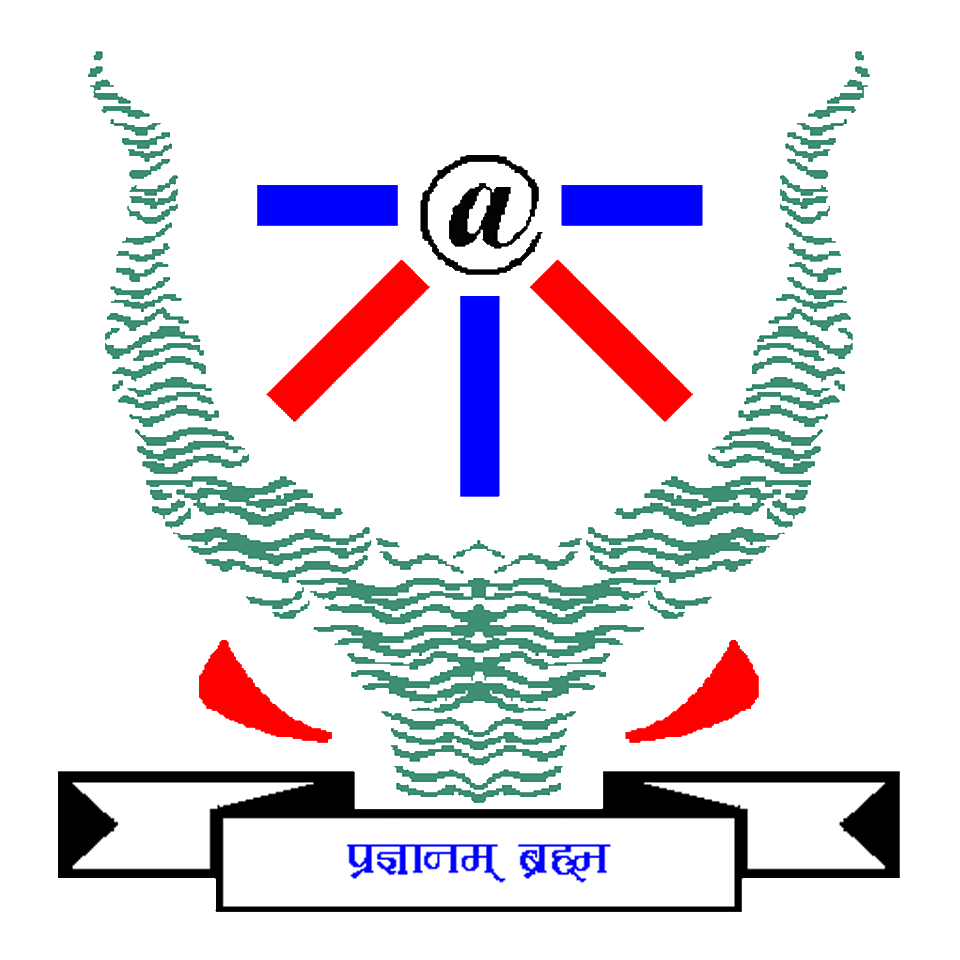
**OOM MINI-PROJECT**

**ATMOSPHERIC MANAGEMENT SYSTEM**

****

Group Number: **7**

Group Members:

**Aryan Gupta IIT2019101**

**Shreyas Gupta IIT2019102**

**Sarthak Maheshwari IIT2019117**

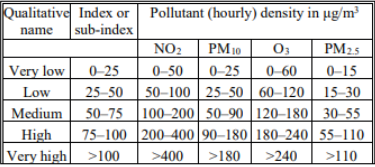
**Ankit Gupta IIT2019138**

Submitted To: **Dr. Sonali Agarwal**

---------

**Problem Statement**

To develop a GUI enabled air quality management system for IIITA campus. The system should be able to monitor the level of hazardous gases present in the environment (via means of random number generation). In an alarming situation the system must be able to generate alarm (also add auto generated message and mail system). Meanwhile, the system must also be capable of measuring the humidity and temperature present in the atmosphere. The code must also maintain the log record of previous days and a generate report button should be added to generate the report with the input given as range of timestamp or the whole record. Here is rough standard air quality measure:



