ESBHA001

System Design Specification

Soft

Issue/Version: 0.3

**Document Control Data Sheet**

|  |  |
| --- | --- |
| **Document information** | |
| **Classification** | Confidential |
| **Document Name** | ESBHA001 System Design Specification |
| **Document Issue No.** | 0 |
| **Document Revision No.** | 3 |
| **Document Id** | ATPL/SDS/01/17/02/2020-01 |
| **Document Issue date** | 17-Feb-2020 |
| **Prepared by** | Veena G |
| **Reviewed By** |  |
| **Approved By** |  |
| **Distribution List** | Alcodex project team |

Contents

[1.0 Introduction 4](#_Toc33084989)

[1.1 Purpose and Scope of Document 4](#_Toc33084990)

[1.2 Project Overview and Scope 4](#_Toc33084991)

[1.3 Assumptions and Constrains 4](#_Toc33084992)

[1.4 Standards and Guidelines 4](#_Toc33084993)

[1.5 Design Approach 5](#_Toc33084994)

[2.0 System Approach 5](#_Toc33084995)

[2.1 AQI Calculation 6](#_Toc33084996)

[3.0 Modular Approach 8](#_Toc33084997)

[3.1 Module Decomposition 8](#_Toc33084998)

[3.2 Data Flow Diagrams 11](#_Toc33084999)

[4.0 Integration Descriptions 11](#_Toc33085000)

[4.1 APIs 11](#_Toc33085001)

[4.2 Ratelimiting 12](#_Toc33085002)

[5.0 Interface Descriptions 12](#_Toc33085003)

[5.1 User Interfaces 12](#_Toc33085004)

[6.0 Database Design 16](#_Toc33085005)

[6.1 Database description 16](#_Toc33085006)

[6.2 Database Recovery 17](#_Toc33085007)

[7.0 Requirement Amplifications 17](#_Toc33085008)

[8.0 Appendix 17](#_Toc33085009)

[8.1 Appendix A: Acronyms 17](#_Toc33085010)

# Introduction

## Purpose and Scope of Document

This document provides the design specification of the environmental sensor server for Bhavnagar Municipal Corporation. The scope of this document includes the detailed design specifications of the server.

## Project Overview and Scope

ESBHA001 environmental sensor server provides a web application server and a database server. It processes and stores data provided by environmental sensors of Envitus IoT Suite by Alcodex. The server includes different features such as live data monitoring, statistical data calculation, alarm management, user management, and error handling. It also has APIs to integrate with third parties, for getting data in digital signage.

**1.2.1 Scope**

In this project, we assure the design, development, testing, supply, installation and deployment of the Envitus sensor server and its dependencies such as NodeJS and MongoDB.

## Assumptions and Constrains

#### Assumptions

* Environmental Sensor Server shall be available 24\*7 online.
* The Internet shall be always available.
* The client shall provide hosting space.
* Data Center environment shall be secure.
* Configuration of servers shall be based on server sizing document.
* A self-signed certificate for HTTPS protocol shall be used in the server.

#### Constraints

* Non-availability of the network.
* Installation of the server will not be easy if the configuration of servers is not based on server sizing document.

## Standards and Guidelines

**1.4.1 Central Pollution Control Board**

* Guidelines to calculate the Air Quality Index (AQI).
* Colour coding of AQI and other parameters.

## Design Approach

Agile methodology is used as the design approach for developing an Environmental Sensor Server. The language used for development is JavaScript. Angular JS framework is used to develop front-end and for backend development, node.js is used. MongoDB is used as the database management platform.

# System Approach

The environmental sensor server consists of different modules such as live data module, alarm management module, API key generator module, user management module, device management module, error diagnostic module, and API interface module.

The environmental sensor server waits for the incoming data from the device.

The data should be in a predefined JSON format. The valid JSON structure is:

*{*

*“device\_Id”:” abc”,*

*“data”:*

*{*

*“temperature”: 30,*

*“pressure”: 90,*

*“humidity”: 100,*

*“PM10”: 30,*

*“PM2p5”: 20,*

*“PM1”:10,*

*“CO”:0.1,*

*“CO2”:69,*

*“NO2”:0.02,*

*“SO2”:0.05,*

*“O3”:0.01,*

*“noise”:80,*

*“rain”:1,*

*“time”:”12-02-2020 10:50:30”*

*“er\_read\_sensor”:10,*

*“er\_init\_sensor”:40,*

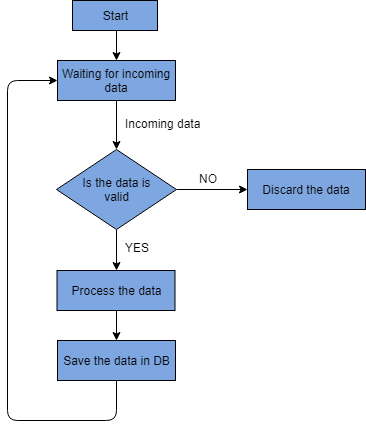
*“build\_ver”:”10.89.89”,*

*“sig\_strength”:”89,23”*

*}*

*}*

If the incoming data is valid then it is processed and calculate the Air Quality Index of that data set. The processed data is stored in the database.



