SOFTWARE REQUIREMENTS SPECIFICATION

for

AirWatcher

Version 1.0 approved

Prepared by Adrien Jaillet, William Jean, Matheus de Barros Silva, Jade Prévôt, Brandon da Silva Alves

INSA Lyon, Group B3204 - B3025

May 2, 2021

Contents

List of Figures

List of Tables

Version History

Version	Date	Authors	Description
0.1	2021-04-03	Adrien Jaillet, William Jean, Matheus de Barros Silva, Jade Prévôt, Brandon da Silva Alves	• Document creation from template that respect IEEE standard. • Add of *Introduction* and *Overall Description* sections. • Draft of *External Interface Requirements* and *System Features* sections.
0.2	2021-04-20	Adrien Jaillet, William Jean, Matheus de Barros Silva, Jade Prévôt, Brandon da Silva Alves	• Add of *External Interface Requirements*, *System Features* and *Other Nonfunctional Requirements* sections.
1.0	2021-05-02	Adrien Jaillet, William Jean, Matheus de Barros Silva, Jade Prévôt, Brandon da Silva Alves	• Document finalization

Chapter 1

Introduction

1.1 Purpose

The purpose of the document is to collect and analyze all assorted ideas that have come up to define the system, its requirements with respect to consumers. Also, we shall predict and sort out how we hope this product will be used in order to gain a better understanding of the project, outline concepts that may be developed later, and document ideas that are being considered, but may be discarded as the product develops.

In short, the purpose of this Software Requirements Specification (SRS) document is to provide a detailed overview of our software product, its parameters and goals. This document describes the project's target audience and its user interface, hardware and software requirements. It defines how our client, team and audience see the product and its functionalities. Nonetheless, it helps any designer and developer to assist in Software Delivery Lifecycle (SDLC) processes.

1.2 Document Conventions

1.3 Intended Audience and Reading Suggestions

This SRS is addressed to all stakeholders of the software: the governmental agency which need to monitor the air quality, the air cleaner providers which want to observe the impact of the cleaners on air quality and private individuals who participate in air quality data generation by installing fixed sensors at their homes. It is also addressed to developers who will develop the application and administrators responsible for software maintenance. The project manager can also follow the progress of the software specifications by reading this document.

End users must focus their reading on *System Features* while developers and administrators should also pay attention to *External Interface Requirements* and *Other Nonfunctional Requirements.*

1.4 Project Scope

The *AirWatcher* application is a console-based application that helps its users monitor the air quality as well as the quality of the data provided by the sensors used for data collection. This application is designed to improve air quality analysis by providing efficient algorithms for calculating indicators as well as checking the data reliability.

More specifically, the application provide, on demand metrics for a circular area specified by the user. Besides, the AirWatcher will continuously analyse data in order to identify malfunctioning sensors as well as whether the defect is willingly made or naturally caused.

Moreover, the application will also be responsible for managing the Governments' points system, which attributes a point to a private sensor owner each time their sensors' data are requested in a query for analysis. On the other hand, if the application identifies that the user is willingly distorting their sensors data, AirWatcher will mark the user as non-applicable for being awarded points.

By doing so, the application will allow the government agency to really monitor the air quality in order to taking the right measures for maintaining or improving air quality and as a consequence, the quality of life.

1.5 References

To calculate the air quality using differents measurements, developers must base their calculs on the ATMO index which is described here:

https://fr.wikipedia.org/wiki/Indice_de_qualit%C3%A9_de_l%27air

Chapter 2

Overall Description

2.1 Product Perspective

The application is part of a more global project. The project consists of a client server application. Sensors send their data to the server where AirWatcher is running. The data collection is out of scope of AirWatcher and is supposed to be fully operational. AirWatcher is a response to a governmental need. Indeed government agencies want to be able to monitor the air quality of specific areas so they could be able to take decisions (for example, install air cleaners in polluted zones) to improve the quality of life.

The data collection application and AirWatcher do not have to interface together. The only constraint is that the data collection, which will imply data set updates, and AirWatcher should not run at the same moment. If they do, the updates are not considered on the results provided by AirWatcher.

2.2 Product Functions

The AirWatcher application will allow analysis of the data generated by a sensor to make sure that it is functioning correctly and otherwise identify and maintain malfunctioning sensors. The application will aggregate the collected information to produce statistics such as the calculation of the mean of the quality of air in a specific area specified by the user. The mean of the quality of air can be calculated for a given moment as well as for a specified period of time. The application will also enable selecting one sensor and then scoring and ranking all other sensors in terms of similarity to the selected sensor. The similarity will be based on the data generated by the sensors during a specified period of time. The purpose of this functionality is to identify areas with similar air quality. The Application must produce the value of air quality at a precise geographical position in the territory at a given moment. The agency and the providers could use AirWatcher to observe the impact of the cleaners on air quality, for example, the radius of the cleaned zone, the level of improvement in air quality, etc... Finally, the application will provide a console based user interface to its different users.

2.3 User Classes and Characteristics

There are four types of users that interact with the system: the government agency, the providers of ”air cleaners”, the private individuals who participate in air quality data generation by installing fixed sensors at their homes and administrators. Each of these four types of users has different use of the system so each of them has their own requirements.

The government agency is able to analyse the data generated by a sensor to make sure that it is functioning correctly. Then it will be able to identify and maintain malfunctioning sensors. There is only one government agency but we can suppose it will use the software relatively frequently. This user can use all the functionalities associated with statistics. This is the most important user of our application.

The providers use AirWatcher to observe the impact of the cleaners on air quality, for example, the radius of the cleaned zone, the level of improvement in air quality, etc... There is less than 100 such kind of users. This user can also use all the functionalities associated with statistics with an relatively high frequency.

The private individuals participate in air quality data generation by installing fixed sensors at their homes. These types of users are very numerous but most of them does not use the software frequently. This user also have access to all the the functionalities associated with statistics.

Each user can manage its account. They all have a username and password.

2.4 Operating Environment

The AirWatcher application operates in any modern operating system without any particular restrictions. It uses a client-server architecture. The software runs on the server while end users send their data. The application is compiled using a *10.2.0+ g++* version. It may work with an older version but it has not been tested.

2.5 Design and Implementation Constraints

The project must be developed in *C++*. The application must not modify the data files. The algorithms used to analyse the data must be efficient. Therefore, the performance of the algorithms must be measurable. The metric of performance will be the duration of execution of an algorithm, measured in milliseconds.

The application should be developed following the Model View Controller (MVC) architecture pattern. Among all the different architectures that we had, we chose the Model-View-Controller (MVC) architecture because it seems to be the most appropriate one. In fact, the main objective of this application is to calculate and transmit data about the air quality. The MVC architecture allows the communication between the view (user interface) and the model (which contains data) and this is a great advantage for efficiency of the application. Moreover, MVC architecture allows to maintain and test relatively easily the UI code.

2.6 User Documentation

The application provides a user manual which consists of a basic description of the user interface. This user manual follows the *Linux man-pages* format.

2.7 Assumptions and Dependencies

We assume that the data the application needs (on sensors, providers, users, cleaners, etc...) is present and precisely located on the server. We alse assume that those files are correct plain text documents in csv format. The software is not responsible for the data collection. We therefore suppose that the data is collected beforehand. We also take for granted that the data will not be modified while the software is running.

The software does not rely on any dependency.

Chapter 3

External Interface Requirements

3.1 User Interfaces

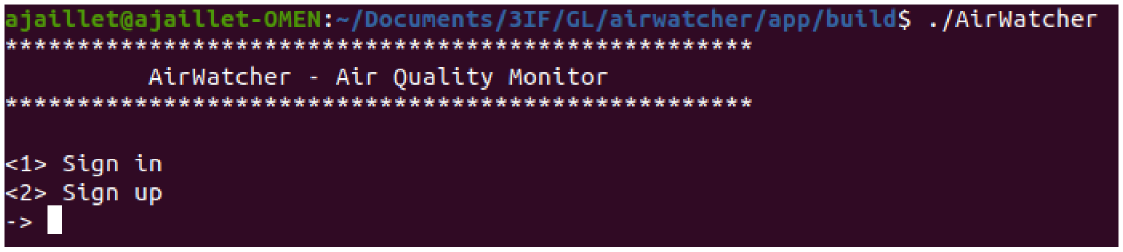
There is one command line user interface. Once the program is running, the user is prompted to inform his username and password. If the credentials are recognized by the application the access is granted and the menu appears. If not, the credentials are asked again.

