexing, and constraints to ensure optimal query performance and referential integrity.

User

*Table 4.2.4 Users Attribute Definitions and Constraints*

| Field | Data Type | Description | Constraints/Indexes |
|---|---|---|---|
| UserID | INT (PK, AUTO_INCREMENT) | Unique identifier for each user. | PRIMARY KEY, auto-increments. |
| Username | VARCHAR(100) UNIQUE | User's login name | UNIQUE INDEX to prevent duplicates. |
| Email | VARCHAR(255) UNIQUE | User's email address. | UNIQUE INDEX to prevent duplicates. |
| PasswordHash | VARCHAR(255) | Securely hashed password using bcrypt. | Must not be null, with no leading/trailing spaces. |
| Role | VARCHAR(50) | e.g., "Manager", "Worker", "Maintenance". | Possibly CHECK or enumerated constraint. |
| CreatedAt | DATETIME | Timestamp of user record creation. | Default to CURRENT_TIMESTAMP if needed. |
| UpdatedAt | DATETIME | Timestamp of the last update to the user record. | On update, can auto-update to the current time. |

Table 4.2.4 defines user-related information, including unique user identification (UserID), authentication credentials (Username, Email, PasswordHash), and user roles (Role). It ensures secure access through unique constraints on Username and Email while maintaining timestamps (CreatedAt, UpdatedAt) for tracking user activity. Passwords are securely hashed for confidentiality.

SensorData

*Table 4.2.5 SensorData  Attribute Definitions and Constraints*

| **Field** | **Data Type** | **Description** | **Constraints/Indexes** |
|---|---|---|---|
| DataID | INT (PK, AUTO_INCREMENT) | Unique ID for each sensor reading. | PRIMARY KEY, auto-increments. |
| Location | VARCHAR(100) | Locations where the sensor device is placed | The location should be a room UNIQUE number. |
| ChannelID | VARCHAR(100) | Identifier for the channel. | INDEX if queries frequently filter by channel. |
| Timestamp | DATETIME | When the reading was taken. | INDEX to facilitate time-range queries. |
| CO | DECIMAL(5,2) NULL | Carbon Monoxide (ppm). | May allow NULL if sensor not installed. |
| PM2_5 | DECIMAL(5,2) NULL | Particulate Matter 2.5µm. | Same constraint style as CO. |
| PM10 | DECIMAL(5,2) NULL | Particulate Matter 10µm. | Same constraint style as CO. |
| VOC | DECIMAL(6,2) NULL | Volatile Organic Compounds (ppm). | Extended precision if needed. |
| Methane | DECIMAL(6,2) NULL | Methane (ppm). | Extended precision if needed. |
| Humidity | DECIMAL(5,2) NULL | Humidity as a percentage (%). | e.g., 0.00 to 100.00. |
| Temperature | DECIMAL(5,2) NULL | Temperature in °C or °F. | e.g., -50.00 to 120.00. |

Table 4.2.5 stores real-time environmental sensor readings, including CO, PM2.5, PM10, VOC, Methane, Humidity, and Temperature. Each record is linked to a ChannelID and Location, with timestamps ensuring chronological tracking. The table uses indexed fields for optimized queries, supporting efficient data retrieval for analytics and threshold evaluations.

Alerts

*Table 4.2.6 Alerts Attribute Definitions and Constraints*

| **Field** | **Data Type** | **Description** | **Constraints/Indexes** |
|---|---|---|---|
| AlertID | INT (PK, AUTO_INCREMENT) | Unique identifier for each alert. | PRIMARY KEY, auto-increments. |
| ChannelID | VARCHAR(100) | Sensor channel that triggered the alert. | INDEX to correlate with SensorData.ChannelID. |
| AlertType | VARCHAR(50) | ThresholdBreach | Possibly enumerated (ThresholdBreach, etc.). |
| Severity | VARCHAR(20) | e.g., "Low," "Medium", or "High". | Possibly enumerated (Low, Medium, High). |
| Message | TEXT | Detailed description of the alert condition. | It can store optional user-friendly text. |
| Timestamp | DATETIME | When the alert was generated. | INDEX for chronological sorting. |
| Acknowledged | BOOLEAN | Whether the alert has been marked as resolved. | Defaults to FALSE. |
| AcknowledgedBy | INT (FK -> Users.UserID) | The user acknowledged the alert. | FOREIGN KEY referencing Users (UserID). |
| AcknowledgedTimestamp | DATETIME NULL | When the alert was acknowledged. | Updates upon acknowledgment if Acknowledged=TRUE. |

Table 4.2.6 records system alerts generated when sensor readings exceed predefined thresholds. Each alert includes an AlertID, ChannelID, AlertType, Severity, and a detailed Message. Alerts can

be Acknowledged, with tracking of the responsible user (AcknowledgedBy) and timestamped logs (AcknowledgedTimestamp). This table ensures traceability and system responsiveness to air quality hazards.