## AQI Calculation

AQI is calculated based on the CPCB guidelines. The parameters affecting AQI calculation are PM10, PM2.5, CO, NO2, SO2, O3 and NH3.

The calculation is based on the following guidelines.

* The web-based system is designed to provide AQI on a real-time basis. It is an automated system that captures data from continuous monitoring stations without human intervention and displays AQI based on running average values
* Overall AQI is calculated only if data are available for minimum three pollutants out of which one should necessarily be either PM 2.5 or PM 10 . Else, data are considered insufficient for calculating AQI.
* Based on the measured ambient concentrations of a pollutant, sub-index is calculated, which is a linear function of concentration. The worst sub-index determines the overall AQI.

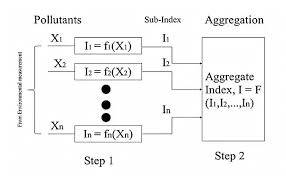
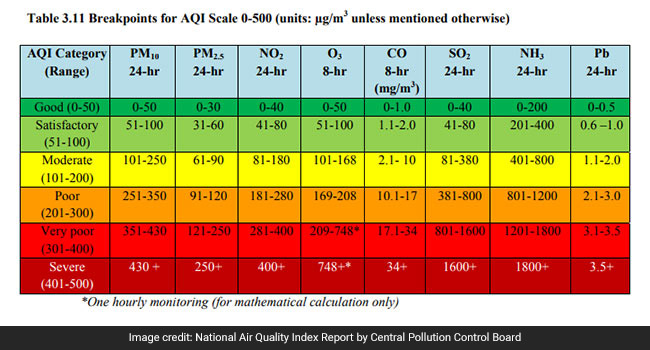


Figure 2.1(a): AQI Calculation

* CO in mg/m3 and other pollutants in μg/m 3.

Figure 2.1(b): AQI categories

# Modular Approach

## Module Decomposition

#### Livedata Module

There should be 3 tables in this module, the first table is to list out all the devices added in the server, the second one is for displaying livedata and the last one is for displaying statistical data. When clicking on a device in the device status table the livedata table and data statistics table should be updated with the latest data of the selected device by fetching the data from its appropriate collections in the database. The livedata table should be designed as a dynamic table. The header of the table should be depending on the parameters defined in the device specification module. There should be both vertical and horizontal scrolling available. The device indicator should be green until 15 minutes after the data reached. If the device is offline then the indicator should be in yellow.

The data statistics table should have headers such as parameter, unit, samples count, min, max, average and date. The calculations of statistical data is done in *statisticsManager* module. The processed data is stored on different collections like *device\_L\_stat\_daily, device\_L\_stat\_monthly and device\_L\_stat\_yearly* to display daily, monthly and yearly statistics respectively.

#### Device Management Module

The device management module is used to add, edit or delete devices. The user can add a new device by providing its details to the device management page. The details are stored in a collection ‘*devices’* in the database. Each device should be stored as different documents in the collection. When editing or deleting the device, the document should be updated or deleted respectively with the new data.

#### Alarm Management Module

The alarm management module is used to generate alarms based on the rules added in the server. Added alarm rules are stored in the collection *alarm\_rules* in the database. Each data of the device is compared with the threshold value set by the authorities which is stored in the collection *alarm\_rule* in the database. If the device value exceeds the value of the threshold then an alarm is generated and stored in the *alarm\_records* collection in the database.

#### User Management Module

User management module is used to add or delete users. New user details can be added to the server by using the user management page. The added informations are stored into the database in a collection *‘users’.*

There are three types of users:

1. Administrator(A)

Admin will be able to access the data analysis, user management, device management, and alarm management modules.

1. Supervisor(S)

The supervisor can view the data analysis module and the alarm management module.

1. Operator(O)

Operator has only permission to view the data.

There is a special user with extra privileges.

1. Super Admin

This is a special user. Super admin has access to every module.

A close up of a map

Description automatically generated

#### API Key Generator Module

This module is used to generate API keys that are used with APIs for fetching data from the server. API keys provide additional security features to the data.

#### Error Diagnostics Module

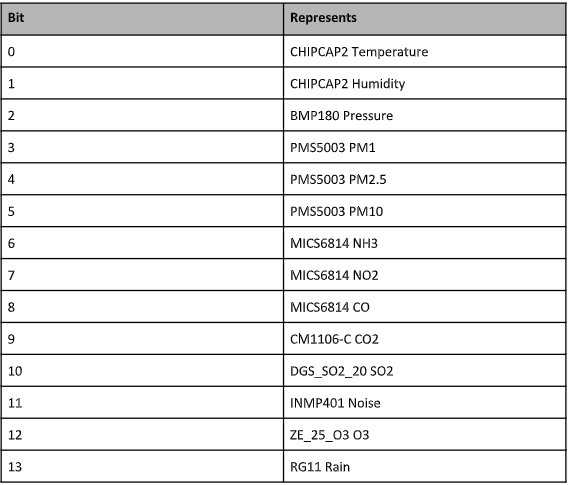
Error diagnostics module is used to display the error with the sensors of the environmental sensor. There will be two types of error,

