CS 331
Computer Networks

Smart Guard IOT-Based Environmental Monitoring System

Presentated by:

- Aryan Sahu
- Bhavik Patel
- Hitesh Kumar
- Jinil Patel

Introduction

- Smart Guard is an IoT-based environmental monitoring system designed for classrooms and labs.
- It collects real-time data on parameters like:
  - Temperature, Humidity and Light Intensity.
- Uses ESP32 microcontrollers connected to sensors to capture environmental data.
- Data is transmitted to a central server via HTTP and visualized on a React-based dashboard.
- Enables live monitoring, historical analysis, and alert notifications when conditions cross predefined thresholds.
- Designed to be modular, scalable, and easy to maintain, supporting multiple rooms and sensor types.

Objectives

Main Goal: Build a scalable, real-time IoT system for monitoring environment conditions in classroom and labs.

Specific Objectives:

- 🌡️ **Sensor Data Collection:** Use DHT22 (temp + humidity), BH1750 (light)
- 🔗 **Data Transmission:** Send sensor readings to server via HTTP
- 🗃️ **Database Storage:** Store readings in PostgreSQL for querying and analysis
- 💻 **Frontend Dashboard:** Visualize data using React + Chart.js
- 🚨 **Alert Notifications:** Trigger email alerts on threshold breaches
- 🧩 **Modular Design:** Easy to add new sensors or locations

System Architecture

4-Layered Architecture

🔌 • **Hardware Layer**

ESP32 Microcontroller + Sensors (DHT22, BH1750)

🌐 • **Backend Layer**

Node.js Server – Handles data reception, preprocessing, and storage with database