Additional Considerations

- Indexes & Performance: ChannelID, Timestamp, and UserID commonly serve as indexes to speed up queries related to time-series analysis, user actions, or device channels.
- Data Integrity: Foreign key constraints (e.g., Alerts(AcknowledgedBy) -> Users(UserID)) ensure referential integrity.
- Check constraints or enumerated fields to help maintain data consistency for attributes like Severity and AlertType.
- Scalability: The system may partition the SensorData table by ChannelID or by date to handle large volumes of time-series data efficiently, especially in high-volume industrial setups.
- Security & Privacy: Sensitive user data (like PasswordHash) must be encrypted or hashed.
- Access to the MySQL database should be restricted by robust credentials and SSL connections for data in transit.
- Data Retention & Archival: Historical sensor readings could grow significantly, so an archiving strategy (e.g., moving old sensor data to cheaper storage) might be necessary if the real-time analysis window is limited.

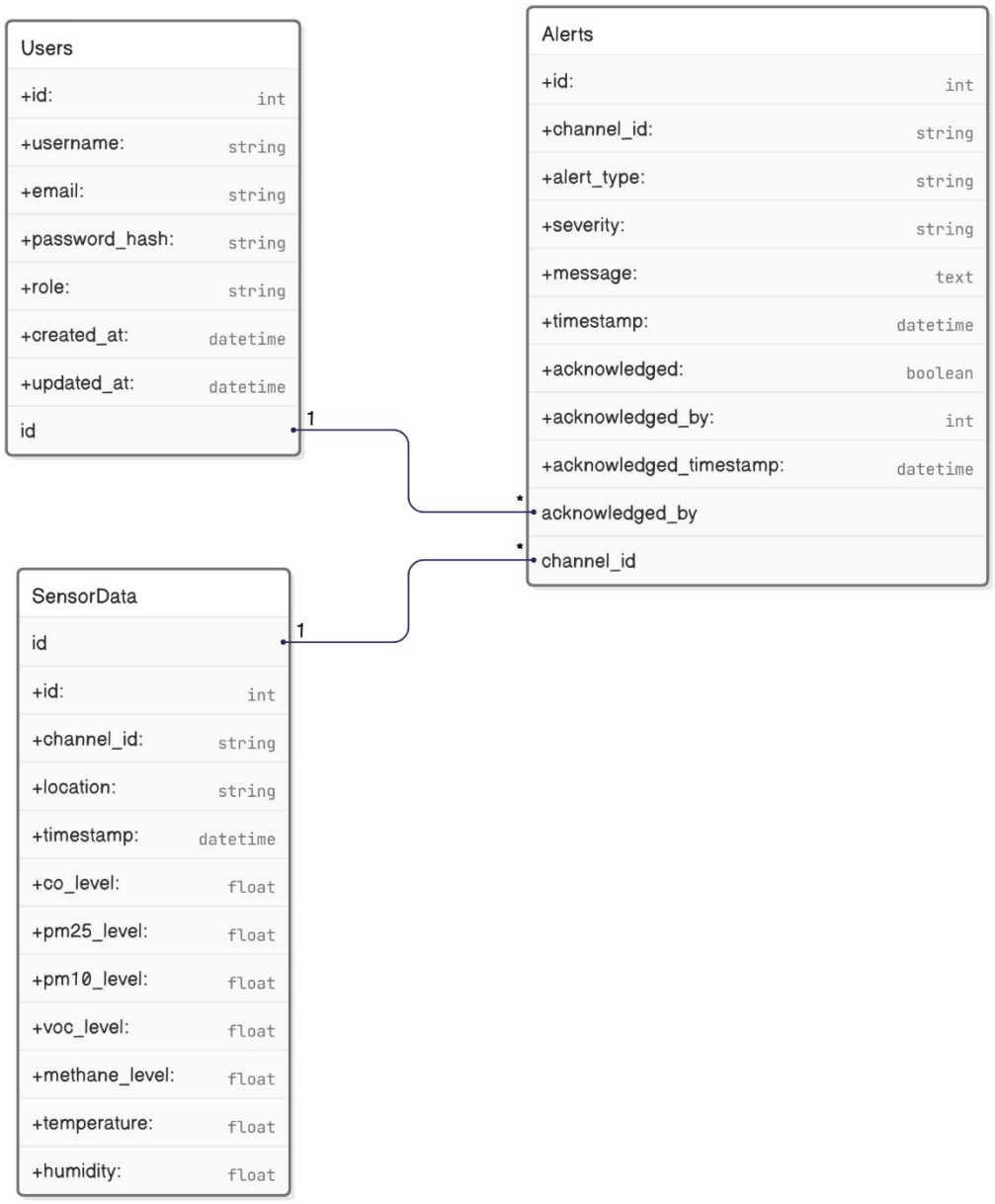


*Figure 4.2.1 Entity Relationship Diagram*

Figure 4.2.1 is an ERD showing how tables relate to each other. It defines entities, attributes, and the relationships between them (one-to-one, one-to-many, many-to-many), helping in database design and organization. It defines key tables such as Users, SensorData, and Alerts, showing how they interact. The SensorData table stores air quality readings collected by IoT devices, while the Alerts table logs threshold breaches. The Users table links to the Alerts table, allowing role-based acknowledgment of system notifications.

# 5 COMPONENT DESIGN

This section details the functionality and interactions of two major components in the system:

- Data Collector & Local Alerting
- Analytics, Reporting & Notifications

These components work together to provide real-time sensor data collection, threshold-based alerts, advanced air quality analysis, and secure user management.

### DATA COLLECTOR & LOCAL ALERTING

The IoT device collects sensor readings, checks against thresholds, manages power consumption, and sends data to the cloud (ThingSpeak). It also provides local alerts (buzzer or LED) for immediate on-site warnings.

**Data Collector**

Continuously read sensor values (e.g., CO, Methane, PM2.5, PM10, VOC, temperature, humidity), compare them to safe thresholds, and trigger local alerts if a reading exceeds its limit. Any valid data is then forwarded for remote transmission.