1. Sensor Initialization error

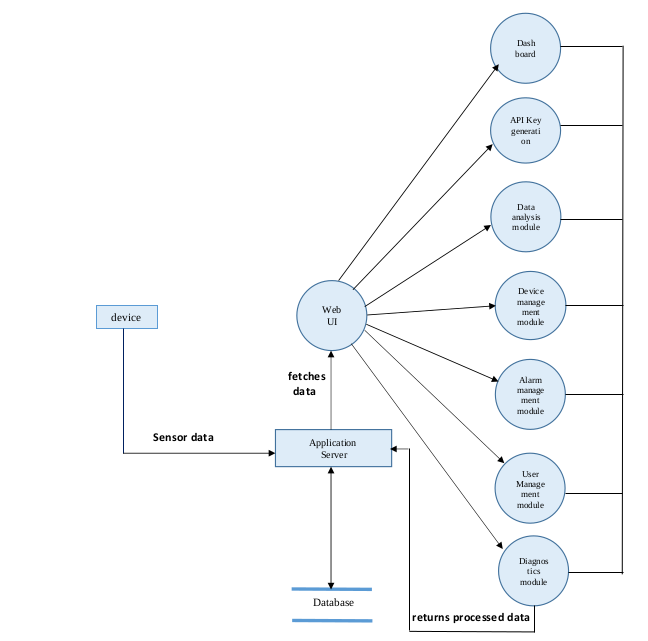
2. Sensor Read error

There should be 2 tables to display these 2 errors.

The data from the device is in a 32-bit format.



## Data Flow Diagrams



# Integration Descriptions

## APIs

Different APIs are used to integrate application server with third parties.

All the APIs are based on the HTTPS protocol. Response from the APIs will be in JSON format.

Different APIs used are:

* **device/sensor/stats**

The API will return daily device statistics sorted in the latest to oldest order.

General example:

<https://ipaddress:portnumber/device/sensor/stats?userId=username&amp;authPassword=password&amp;deviceIds=name-ofthedeviceID&amp;timeFrame=daily&amp;limit=1>

* **device/sensor/stats/count**

This API will return the number of records of statistics data for a device.

General example:

<https://ipaddress:portnumber/device/sensor/stats/count?userId=username&authPassword=password&deviceIds=name> of the device id&timeFrame=daily

* **device/sensor/livedata/v1**

This API will return the live data for a specific device.

General Example:

<https://ipaddress:portnumber/device/sensor/livedata/v1?userId=username&authPassword=password&deviceIds=name> of the deviceID&limit=1&offset=0

* **device/sensor/livedata/v1/count**

This API will return the number of data samples recorded for a specific device.

General Example:

<https://ipaddress:4001/device/sensor/livedata/v1/count>[? userId=username&aut](http://ipaddress:4001/device/sensor/livedata/v1/count?userId=admin&authPassword=admin&deviceIds=name) hPassword=password&deviceIds=name of the device ID

## Ratelimiting

API rate limiting is used to limit the maximum number of calls to an API. The most basic rate limiting strategy is that "clients can only send X requests per second". Many APIs implement rate limiting to ensure relative stability when unexpected things happen. If for some reason one client causes a spike in traffic, the API must continue running smoothly for other users instead of crashing. Performance isn’t the only reason to limit API requests, either. Rate limiting is done by using API keys. Authorized users can generate the API key by providing the limit in the application server. The generated API key is valid until the limit is reached.

# Interface Descriptions

## User Interfaces

Below given the screenshots of web pages of Server Application.