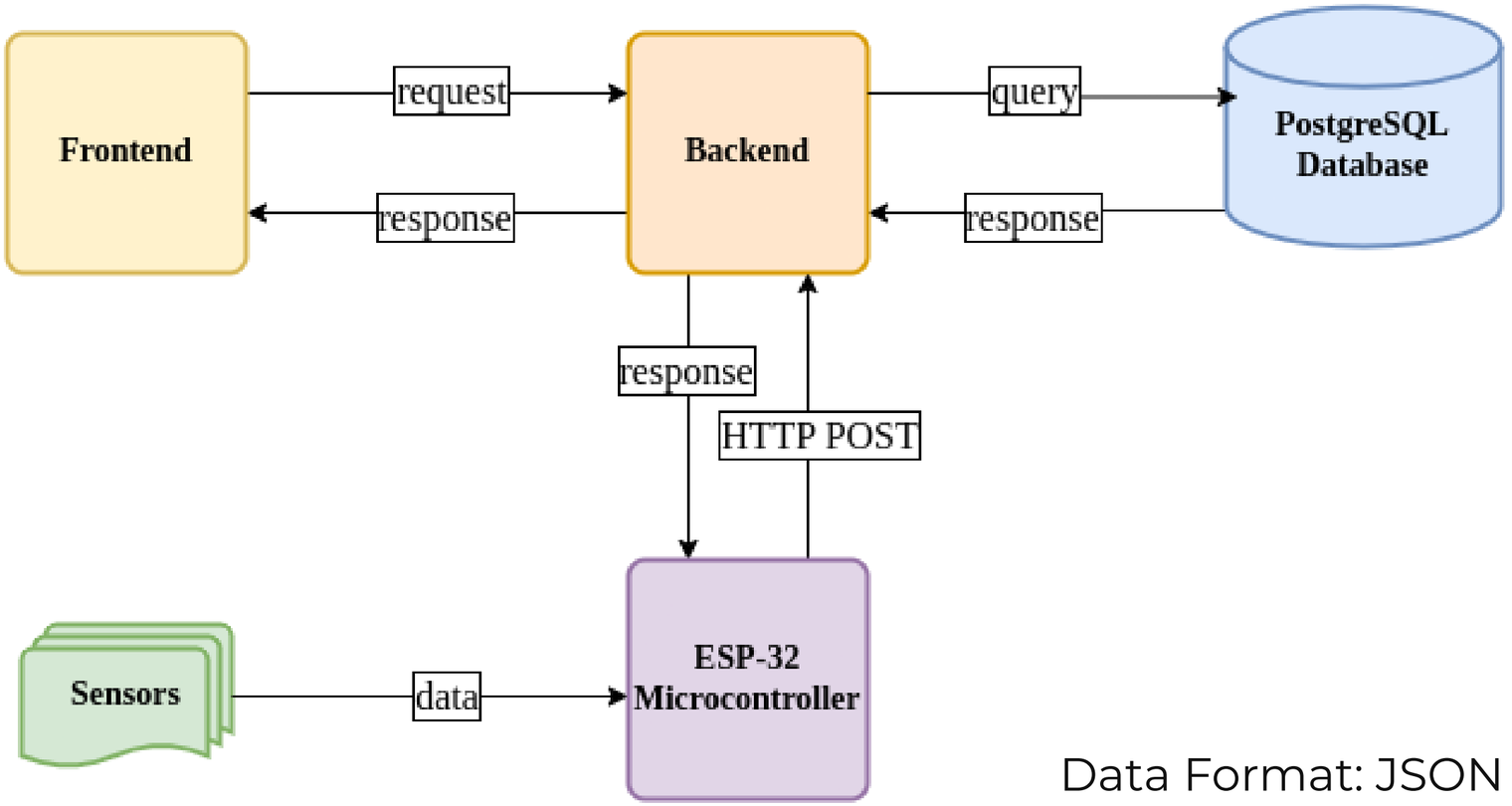
🗄️ • **Database Layer**

PostgreSQL – Stores time-series sensor data

📊 • **Frontend Layer**

React Dashboard – Visualizes real-time and historical data

System Architecture



Data Format: JSON

Frontend

🖼️ UI Features

- Room-based filtering
- Live graphs
- Date-range selection
- Alert Visualization

🧩 Designed for intuitive use and scalability.

⚙️ Key Technologies

- **React.js** – SPA for real-time interactivity
- **Chart.js** – Visualize sensor data using responsive graphs
- **Context API** – Global state management across components
- **Axios** – API calls to backend

⚙️ Built with:

React.js, Vite.js, Typescript

Backend

🛠️ **Key Responsibilities:**

- Receive sensor data via HTTP requests
- Validate & preprocess data
- Store it in PostgreSQL
- Expose REST APIs for frontend