Air Quality Measures

**Technologies Used:**

Java, Java Swing/AWT

**Required Functionality:**

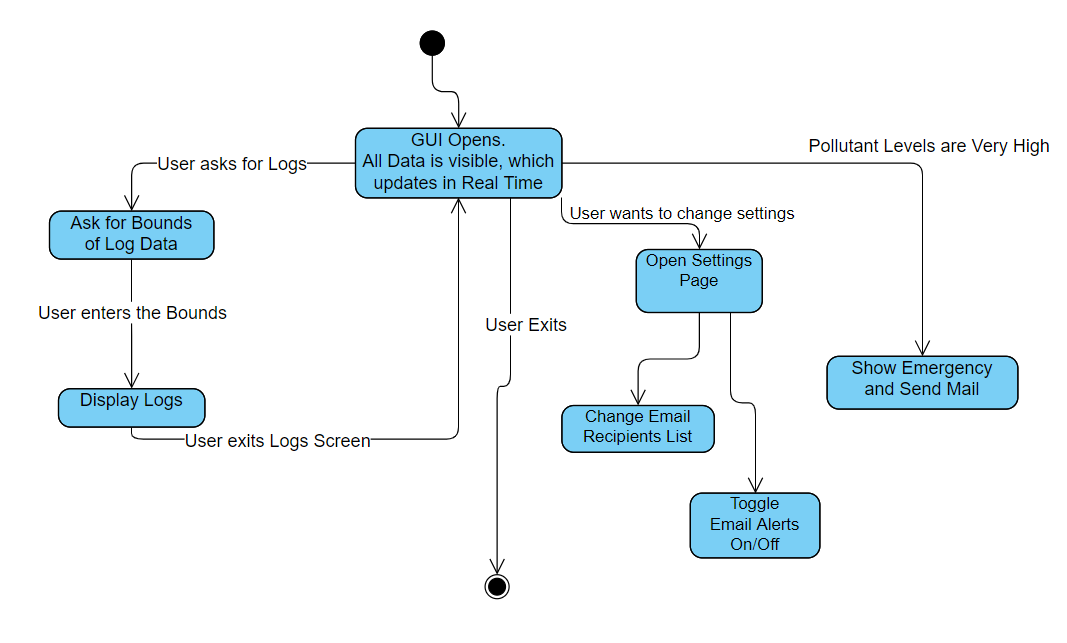
1. Main Screen: All air quality measures, pollutant levels, temperature and humidity is displayed and updated in real time.
2. Logs Screen: Generate log records of pollutant levels of previous days
3. Automatic Alerts: Email Alert when pollutant level is high.