1. Home

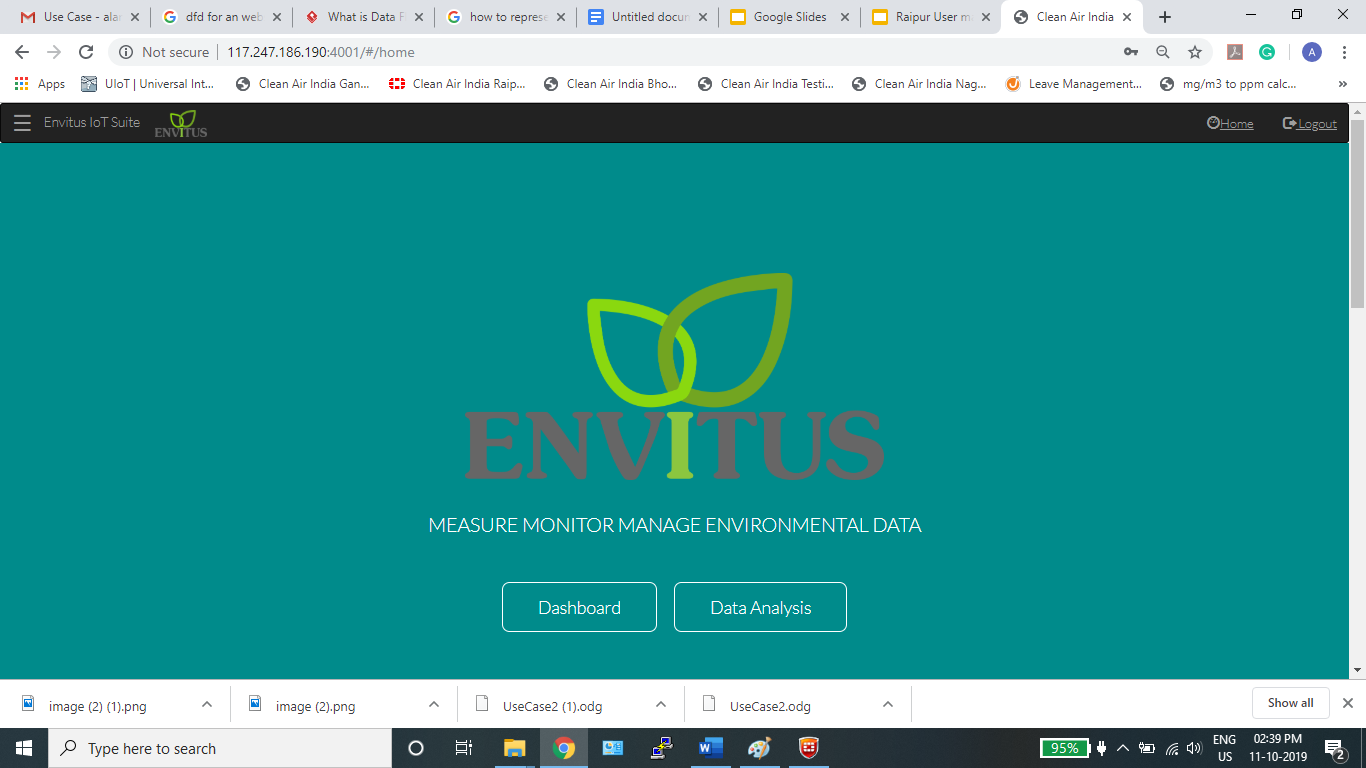


Figure 1: Home

1. Data Analysis

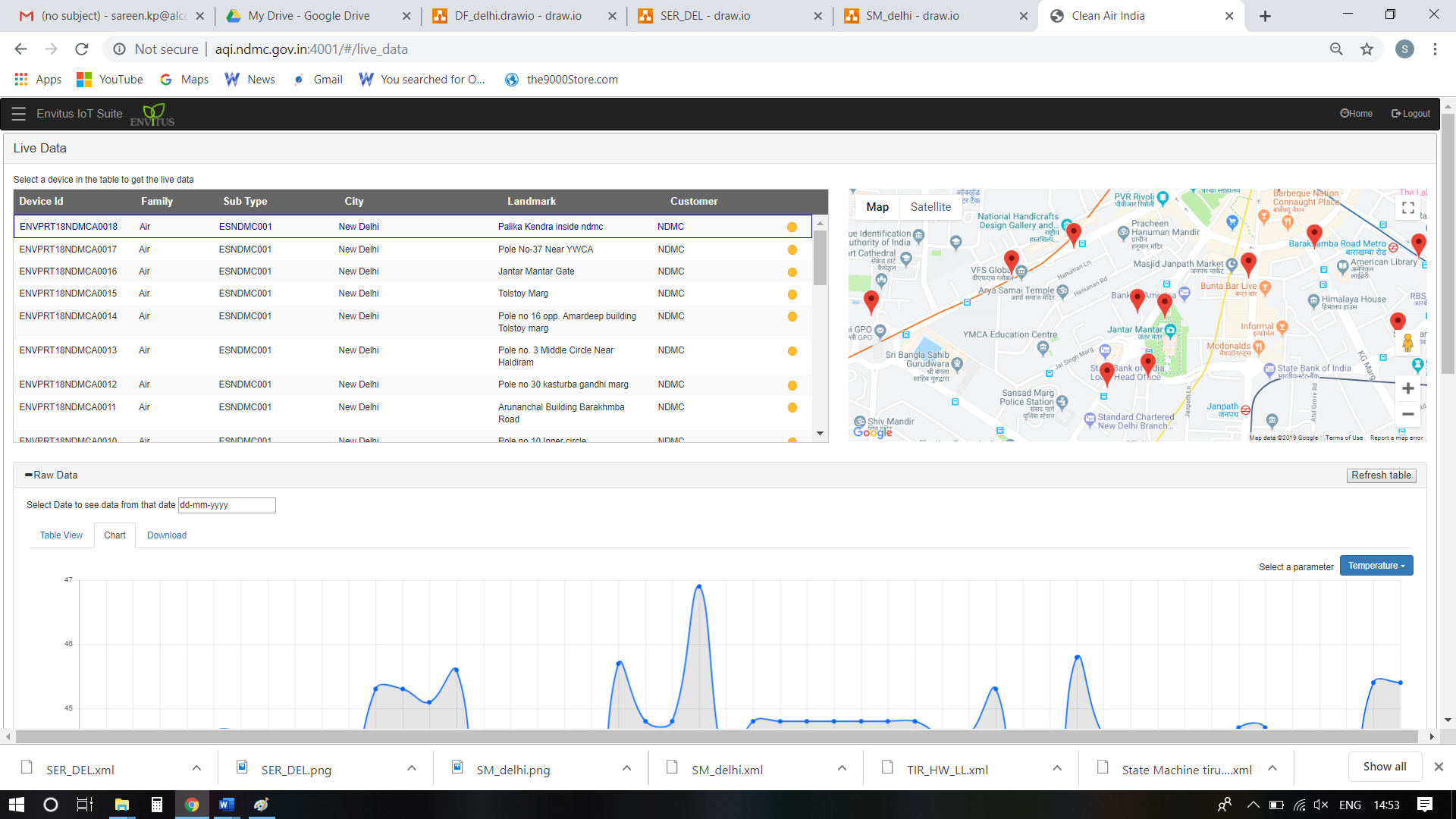


Figure 2: Data Analysis

1. Data Statistics

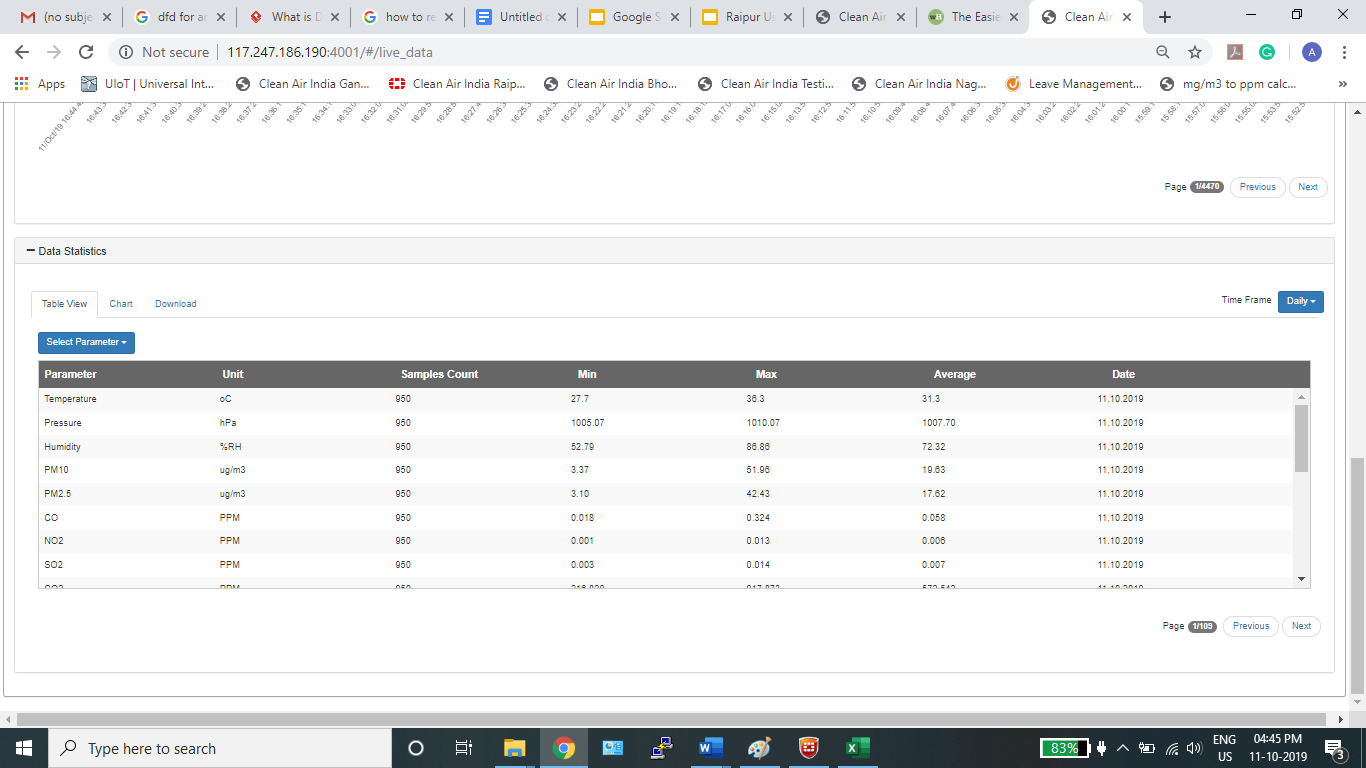


Figure 3: Data Statistics

1. Device Management

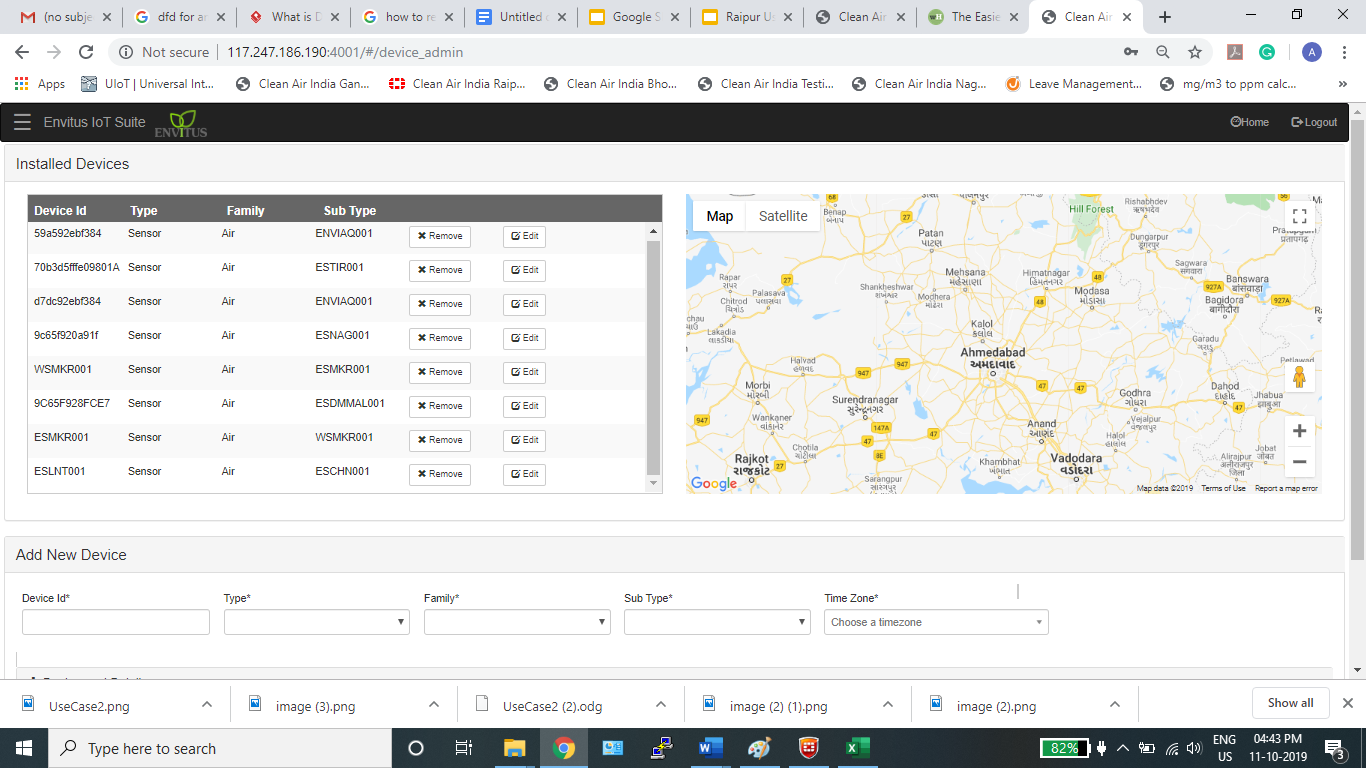


Figure 4: Device Management

1. User Management

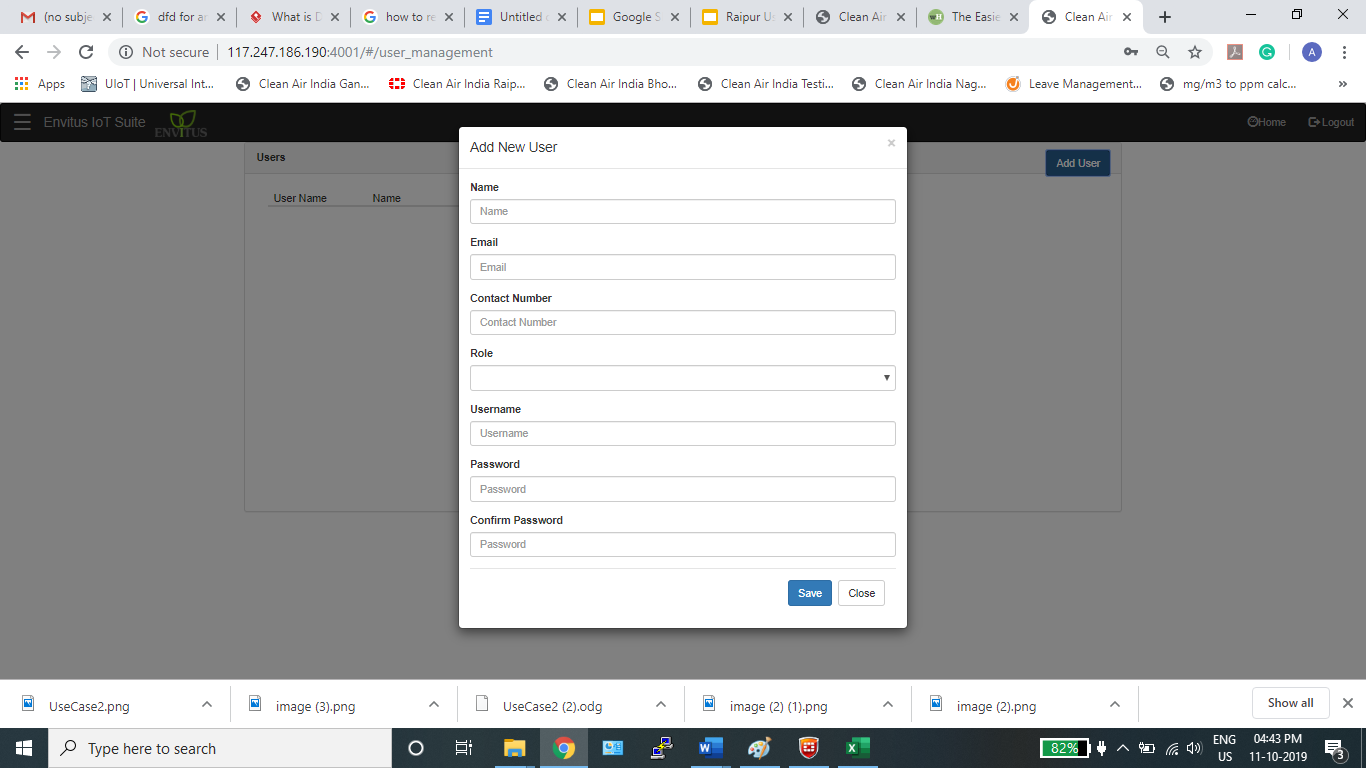


Figure 1: User Management

1. Alarm Management

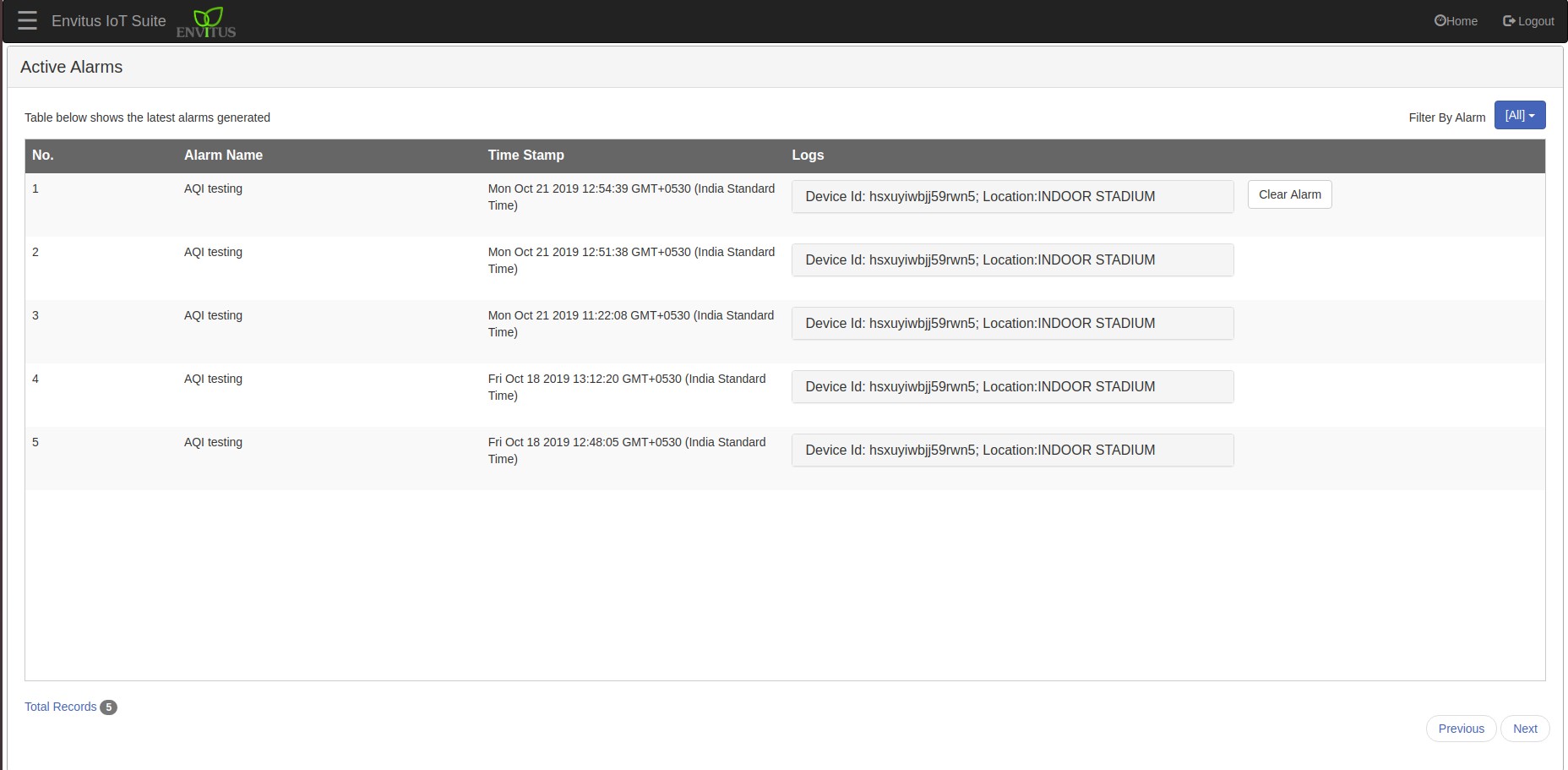


Figure 6: Alarm Management

1. Dashboard

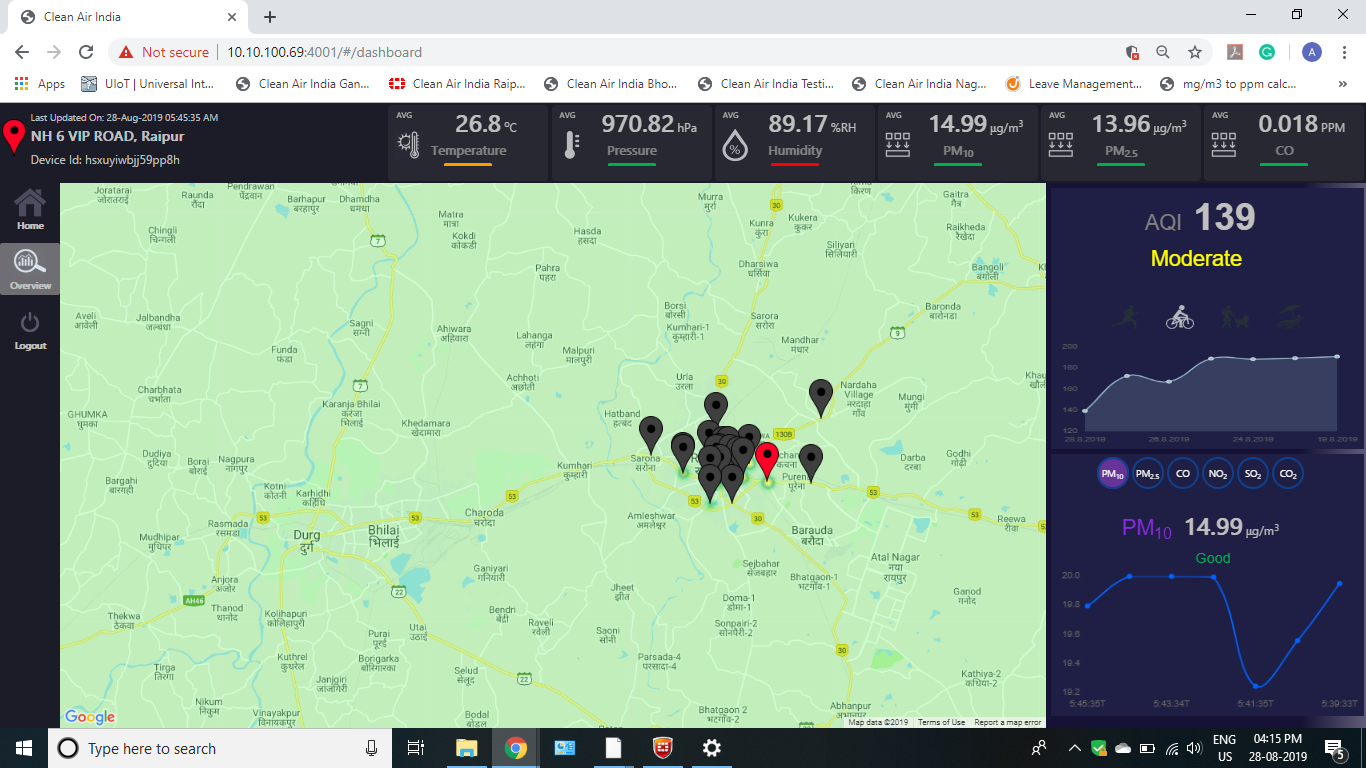