Process Overview:

- Initialization: Calibrate and initialize all sensors.
- Data Acquisition: In a continuous loop, read sensor values.
- Threshold Check & Alerting: If any sensor value surpasses its threshold, activate a buzzer or switch the LED to red. If values remain safe, ensure alerts are off (LED green).
- Data Forwarding: Forward the sensor readings to the communication module, which sends them to ThingSpeak.

Pseudocode

```
FUNCTION runDataCollector():
  initializeSensors()
  WHILE true:
    reading = readSensorValues()     // e.g., CO, Methane, PM2.5, etc.
    alertTriggered = false
    For each pollutant IN reading:
      IF reading[pollutant] > threshold[pollutant]:
```

```
            triggerLocalAlert()      // buzzer on, LED red
            alertTriggered = true
            BREAK
        END IF
    END FOR
    IF NOT alertTriggered:
        resetLocalAlert()          // buzzer off, LED green
    END IF
    forwardDataForTransmission(reading)
    wait(samplingInterval)         // e.g., 30 seconds
  END WHILE
END FUNCTION
```

**Power Management & Communication**

- Power Management ensures the device only transmits or runs sensors frequently when necessary, conserving battery life in low-activity periods.
- Communication packaging and sending sensor data to ThingSpeak via GSM.

Pseudocode (Power Management)

```
FUNCTION powerManagementLoop():
  previousData = readSensorValues()  // Initial baseline
  WHILE true:
    currentData = readSensorValues()
    // Check for significant changes or alerts
    IF isAlertCondition(currentData) OR hasSignificantChange(currentData, previousData):
      wakeUpTransmitter()
      sendDataToCloud(currentData)
      previousData = currentData
    ELSE:
      enterLowPowerState()
```

```
    wait(samplingInterval)
  END WHILE
END FUNCTION
```

Pseudocode (Communication)

```
FUNCTION sendDataToCloud(reading):
  payload = encodeToJSON(reading)
  response = sendHTTPPost(thingSpeakURL, payload)
  logResponse(response)
  IF response contains configurationUpdates THEN
    updateLocalThresholds(response.config)
  END IF
END FUNCTION
```

## ANALYTICS, REPORTING & NOTIFICATIONS

The web dashboard component provides data analytics, report generation, notifications, and user management. It retrieves sensor data from ThingSpeak or the local database, computes aggregated metrics, dispatches alerts to remote users, and controls who can access which features.

### Analytics & Reporting

Purpose: Aggregate sensor data (either from ThingSpeak or the MySQL database) to produce real-time and historical insights. When needed, the system generates compliance or managerial reports in PDF format, summarizing air quality trends over a specified date range.

Key Methods: -

- fetchData(startTime, endTime): Retrieves sensor data for a given period and queries the local database or ThingSpeak's APIs.
- calculateStatistics(dataSet): Computes key metrics such as minimum, maximum, and average pollutant levels.
- generateReport(userID, startTime, endTime): Fetches data from the Analysis module, formats the data into a report document, logs the report creation event, and returns a file path or URL.

Pseudocode (Combined Analysis & Reporting)

```
CLASS AnalyticsEngine:
    METHOD fetchData(startTime, endTime):
        // Query local DB or ThingSpeak for sensor data in this range
        return querySensorData(startTime, endTime)
    METHOD calculateStatistics(dataSet):
        // Compute average, min, max for each pollutant
        return computeMetrics(dataSet)
    METHOD generateReport(userID, startTime, endTime):
        dataSet = this.fetchData(startTime, endTime)
        IF dataSet is empty:
            return "No data available"
        // Generate PDF from dataSet
        reportFile = createDocument(dataSet, "PDF")
        logReportCreation(userID, startTime, endTime, reportFile)
        return reportFile
END CLASS
```

**Notifications & User Management**

- Notifications: Dispatch web app notifications if the analysis or threshold detection indicates a problem.
- User Management: Handle user authentication, authorization, and role assignments, ensuring only authorized personnel can modify thresholds or view sensitive data.

Key Methods (Notifications)

```
CLASS NotificationCenter:
    METHOD dispatchAlert(alertDetails):
        recipients = findRecipients(alertDetails)
        FOR each recipient in recipients:
            sendNotification(recipient, alertDetails)
```

```
    END FOR
    logNotificationDispatch(alertDetails)
    return "Alerts sent"
END CLASS
```

Key Methods (User Management)

```
CLASS UserAuth:
  METHOD login(username, password):
    userRecord = findUserByUsername(username)
    IF userRecord exists AND verifyPassword(password, userRecord.PasswordHash):
      return createSession(userRecord)
    ELSE:
      return "Invalid credentials"
  METHOD assign role(userID, newRole):
    // Update the user's role in DB
    updateUserRole(userID, newRole)
    return "User role updated successfully"
END CLASS
```

Summary of Interaction

- Data Collector & Local Alerting handles sensor sampling, threshold checks, local buzzer/LED alerts, and data transmission.
- Analytics, Reporting & Notifications (on the web) fetches sensor data, calculates statistics, generates PDF reports, and sends notifications.
- User Management ensures secure logins and permissions, restricting critical operations (e.g., editing thresholds, viewing advanced analytics) to designated roles (Manager, Administrator, etc.).

This two-component approach ensures a clean separation of concerns, data acquisition, and immediate local responses on the IoT side, and remote analytics, reporting, notifications, and user control on the web.

# 6 HUMAN INTERFACE DESIGN

## 6.1 Overview of User Interface

The web-based user interface is designed to support the following key functionalities:

- Real-Time Air Quality Monitoring & Alerts: The dashboard displays live sensor data for air quality (including CO, $CO_2$, PM2.5, PM10, VOC, Methane, temperature, and humidity). It highlights current conditions and immediately visualizes any breaches of safety thresholds. In the event of an alert, the interface shows clear notifications that prompt immediate attention.
- Analytics for Air Quality Trends & Notifications: The system analyzes historical trends through interactive charts and graphs summarizing key metrics over selectable time intervals. This analytical view allows users to identify patterns and anomalies. Additionally, a dedicated notifications area ensures that any detected trends or sudden changes trigger remote alerts via web app notifications.
- Report Generation for Air Quality Data: Users can generate detailed compliance or operational reports by specifying date ranges and sensor types. These reports, available in PDF format, consolidate real-time and historical data, enabling data-driven decision-making and supporting regulatory audits.

## 6.2 Screen Images

- Main Dashboard: Figure 6.2.1 is a screen that shows real-time sensor readings in gauges or charts, with visual cues for any alerts.

