**Software Requirements Specification**

**for**

Air Quality Management

**Version 1.0**

**Prepared by Cole Hoener & Lukas Schirren**

**INSA Lyon**

**30 November 2021**

**Table of Contents**

**Table of Contents ii**

**Revision History ii**

**1. Introduction 1**

1.1 Purpose 1

1.2 Document Conventions 1

1.3 Intended Audience and Reading Suggestions 1

1.4 Product Scope 1

1.5 References 1

**2. Overall Description 2**

2.1 Product Perspective 2

2.2 Product Functions 2

2.3 User Classes and Characteristics 2

2.4 Operating Environment 2

2.5 Design and Implementation Constraints 2

2.6 User Documentation 2

2.7 Assumptions and Dependencies 3

**3. External Interface Requirements 3**

3.1 User Interfaces 3

3.2 Hardware Interfaces 3

3.3 Software Interfaces 3

3.4 Communications Interfaces 3

**4. System Features 4**

4.1 System Feature 1 4

4.2 System Feature 2 (and so on) 4

**5. Other Nonfunctional Requirements 4**

5.1 Performance Requirements 4

5.2 Safety Requirements 5

5.3 Security Requirements 5

5.4 Software Quality Attributes 5

5.5 Business Rules 5

**6. Other Requirements 5**

**Appendix A: Glossary 5**

**Appendix B: Analysis Models 5**

**Appendix C: To Be Determined List 6**

**Revision History**

| **Name** | **Date** | **Reason For Changes** | **Version** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |

# **Introduction**

## **Purpose**

This document outlines the requirements and architecture for the Air Quality Management System proposed by IST INSA Lyon génie logiciel class. The entire scope of the project is discussed in this diagram, however not every diagram is included, but rather only overviews and high level abstractions.

## **Document Conventions**

No specific conventions should be followed or read as such. All information and each requirement stated has the same priority. No section takes a higher precedence than any other.

## **Intended Audience and Reading Suggestions**

This document is intended for any developers, testers, documentation writers, and the grader for this project. It is recommended to read the document in a linear fashion. All diagrams and explanations are inline with the document in their appropriate section.

## **Product Scope**

The Air Quality Management System is designed based on the following description, “This agency requires a software application to follow the sensor's state and to get aggregate information. Aggregate information will help decision makers to take decisions to enhance air quality. For example, on a given territory, get the mean air quality at a given time, or on a given timespan. It is also required to identify sensors with similar behaviors, i.e., having similar values for all the measures. Finding the values that characterize air quality at a given place is also a need.”

## **References**

*N/A*

# **Overall Description**

## **Product Perspective**

The Air Quality Management System is a new standalone product. This SRS covers the eternity of the system. This project is required for the INSA Lyon IST Génie logiciel course.

## **Product Functions**

* Aggregate sensor data
* Give air quality details of specific sensor
* Identify sensors with similar behaviors

## **User Classes and Characteristics**

Basic User - Every user will need the same basic knowledge. This will include just being able to read the text on screen and be able to point and click on sensors. No further knowledge will be needed. It will all be included within the application.

## **Operating Environment**

The software will be designed, tested, and maintained on macOS.

## **Design and Implementation Constraints**

There are no constraints on this software system.

## **User Documentation**

There will be no other documentation materials other than this document. All diagrams and design documents are inline.