📡 **Email Alert Integration:**

- Uses Nodemailer
- Sends emails when sensor thresholds are breached

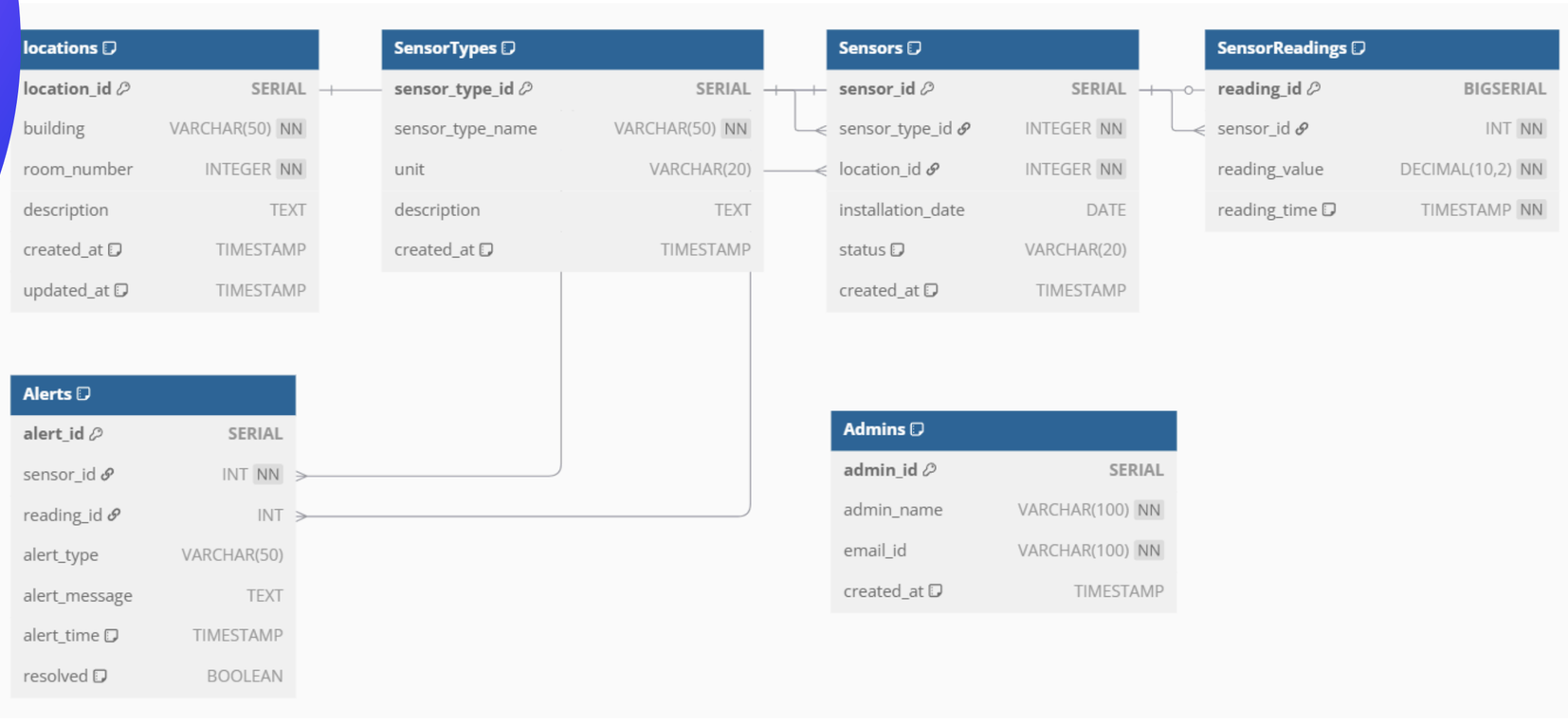
⚙️ **Built with:**

Node.js, Express.js, TypeScript

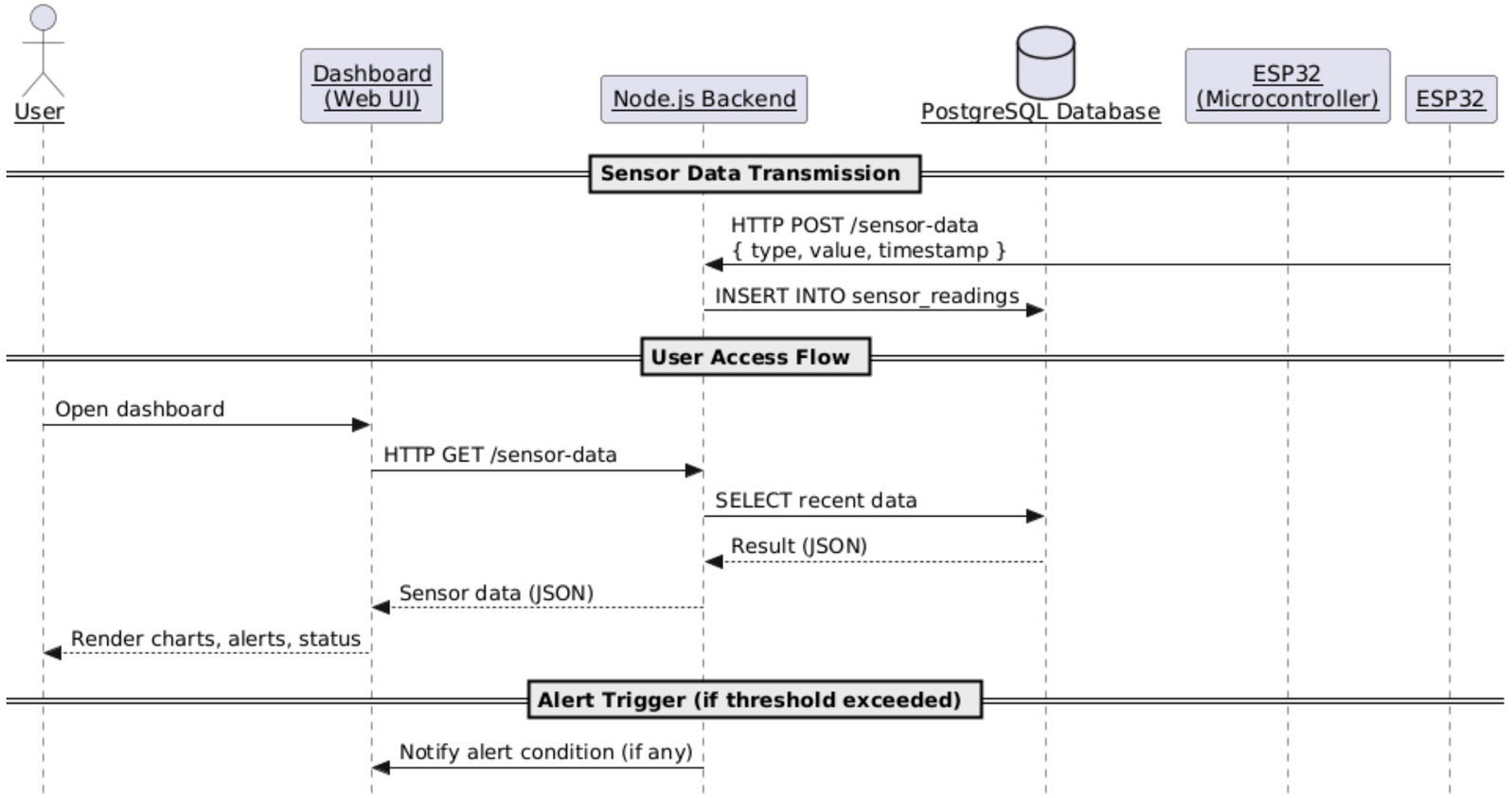
PostgreSQL Database

⚡ **Optimized for:**

- Time-series data
- Fast querying for historical trends
- Scalable storage

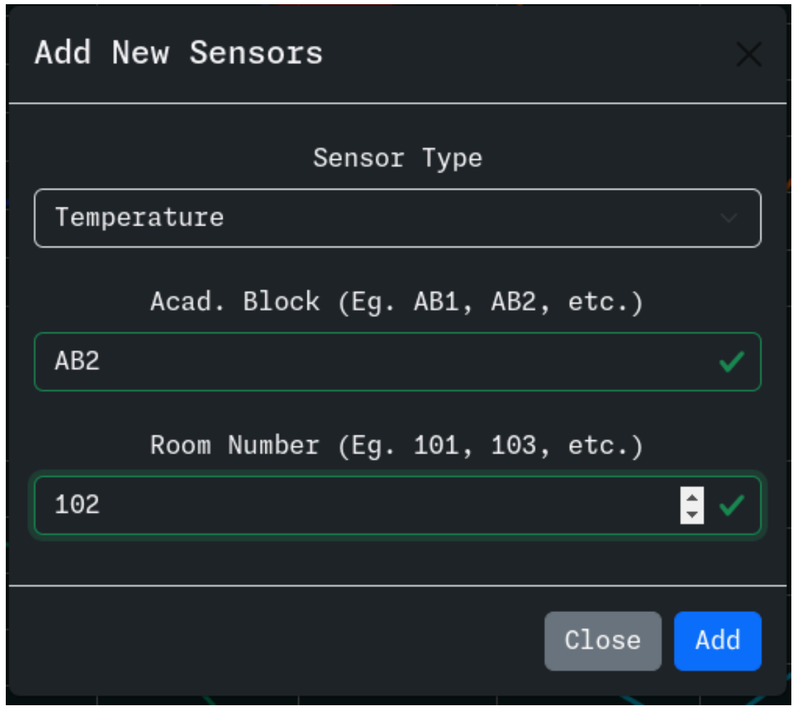


Data Flow


User
Dashboard (Web UI)
Node.js Backend
PostgreSQL Database
ESP32 (Microcontroller)
ESP32
Sensor Data Transmission
HTTP POST /sensor-data
{ type, value, timestamp }
INSERT INTO sensor_readings
User Access Flow
Open dashboard
HTTP GET /sensor-data
SELECT recent data
Result (JSON)
Sensor data (JSON)
Render charts, alerts, status
Alert Trigger (if threshold exceeded)
Notify alert condition (if any)

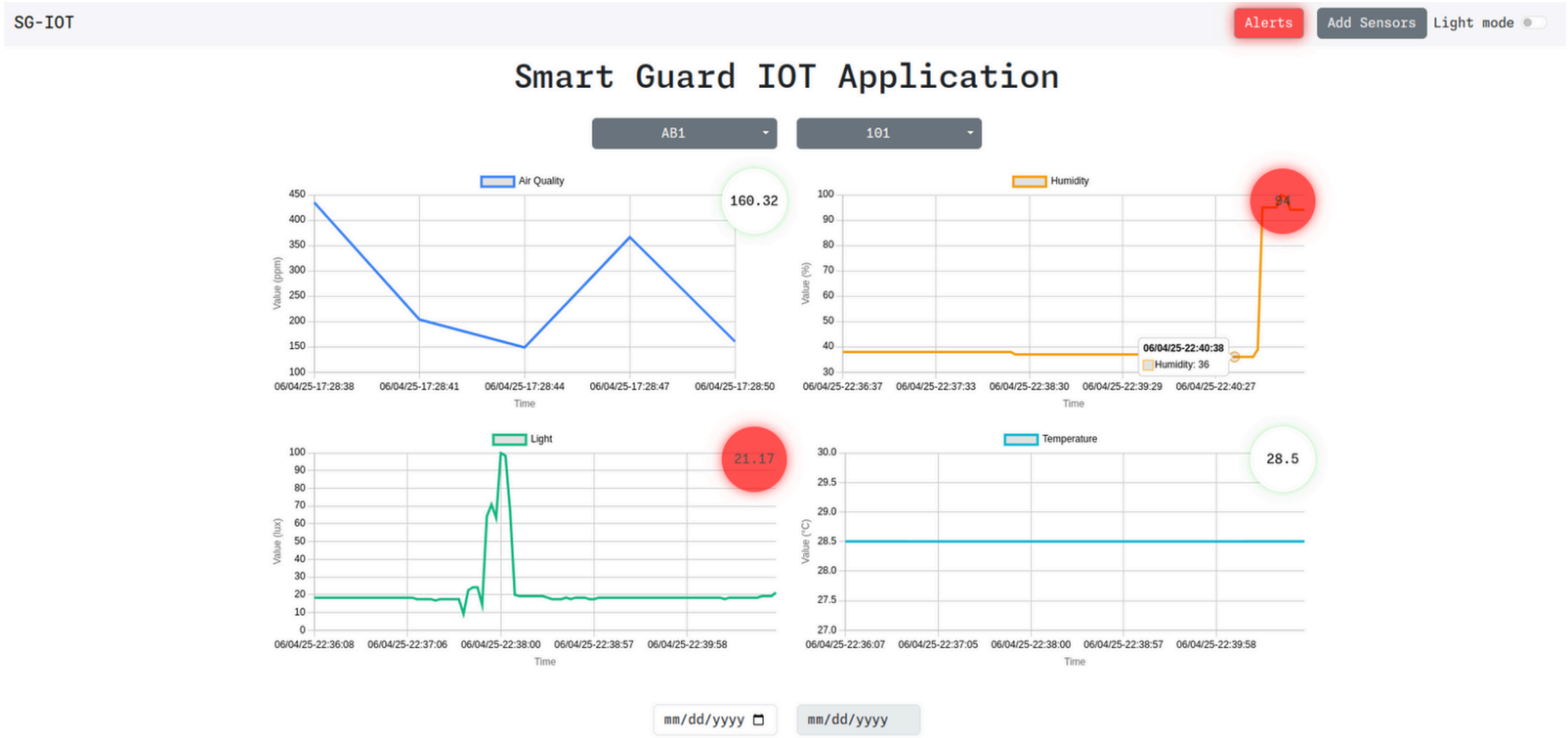
Objectives Achieved

Sensor Management:



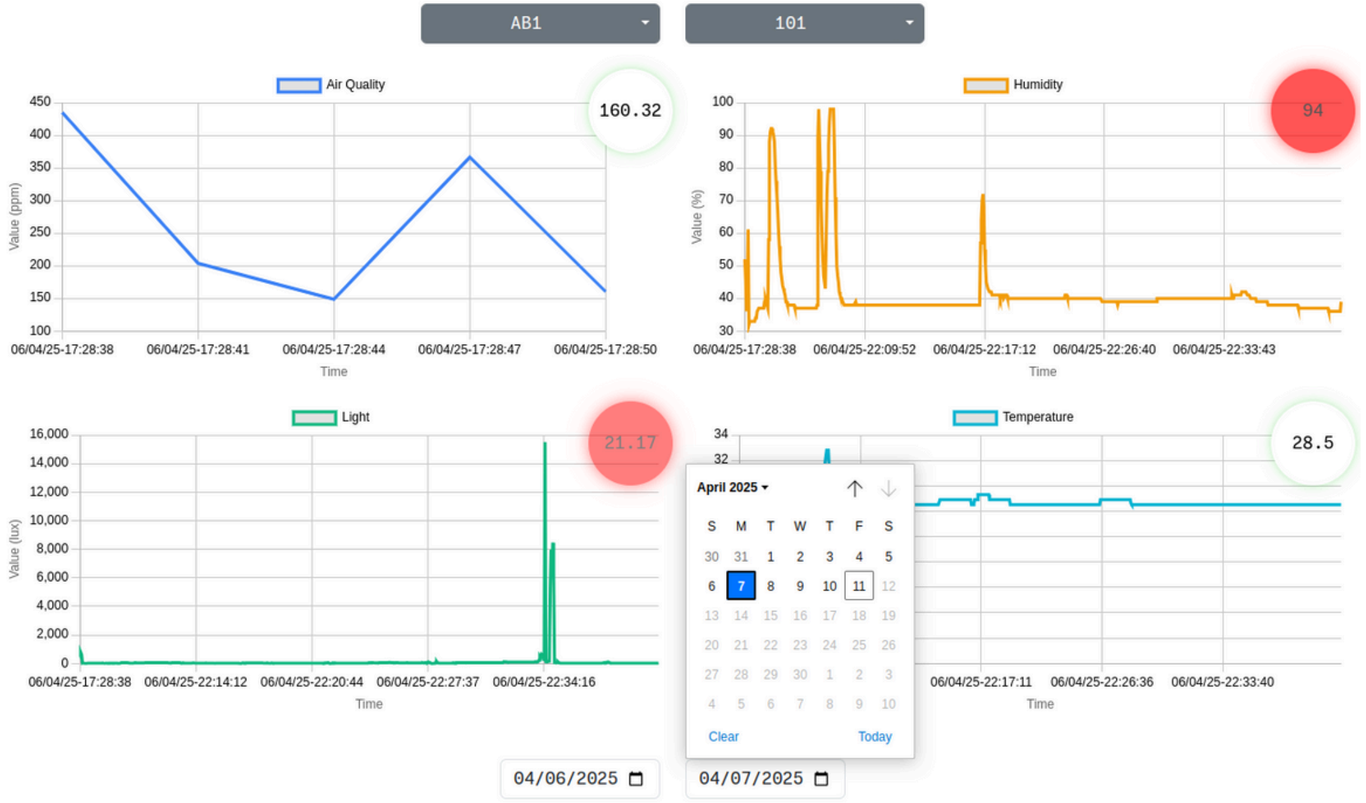
Objectives Achieved

Room-based Real-Time Data Access:



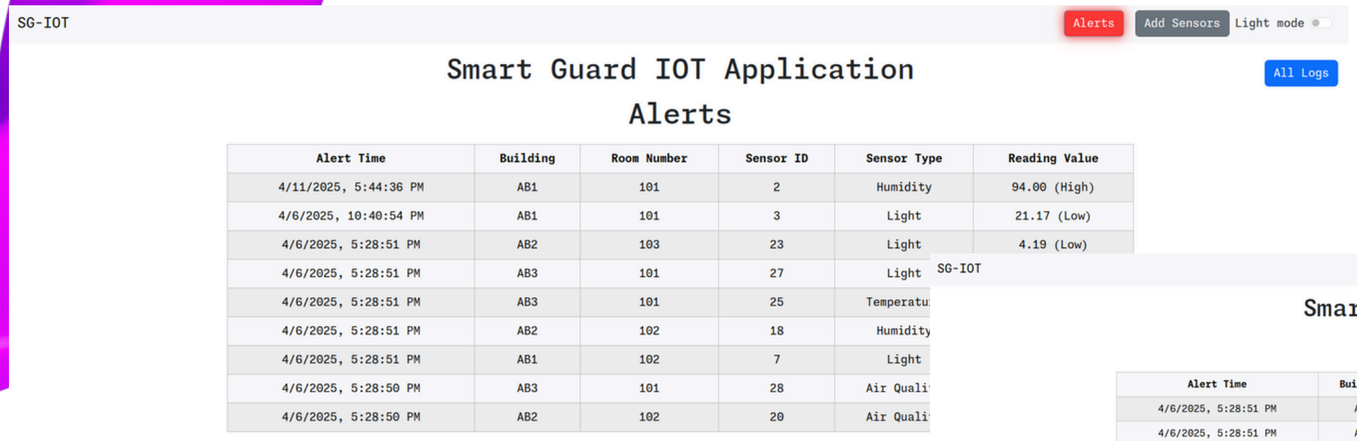
Objectives Achieved

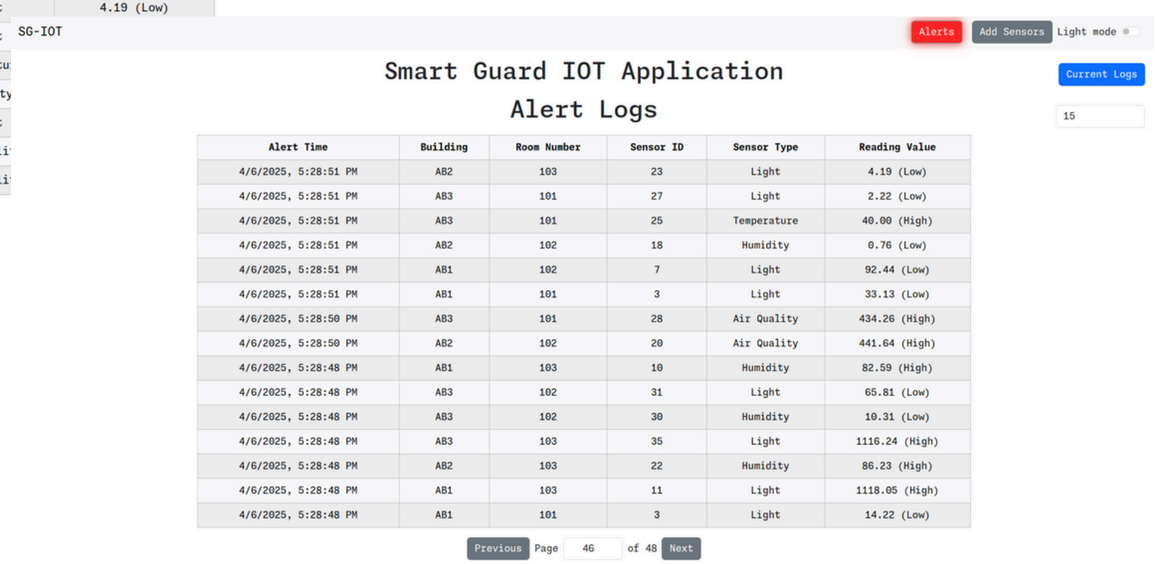
Past Data Visualization:



Objectives Achieved

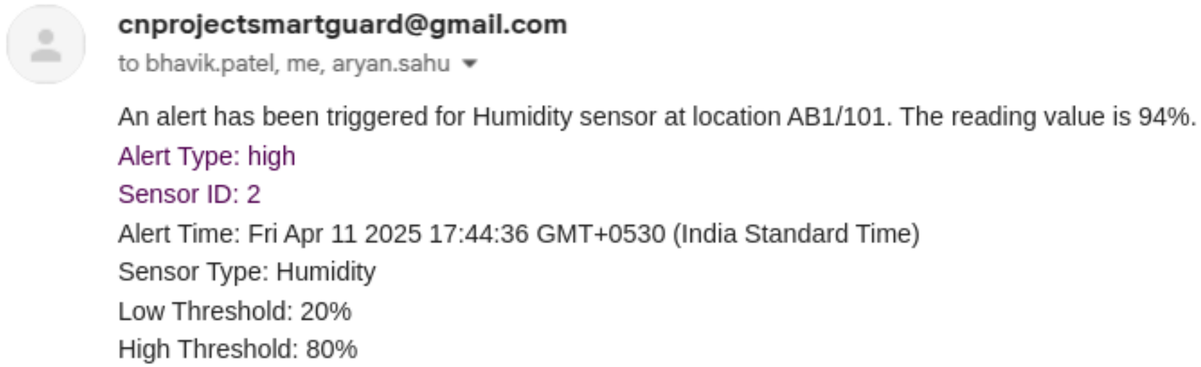
Alert Log Viewer:





Objectives Achieved

Email Alert System:



cnprojectsmartguard@gmail.com
to bhavik.patel, me, aryan.sahu

An alert has been triggered for Humidity sensor at location AB1/101. The reading value is 94%.
Alert Type: high
Sensor ID: 2
Alert Time: Fri Apr 11 2025 17:44:36 GMT+0530 (India Standard Time)
Sensor Type: Humidity
Low Threshold: 20%
High Threshold: 80%

Let’s Demonstrate

Thank you