**Improvised Functionality Added:**

1. Settings Screen: Change the mailing list for email alerts and also ability to toggle on/off the emails.

**State Diagram**

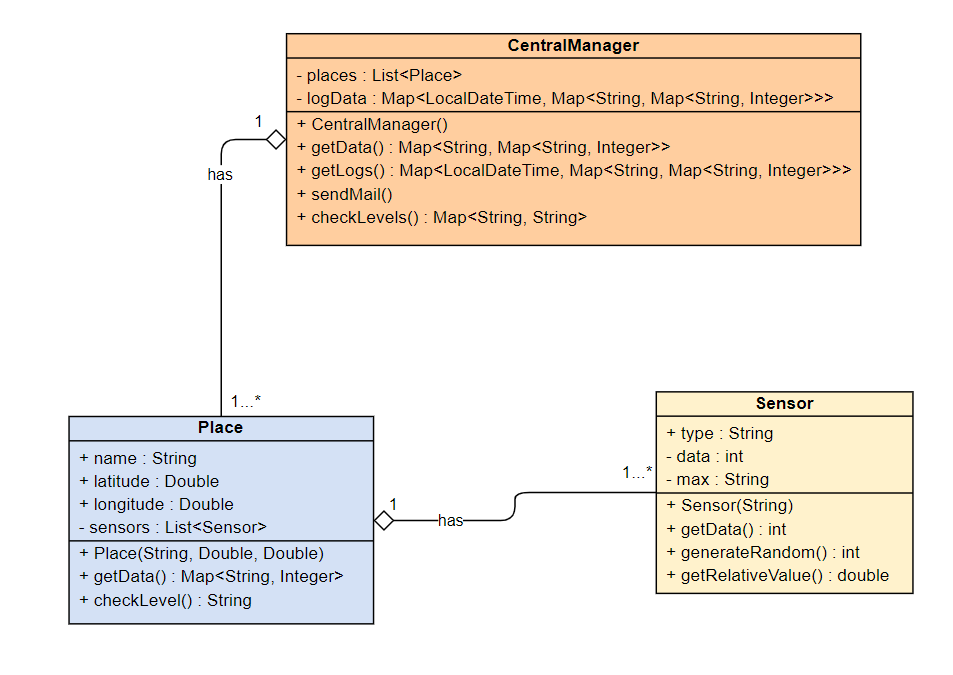
A State Diagram is a **behavioral** diagram which represents the behavior of a system or a part of a system using finite state transitions. The first state shows that the app opens and shows the levels of pollutants updating in real time. Now if the pollutants levels in the atmosphere are high, then the app shows an emergency and sends an Email to the admin. The user can also ask for logs details, where he is asked for bounds of the Log data required and once the user enters the Bounds then the app will display logs. The user can also open the settings page and change email recipients for emergency mails and can also toggle the emails off entirely.



State Diagram

**Class Diagram**

The class diagram has appropriate structure to represent the classes, inheritance, relationships, and everything that OOPs have in its context. So here we have a class named CentralManager with attributes places and logData to store a list of places and log data. It has a default constructor for initializing the Places, checkLevels() to check the pollutants level in air and getData() and getLogs() that return data and logs , sendMail() to send a mail to admin on an emergency. CentralManager can manage one or more Place at a time. A Place class will have name, latitude, longitude and a list of sensors as its attributes. It has a constructor for initializing the Sensors. It also has the following functions: checkLevel() to check if pollutant level at the place is high or not,