Figure 7: Dashboard

# Database Design

The main repository for the Application Server is a NoSQL database utilizing MongoDB as the management system and hosted on a Linux machine. This repository is where all sensor data is stored, processed and maintained. The application server uses this database for data processing. The server stores the upcoming sensor values directly into the database by using some queries. The data can be fetched and modified by using the web interface. The data can also be deleted by the authorities by using the website. MongoDB is a general-purpose, document-based database. All data is stored as documents. There is no structure or schema for storing the data. MongoDB documents are like JSON objects. The value of fields may include other documents, arrays, and an array of documents.

## Database description

The database used for the ESBHA001 environmental sensor server is ESBHADB. There are generally 10 types of collections in this database. They are:

1. device1\_L
2. device1\_L\_raw
3. device1\_L\_stat\_daily
4. device1\_L\_stat\_monthly
5. device1\_L\_stat\_yearly
6. ThirdPartyUsers
7. alarm\_records
8. alarm\_rules
9. devices
10. users

## Database Recovery

The data Recovery system is used as a backup of the ESBHA001 application server.

The working of DR is as follows:

1. Environmental Sensor Devices are configured to send data to both DC and DR server.

2. Both servers will be in Active mode.

3. DR server is an exact replica of the DC server. So, all the configurations and settings of the DR

server are the same as that of the DC server.

4. If Both servers are in Active mode the ES devices will send data to both servers.

# Requirement Amplifications

<List the features, functions, and interfaces etc that have resulted in amplifying the requirements or have caused changes to the requirements. Ensure the changes are reflected in the requirements document as well. >

# Appendix

## Appendix A: Acronyms

| **Term/ Acronym** | **Definition** |
| --- | --- |
| API | Application Program Interface |
| AQI | Air Quality Index |
| AQMS | Air Quality Monitoring Station |
| DC | Data Center |
| DR | Data Recovery |
| IoT | Internet of Things |
| JSON | JavaScript Object Notation |
| SQL | Structured Query Language |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |