DASS Design Doc v1.0

Team 51

Abhijit, 2019113032
Gaurav Singh, 2020111014
Kavya, 2019101127
Pratyay Suvarnapathaki, 2020111016

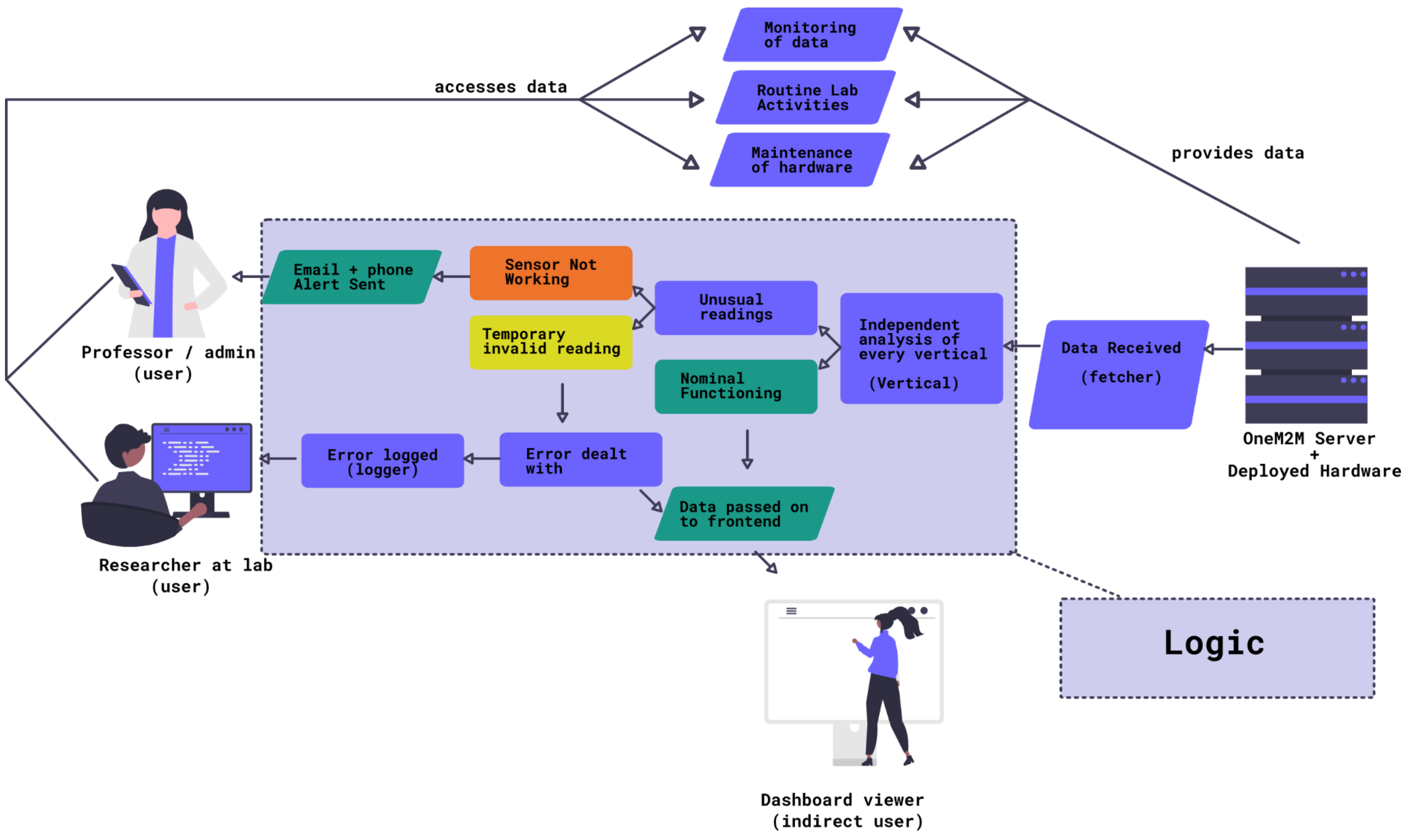
Overview

Project number	50
Project Title	Framework to access and Validate sensor data
Document	DASS Design Document v1.0
Creation date	15th February, 2021
Created By	Pratyay Suvarnapathaki, Gaurv Singh, Abhijit
Client	Prof. Anuradha Vattem, Smart City Living Lab, IIITH

Problem Statement

The Smart City Living Lab has recently deployed more than 100 data collection nodes across the campus. These nodes track various metrics for air quality, water quality, energy, weather, etc. Thus, a large amount of data is being generated.
The natural extension of this hardware framework is to build a robust software setup to ensure accuracy and consistency in sensor readings. Factors such as hardware malfunctions, network failures, or software glitches affect data quality and result in inaccurate data generation.
Therefore, our role is to develop a framework to integrate with the existing oneM2M-based setup, and implement features such as uniform data quality validation, outlier detection, and hardware malfunction indication.

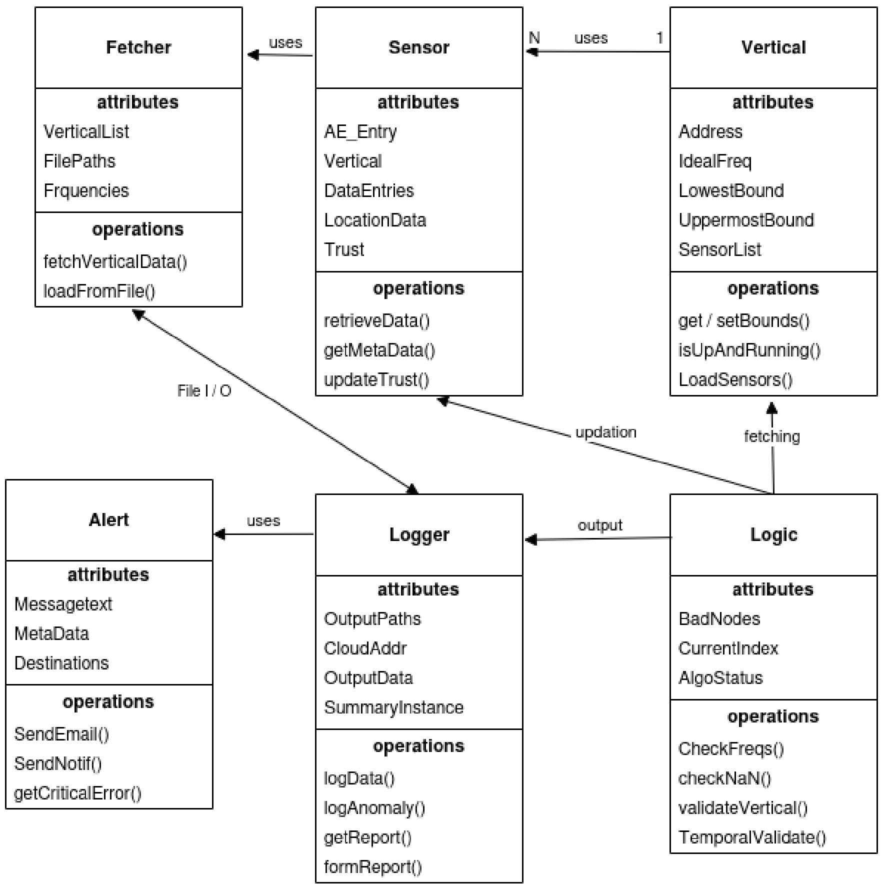
Architectural Design



User Interfaces

As the main end users of the framework are the researchers at the Smart City Living Lab, they require minimal abstraction and maximum control. They should be able to granularly edit the existing setup of nodes, and be able to conveniently monitor the (predicted) data collection status of each deployed node. The client has not specified the need for any UI. However, depending on further progress and ease of use, a simple GUI to view program status and a few recent detections/analytics might optionally be implemented.
There is no API as such, as this is a standalone framework that will live and be periodically executed on the lab's servers.

Model

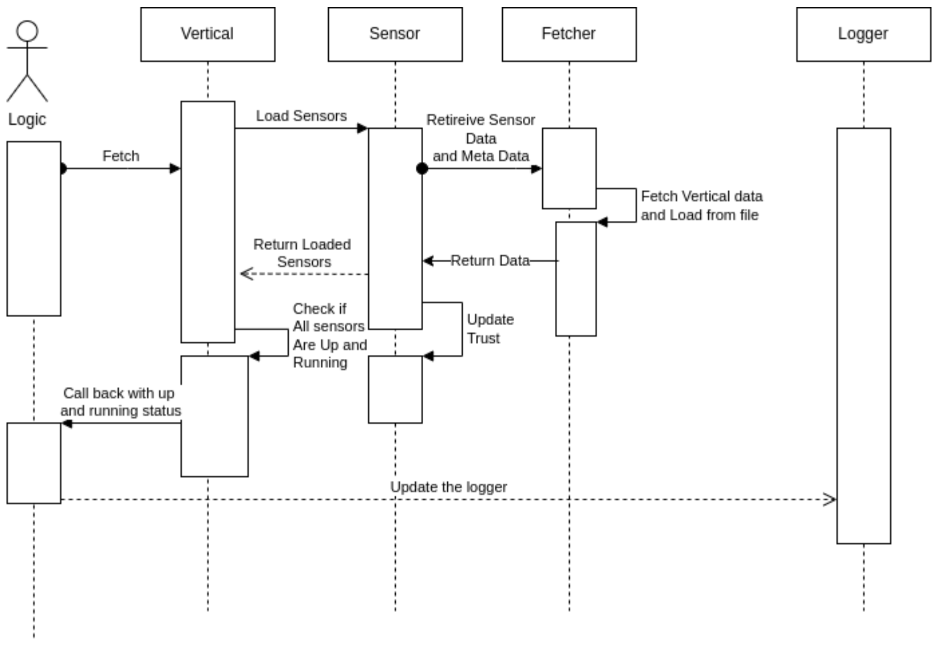


Fetcher	State: Information regarding OneM2M interfacing and the overall state of the current setup that persists between framework invocations. Behaviour: - Formulating HTTP requests - Fetching data from OneM2M server at specified intervals - Keeping track of time of data posting - Loading and updating program data from local files
Vertical	State: Stores relevant information regarding a particular OneM2M vertical Behaviour: - Stores and provides ideal posting frequency - Stores and provides sane data bounds - Updates other relevant vertical-related metadata - Calls the fetcher for the concerned vertical
Sensor	State: Data concerning a particular sensor Behaviour: - Updates the readings received from a particular sensor - Stores and provides other sensor-related metadata
Logic	State: Class that deals with the main framework logic, processing and analytics. Behaviour: - Performs the task of monitoring update frequency for each sensor of

	each vertical. - Checks for NaN values - Executes a data validation and error detection algorithm based on the values of the sensor readings. - Calls the logger and alert in case anomalies are detected.
Logger	State: Deals with the data that is to be written out / posted at the end of an invocation. Behaviour: - Writing out / presenting analytics generated by the logic to appropriate output files in a specific format. - Posting data to cloud files for UC-05 - Maintaining relevant files for summary health reports
Alert	State: Relevant data and procedures to send alert messages Behaviour: - Pushes email alerts using logged information to the specified users - Pushes phone based alerts

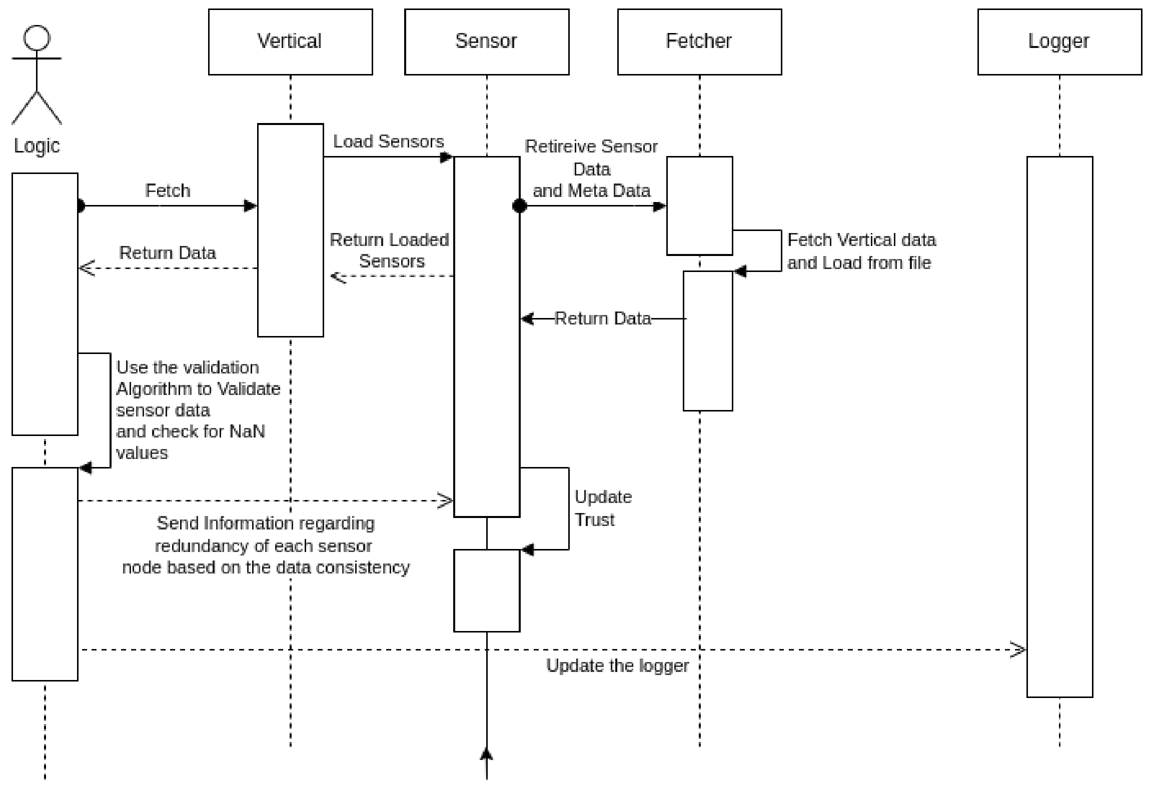
Sequence Diagrams

UC - 01: Up and Running

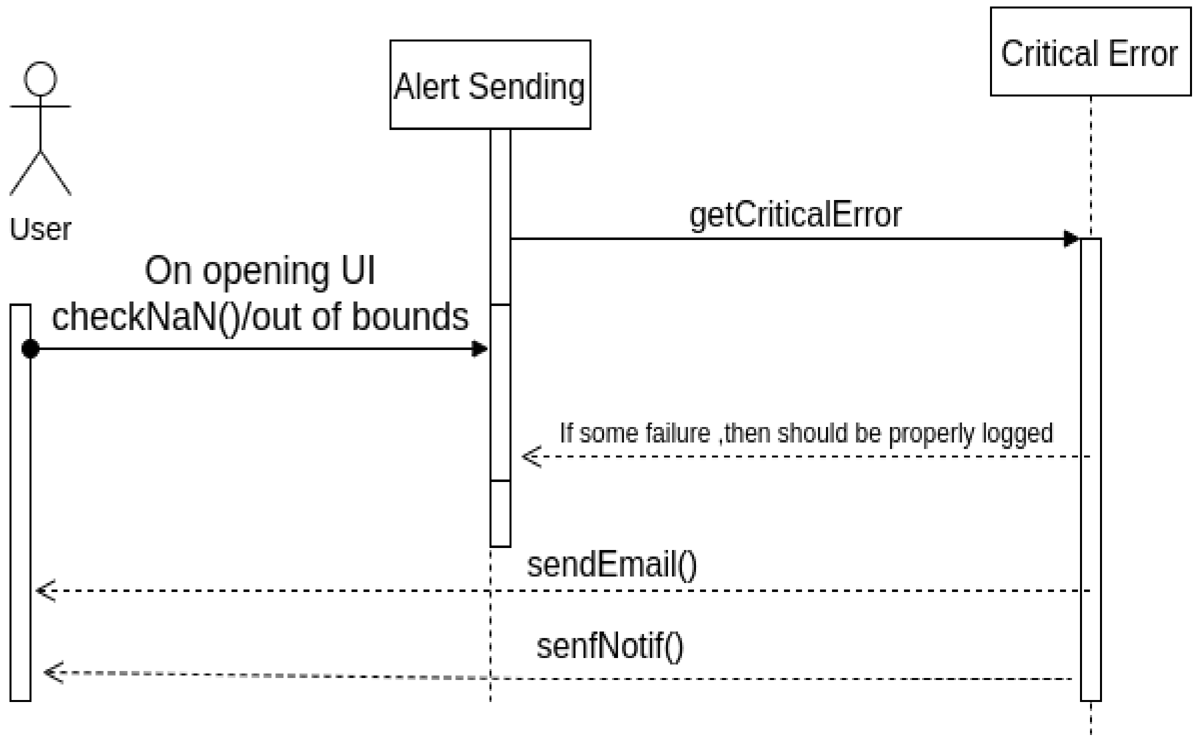


Vertical
Sensor
Fetcher
Logger
Logic
Load Sensors
Retireive Sensor Data and Meta Data
Fetch
Fetch Vertical data and Load from file
Return Loaded Sensors
Return Data
Check if All sensors Are Up and Running
Update Trust
Call back with up and running status
Update the logger

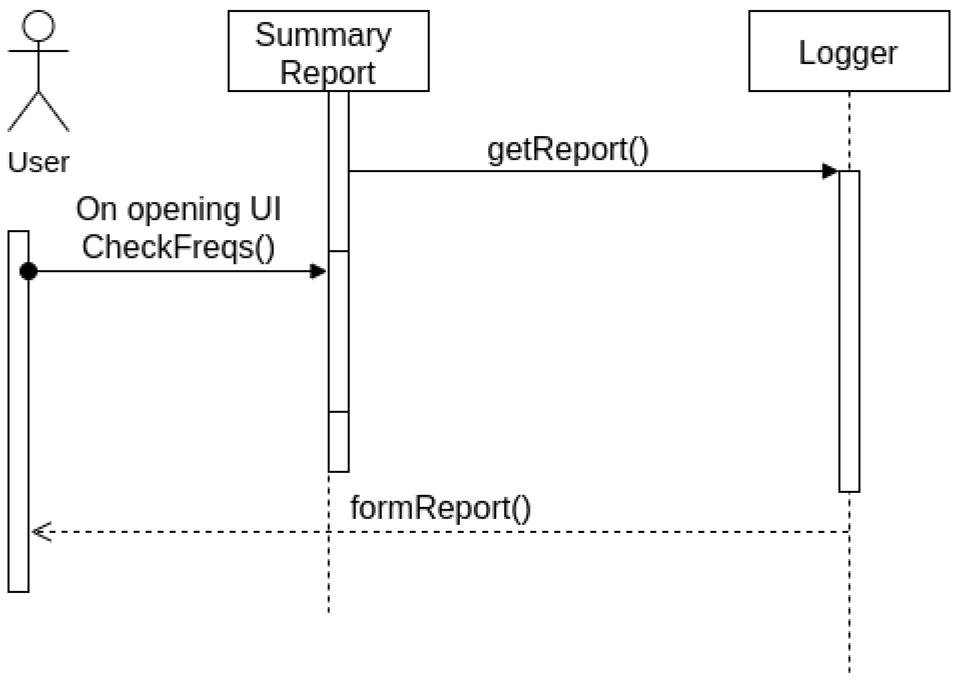
UC - 02 : Data Validation


Vertical
Sensor
Fetcher
Logger
Logic
Load Sensors
Retireive Sensor Data and Meta Data
Fetch
Return Data
Return Loaded Sensors
Fetch Vertical data and Load from file
Return Data
Use the validation Algorithm to Validate sensor data and check for NaN values
Update Trust
Send Information regarding redundancy of each sensor node based on the data consistency
Update the logger

UC - 03 : Alert Sending


Critical Error
Alert Sending
User
getCriticalError
On opening UI checkNaN()/out of bounds
If some failure ,then should be properly logged
sendEmail()
senfNotif()

UC- 04 : Summary Health Report



Design Rationale

V1.0 : The major challenge here is properly abstracting the different verticals in order to allow for the modularity the client requires. Hence, an object oriented approach has been followed, with each feature that is unique w.r.t. a vertical being associated with that very vertical, as a data member. Separation of the different parts of the overall logic has been achieved too. There might be a need to further divide the 'Logic' class depending on how the program code shapes up.

V2.0 : The program structure and requirements remain largely the same. There have been significant changes to the internal working of some classes due to the client recommending different approaches to achieve our intended functionality. So it may be necessary to change a few methods in the 'Fetcher' and 'Logger' classes in the future, but that is still somewhat uncertain.