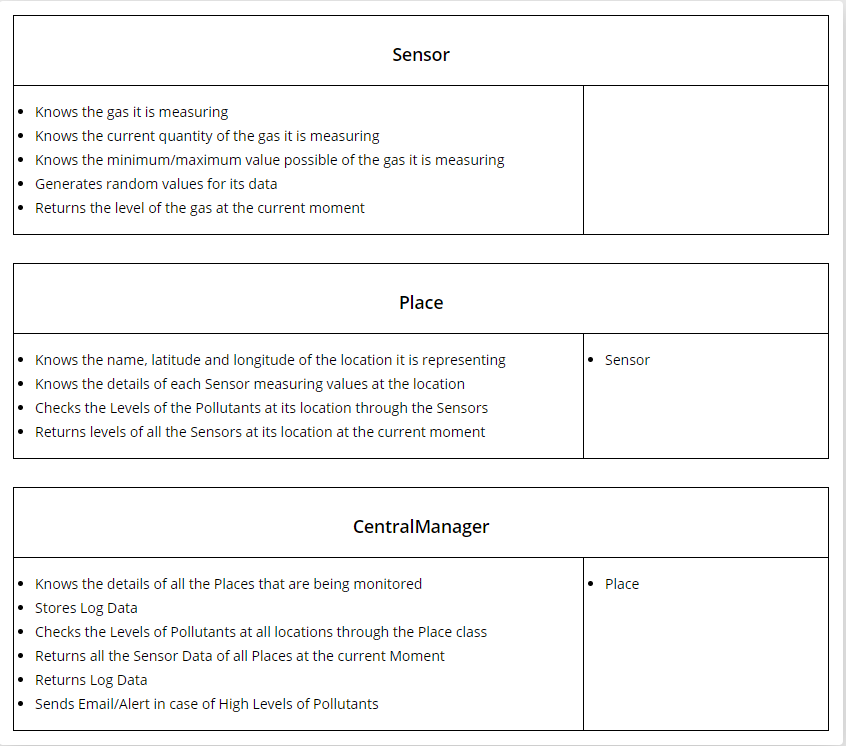


Class Diagram

getData() to get pollutant levels at the place. Again a place can have one or more sensors to check different pollutant levels. The Sensor class has its type(what type of pollutant it will measure), data, min and max as its attributes. The constructor initializes the sensor type and min-max values, generateRandom() which will generate a random value of pollutant level and a getData() function that will return the data collected by the particular sensor.

**CRC Diagram**

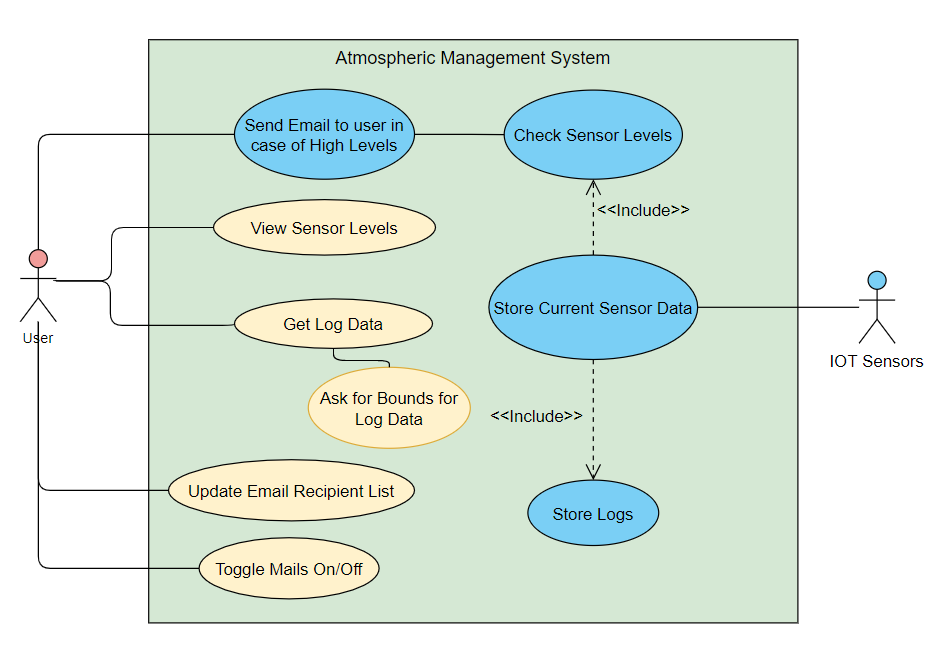
CRC Diagram consists of some standard cards which represent the responsibilities and collaborators of each class. So, the first card shows the Sensor Class and all its responsibilities are listed in the left column of the card and the right column is empty which means the Sensor Class doesn’t require any other to fulfill its mentioned responsibilities. Then the next card shows the Place Class with its responsibilities mentioned in the left column and it needs to collaborate with the Sensor Class to get all of its responsibilities done and the last card shows the CentralManager Class with its responsibilities mentioned in the left column and CentralManager needs to collaborate with the Place class as it manages and stores details of all the Place instances.