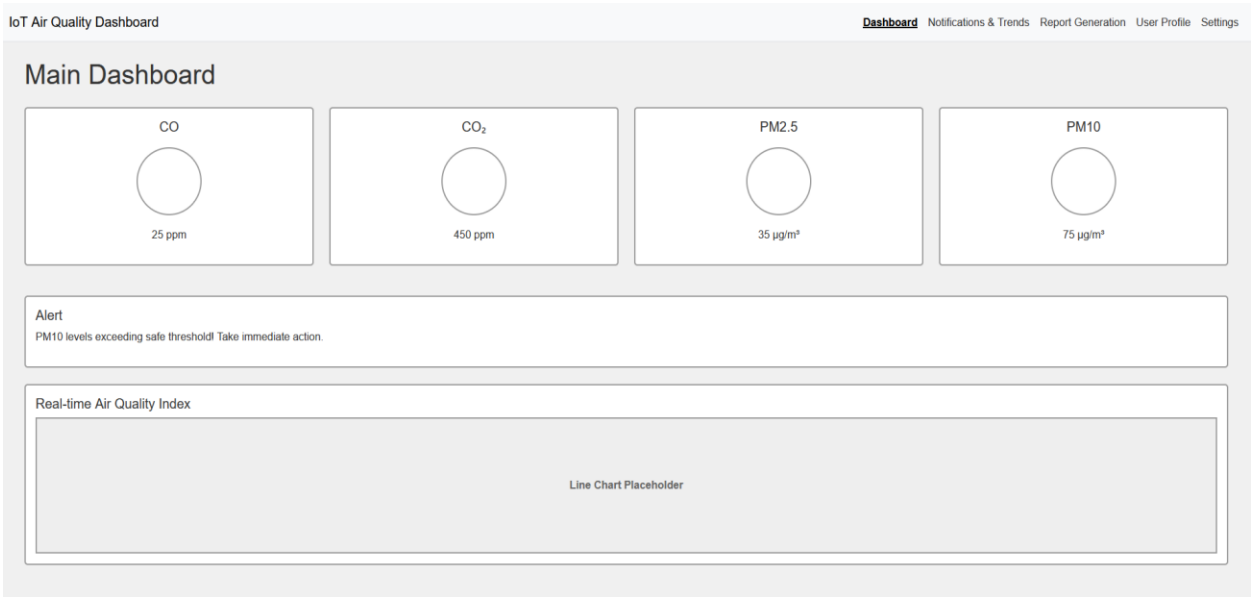


*Figure 6.2.1 Main Dashboard*

- Notifications & Trends: Figure 6.2.2 shows a dedicated section that displays historical trend graphs alongside a list of system notifications alerting users of past events or threshold breaches.

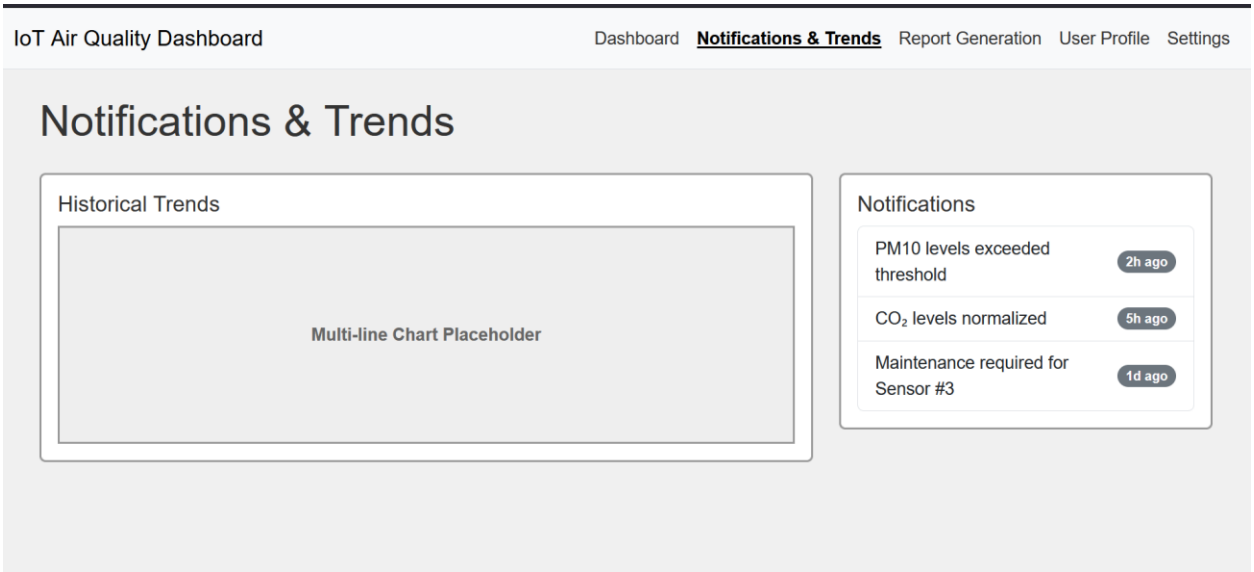


*Figure 6.2.2 Notification & Trends*

- Report Generation Interface: Figure 6.2.3 is a form where users select a date range and sensor categories to generate and download compliance reports.