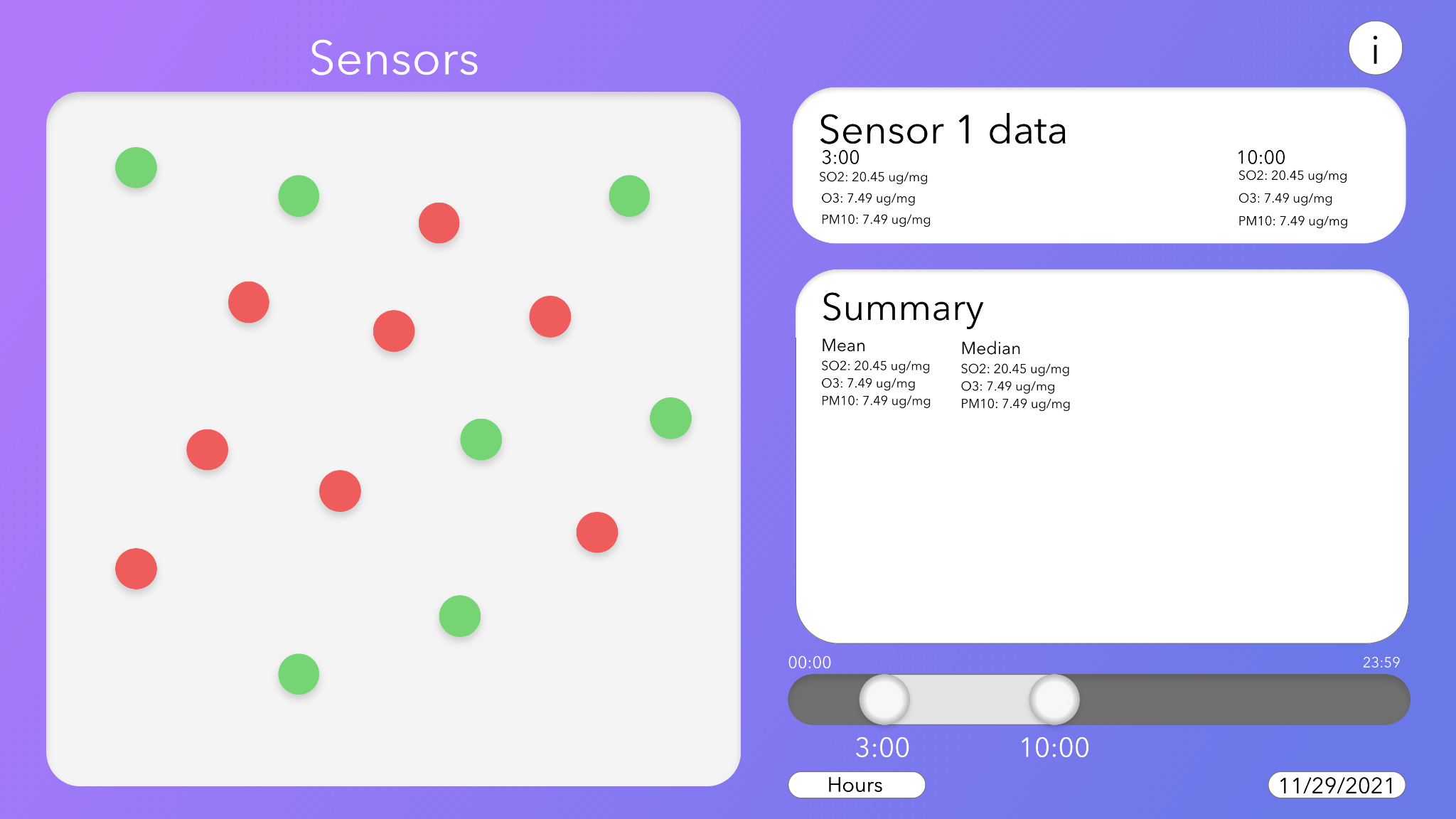
## **Assumptions and Dependencies**

There are no assumptions or dependencies. All technology in the scope of this project has been pre-vetted to work with the designed system.

# **External Interface Requirements**

## **User Interfaces**

Shown is the use case diagram for the Air Quality Management System. The actor portrays any user that wishes to use the system, which includes a wide array of people. The actor can access information on overall sensor data that is aggregated. The user can adjust the time frame looked at by the system. The user can select an individual sensor, which in turn shows the details and reading of that sensor.

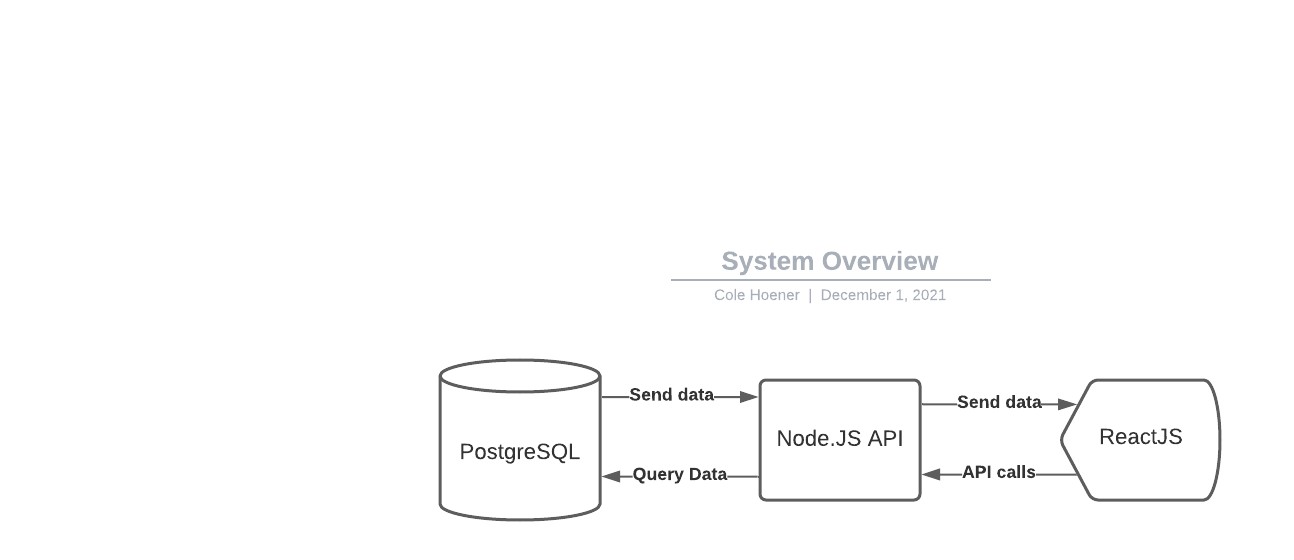


Shown above is a mockup of the user GUI. It shows the major components and overall look of the front interface for the application.

## **Hardware Interfaces**

*N/A -* All software is on one hardware device for this system.

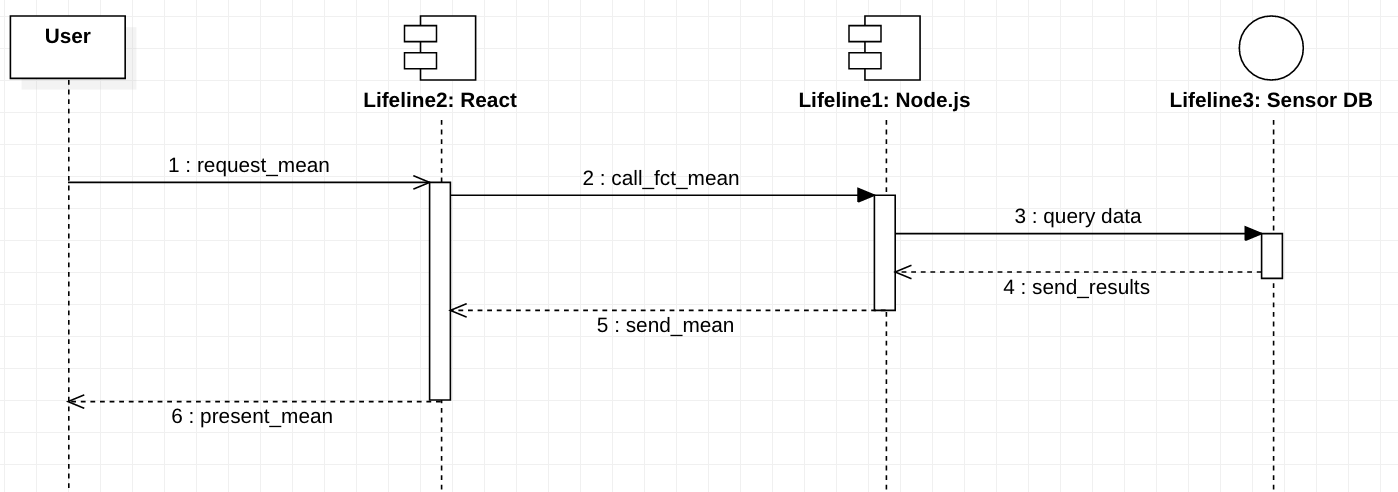
## **Software Interfaces**



The system will work as a common web application. The front end will use the ReactJS framework. Any data calls will be through the Node.JS API through only GET requests. The Node.JS API will communicate with the backend PostgreSQL database to pull any needed data.

## **Communications Interfaces**

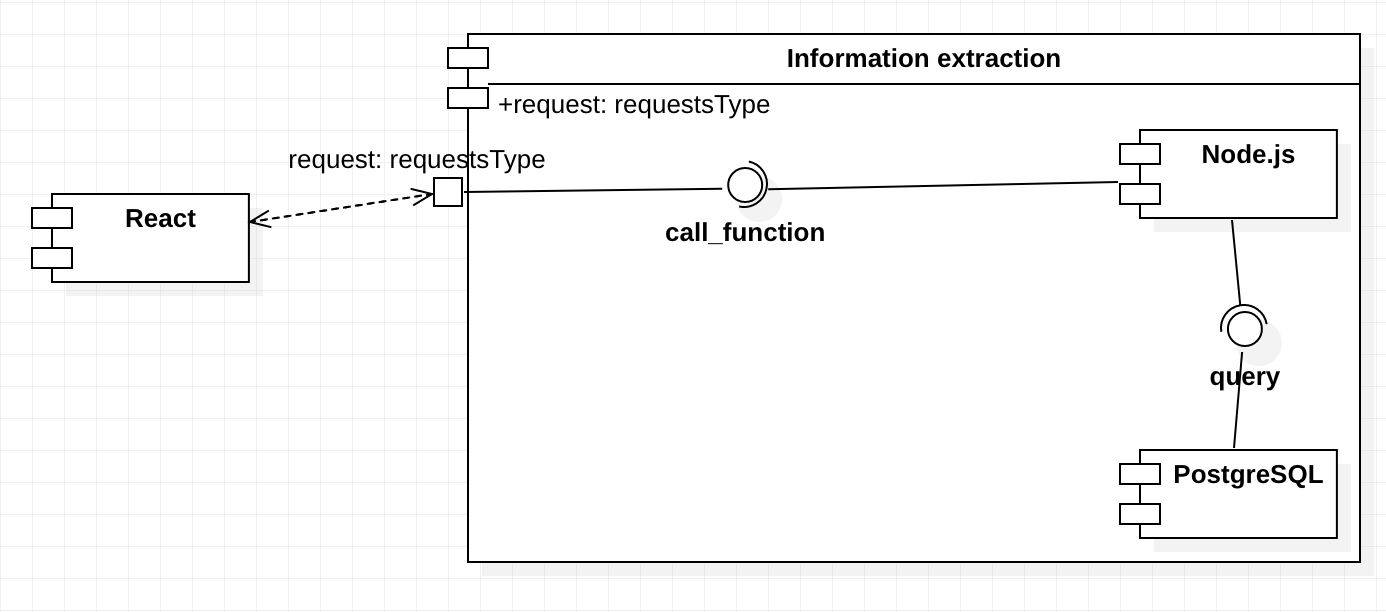
The user communicates with the website via the front-end React. Through Node.js our React application is connected to PostgreSQL so we can process HTTP-requests. If the user makes an HTTP-request and wants to calculate the mean, it sends a function call to Node.js, which calls the mean-function. This performs a query with SQL in the Database with our sensor data and sends the results back to the Node.js and gets presented on our frontend for the user.

**

# **System Features**

*<This template illustrates organizing the functional requirements for the product by system features, the major services provided by the product. You may prefer to organize this section by use case, mode of operation, user class, object class, functional hierarchy, or combinations of these, whatever makes the most logical sense for your product.>*

## Information extraction

**

4.1.1 Description and Priority

The information extraction is used to perform queries on the Database and generate an output for the website.

4.1.2 Stimulus/Response Sequences

This was already described in the communication interface.

4.1.3 Functional Requirements

The requests from the user must be manageable with a SQL request. This is already limited through the actions that can be performed on the user interface. Important is the timeframe. The user can either perform a query on multiple days or one day and define which hours are used for the calculation. The website shows a button with a range and a field with the possible dates to choose (year 2015). Due to this the input is always verifiable and unambiguous.

# **Other Nonfunctional Requirements**

## **Performance Requirements**

There are no performance requirements associated with this software system.

## **Safety Requirements**

There are no safety requirements associated with this software system.

## **Security Requirements**

There are no security requirements associated with this software system.

## **Software Quality Attributes**

The software quality attributes include, but are not limited to, just code cleanliness and correctness. Common code documentation and formatting should be followed for future readability.

## **Business Rules**

There are no business requirements associated with this software system, as everyone can access and use the system regardless of who they are..

# **Other Requirements**

*NA*

**Appendix A: Glossary**

AQMS - Air Quality Management System

**Appendix B: Analysis Models**

*<Optionally, include any pertinent analysis models, such as data flow diagrams, class diagrams, state-transition diagrams, or entity-relationship diagrams*.>

**Appendix C: To Be Determined List**

*<Collect a numbered list of the TBD (to be determined) references that remain in the SRS so they can be tracked to closure.>*