CRC Diagram

**Use Case Diagram**

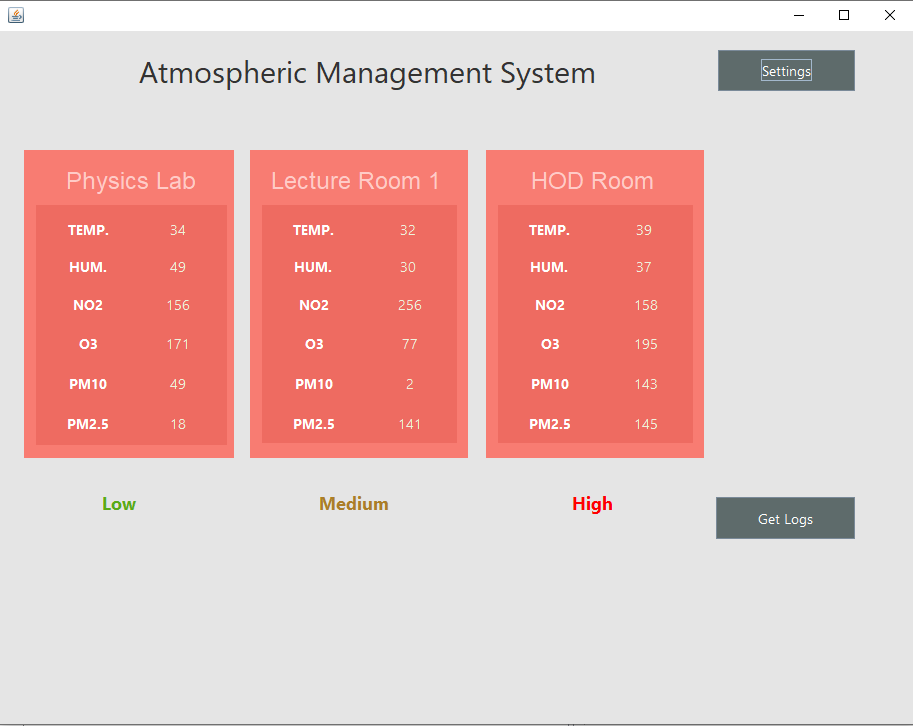
Use Case Diagram is used to represent the dynamic behavior of a system. It encapsulates the system's functionality by incorporating use cases, actors, and their relationships. The user can view the sensor levels, get log data, update email recipients list, or toggle mails on/off. In the backend, the IOT Sensors(random number generation here) will generate the data, and the included functionality is that it checks the sensors levels and stores them as logs, it will also send an email alert when the pollutant levels are high.



Use Case Diagram

**App Walkthrough**

This is the homescreen of our system. On the homescreen we display the level of different gases present in three different places of CC3 (Physics Lab, Lecture Room 1, HOD Room). In each card we display the level of hazardous gases in the particular place and also the temperature and humidity.. These values that are generated randomly are updated every 4 seconds in each card and according to the level of gases.



Home Screen

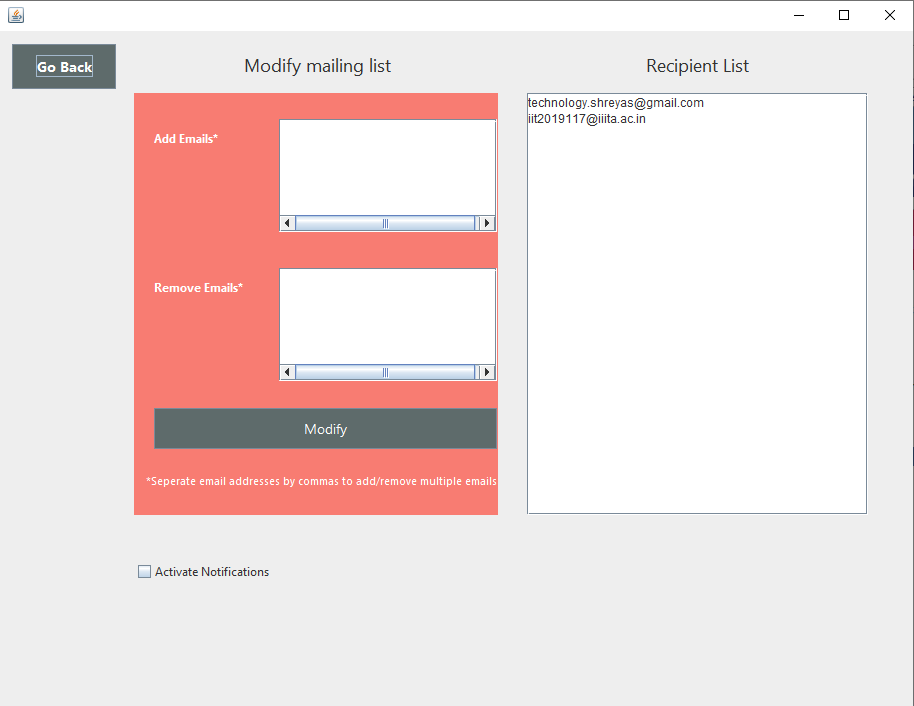
According to the relative levels of the pollutants, an AQI level is generated as Very Low, Low, Medium, High and Very High.

Accordingly, in the particular screenshot, it shows the pollutant level at the places as Low, Medium and High.

If the level is High or Very High an email is sent as an alert.

The home screen has buttons to get the logs, or to access the settings.

If we click the Settings button, the Settings screen is displayed.



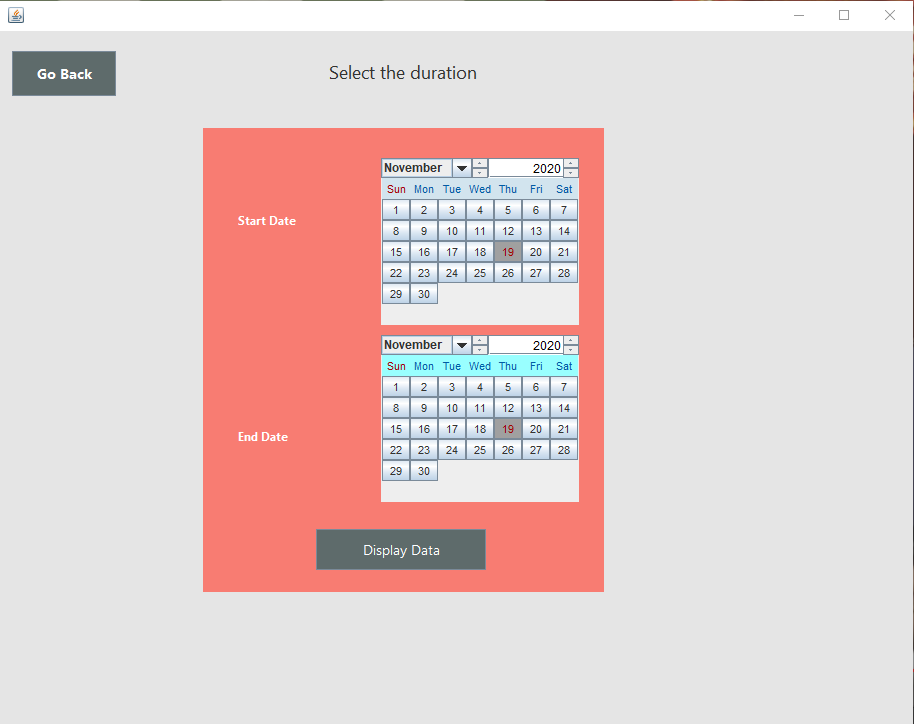
Settings Screen

In the Settings Screen, the mailing list is displayed and emails from the list can be added or removed. This mailing list is used to determine who will get the email alerts in case of high pollutant level.

The active notifications checkbox is present to toggle the alert mails on or off. If the box is unchecked the high alert mail would not be sent. This is added to prevent spamming of inbox with alert mails.

We can always go back to our home screen by pressing the Go Back button.

If we click on the Get Logs button from the homescreen, the Select Logs Duration Screen opens up.

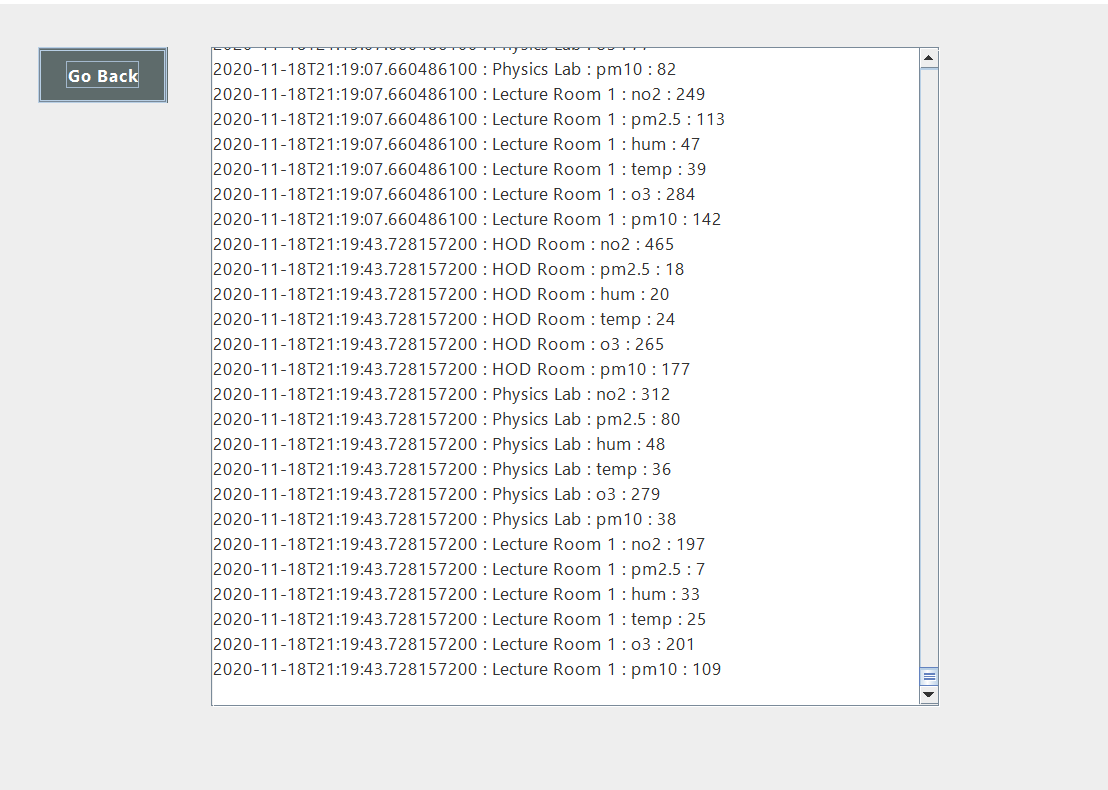


Select Logs Duration Screen

Here we have to decide for what duration we want our logs to be shown. We have to select the start and the end date and click on the display data option. Also just to note, while testing, we need to mention the dates of 18 as start, and 20 as end to get the logs for 19. So, Start and End Date are not included in the Log Data, only dates in between.

We can also always go back to the Home Screen by clicking the Go Back button.

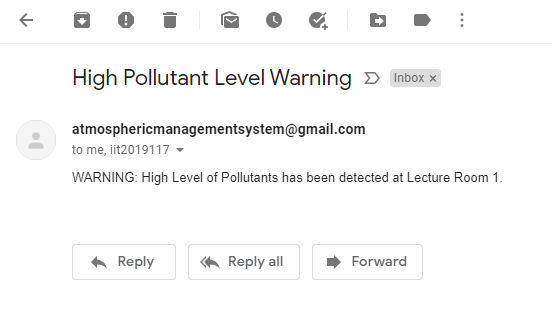
When we click Display Data after selecting the bounds, the Logs screen is displayed.



Logs Screen

The logs will give the exact details, so what was the level of each pollutant at a place with an exact timestamp.

Also, here is a sample mail that is sent in case of emergency:



Sample Mail