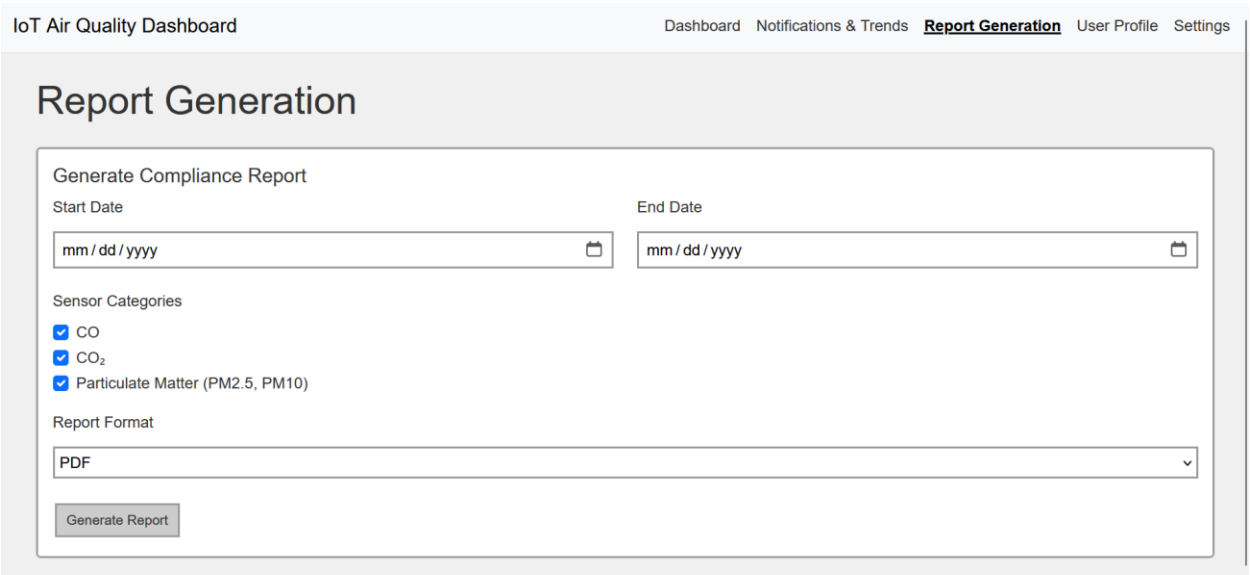


*Figure 6.2.3 Report Generation Interface*

## 6.3 Screen Objects and Actions

- Real-Time Gauges/Charts: Interactive elements that update in real-time to display current sensor values. Clicking on a gauge expands the view to show detailed historical data for that metric.
- Alerts/Notifications Area: A section where active alerts are displayed. Each alert includes an "Acknowledge" button that updates its status and logs the action in the system.
- Report Generation Form: Includes date pickers and sensor selection options; submitting the form triggers the report generation process (via the Report Manager) and presents a download link for the report.
- Navigation Menu/Sidebar: Quick links provide easy access to the Dashboard, Alerts, Reports, and User Management sections.

# 7 REQUIREMENTS MATRIX

*Table 7.1 Requirements Matrix*

| Req. ID | Requirement | Related Components |
|---|---|---|
| REQ-1 | The system shall collect air quality data from IoT sensors. | IoT Subsystem: Monitoring – Sensor initialization, data acquisition, and threshold checks for pollutant levels. |
| REQ-2 | The system shall process and display air quality data on a user interface. | Web Dashboard: Analysis & Data Visualization – Responsible for real-time data processing and visualization. |
| REQ-3 | The system shall update the dashboard with new data. | Web Dashboard: Data Visualization Components – Components that dynamically refresh sensor data on the monitoring interface. |
| REQ-4 | The system shall store air quality data collected from the sensors. | IoT Subsystem: Communication – Transmits sensor data to ThingSpeak cloud storage. |
| REQ-5 | The system shall analyze historical data to predict air quality trends. | Web Dashboard: Analysis Module – Computes trends and predictive analytics based on stored sensor data. |
| REQ-6 | The system shall display air quality trends on the dashboard. | Web Dashboard: Analysis & Data Visualization Components – Generates and presents historical trends and forecasts. |
| REQ-7 | The system shall notify users when predicted air quality is expected to exceed safety limits. | Web Dashboard: NotificationCenter – Sends alerts based on analyzed data and predefined thresholds. |
| REQ-8 | The system shall trigger alerts when pollutant levels exceed safety thresholds. | IoT Subsystem: Monitoring (Local Alerting) – Activates local alerts (buzzer, LED) when thresholds are breached. |
| REQ-9 | The system shall send alerts via web app notifications based on user preferences. | Web Dashboard: NotificationCenter – Handles notification delivery based on user settings. |
| REQ-10 | The system shall log all sent alerts for future review. | Web Dashboard: NotificationCenter & Logging Mechanism – Stores all dispatched alerts for auditing. |
| REQ-11 | The system shall retry sending alerts if the initial attempt fails. | Web Dashboard: NotificationCenter (Error Handling & Retry Logic) – Ensures alert delivery through retry mechanisms. |
| REQ-12 | The system shall allow users to select specific periods for report generation. | Web Dashboard: ReportManager (User Interface Controls) – Enables users to set time ranges for reports. |

| | | |
|---|---|---|
| REQ-13 | The system shall generate reports in PDF format. | Web Dashboard: ReportManager (Report Generation & Formatting) – Formats reports into standard document types. |
| REQ-14 | The system shall store generated reports for future access. | Web Dashboard: ReportManager (Report Storage Integration) – Archives reports in the system for retrieval. |
| REQ-15 | The system shall notify the user when the report is ready for download. | Web Dashboard: ReportManager & NotificationCenter – Notifies users upon report generation completion. |

# 8 APPENDICES

**Appendix A**: Hardware Schematics

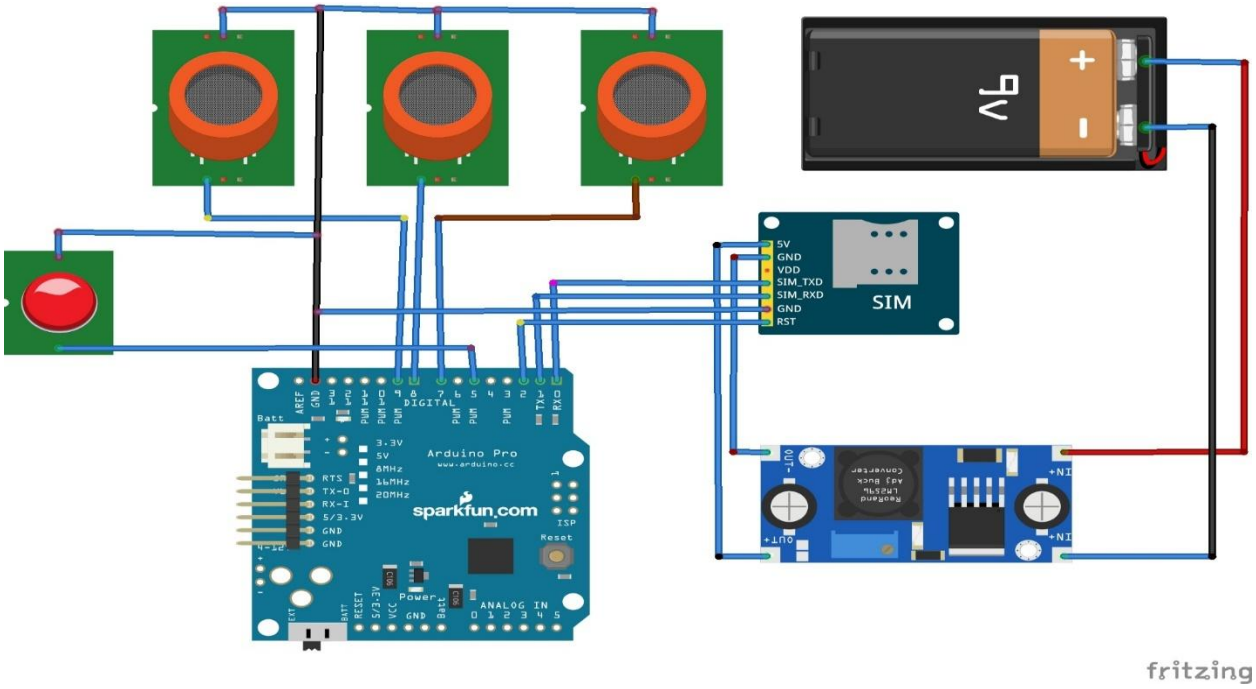


*Figure 8.1 Hardware Schematic for the IoT System*


*Figure 8.2 MQ-135 Gas Sensor*


*Figure 8.3 Arduino Microcontroller*

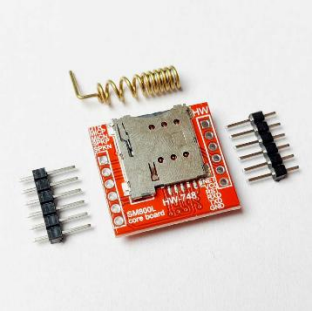

*Figure 8.4 GSM Module*

*Figure 8.5 Buzzer*

Descriptions for each figure

- Figure 8.1: Hardware Schematic for the IoT System: This diagram represents the complete hardware setup for the IoT-based air quality monitoring system. It illustrates the interconnections between various components, including sensors, a microcontroller, a GSM module for communication, a power supply, and alert mechanisms such as a buzzer and LEDs.
- Figure 8.2: MQ-135 Gas Sensor: The MQ-135 gas sensor is used to detect harmful air pollutants such as ammonia, benzene, $CO_2$, and other volatile organic compounds (VOCs). It continuously monitors air quality and transmits readings to the microcontroller for processing.
- Figure 8.3: Arduino Microcontroller: The Arduino microcontroller serves as the central processing unit for the IoT system. It collects real-time sensor data, evaluates threshold conditions, triggers alerts, and transmits readings to the cloud using the GSM module.
- Figure 8.4: GSM Module: The GSM module enables wireless data transmission from the IoT device to the cloud-based ThingSpeak platform. It ensures remote access to sensor readings and allows for real-time monitoring and notifications.
- Figure 8.5: Buzzer: The buzzer acts as an alert mechanism in the system. It produces an audible sound whenever air quality readings exceed predefined safety thresholds, ensuring immediate attention to hazardous conditions.

## Appendix C: Testing & Validation Procedures

The testing and validation of the Real-Time IoT-Driven Air Condition Monitoring System ensures the IoT firmware and web dashboard components function correctly. The process involves unit testing for firmware, integration testing for system components, and validation through real-world deployment.

Verifying IoT Firmware Logic: The IoT firmware is tested for: -

- Alert Mechanism – Simulating pollutant level changes to confirm buzzer and LED alerts activate correctly and logs are updated.
- Sensor Monitoring & Accuracy – Calibrating sensors and testing data sampling intervals to ensure reliability.
- Communication Reliability – Simulating network failures to verify data transmission to ThingSpeak resumes properly.
- Power Management – Testing power consumption in different states (normal, standby, alert mode) to optimize battery life.

Integration Testing for Web Dashboard: The web dashboard is tested to ensure smooth interaction between its modules:

- Data Analysis – Verifying air quality data visualization.
- Notifications – Ensuring alerts are triggered correctly and displayed based on user roles.
- Report Generation – Checking PDF reports for accuracy in pollutant levels and timestamps.
- User Management – Testing authentication, authorization, and role-based access restrictions.

By systematically testing these components, the system ensures high performance, accuracy, and usability.

## Appendix D: Deployment & Installation Guidelines

Deploying the system requires configuring IoT devices, ThingSpeak cloud storage, a MySQL database, and the web dashboard.

ThingSpeak Setup & API Security

- Create a ThingSpeak account and set up channels for sensor data.
- Secure Write API Keys for device authentication and Read API Keys for dashboard data retrieval.
- Configure IoT devices to transmit sensor data to ThingSpeak with retry mechanisms for failed transmissions.

Database & Web Dashboard Deployment

- Database Setup – Install MySQL, create necessary tables (users, sensor data, alerts), and set up secure credentials.
- Web Dashboard Installation – Deploy the system on a server, install required dependencies, and configure API keys and database access.
- Hosting & Security – Use secure hosting with encrypted connections and access control mechanisms.

These steps ensure a secure and efficient deployment, enabling seamless air quality monitoring and analysis.