The menu of government agency and air cleaner provider consists of 8 options:

1. get statistics at a specific time
2. get statistics in a time interval
3. rank sensors in function of similarity
4. get the value of air at a specified position
5. get cleaners impact on the air quality
6. modify my account
7. logout
8. shutdown

The menu of private individuals consists of 9 options:

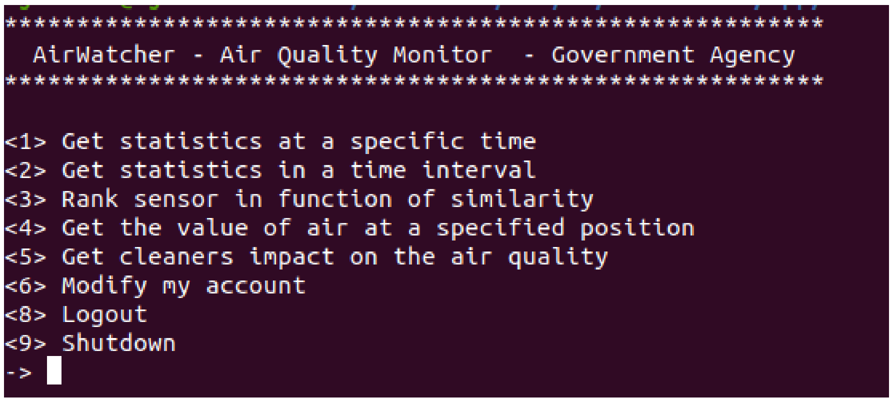
1. get statistics at a specific time
2. get statistics in a time interval
3. rank sensor in function of similarity
4. get the value of air at a specified position
5. get cleaners impact on the air quality
6. modify my account
7. Get my personal score
8. logout
9. shutdown



```
ajaillet@ajaillet-OMEN:~/Documents/3IF/GL/airwatcher/app/build$ ./AirWatcher
****************************************************
          AirWatcher - Air Quality Monitor
****************************************************

<1> Sign in
<2> Sign up
->
```

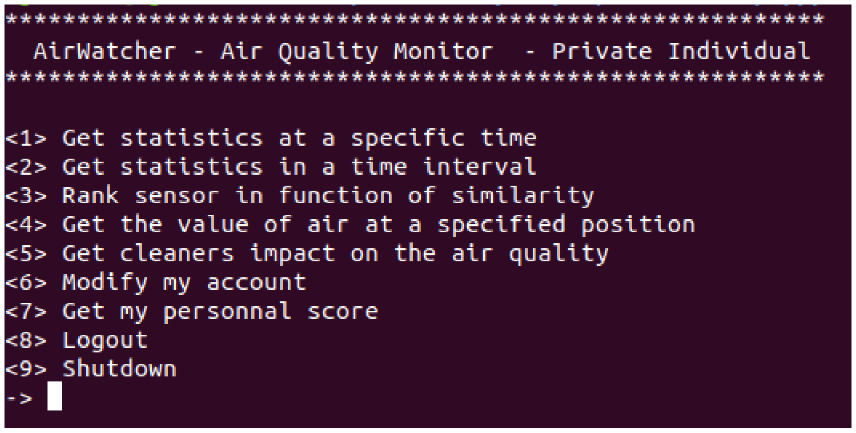
Figure 3.1: GUI - Log-in / Sign-in Menu



```
*********************************************************
  AirWatcher - Air Quality Monitor  - Government Agency
*********************************************************

<1> Get statistics at a specific time
<2> Get statistics in a time interval
<3> Rank sensor in function of similarity
<4> Get the value of air at a specified position
<5> Get cleaners impact on the air quality
<6> Modify my account
<8> Logout
<9> Shutdown
->
```

Figure 3.2: GUI - Government Agency Menu



```
*********************************************************
  AirWatcher - Air Quality Monitor  - Private Individual
*********************************************************

<1> Get statistics at a specific time
<2> Get statistics in a time interval
<3> Rank sensor in function of similarity
<4> Get the value of air at a specified position
<5> Get cleaners impact on the air quality
<6> Modify my account
<7> Get my personnal score
<8> Logout
<9> Shutdown
->
```

Figure 3.3: GUI - Private Individual Menu

3.2 Software Interfaces

The application will load data from csv files locally using a File System reader.

Chapter 4

System Features

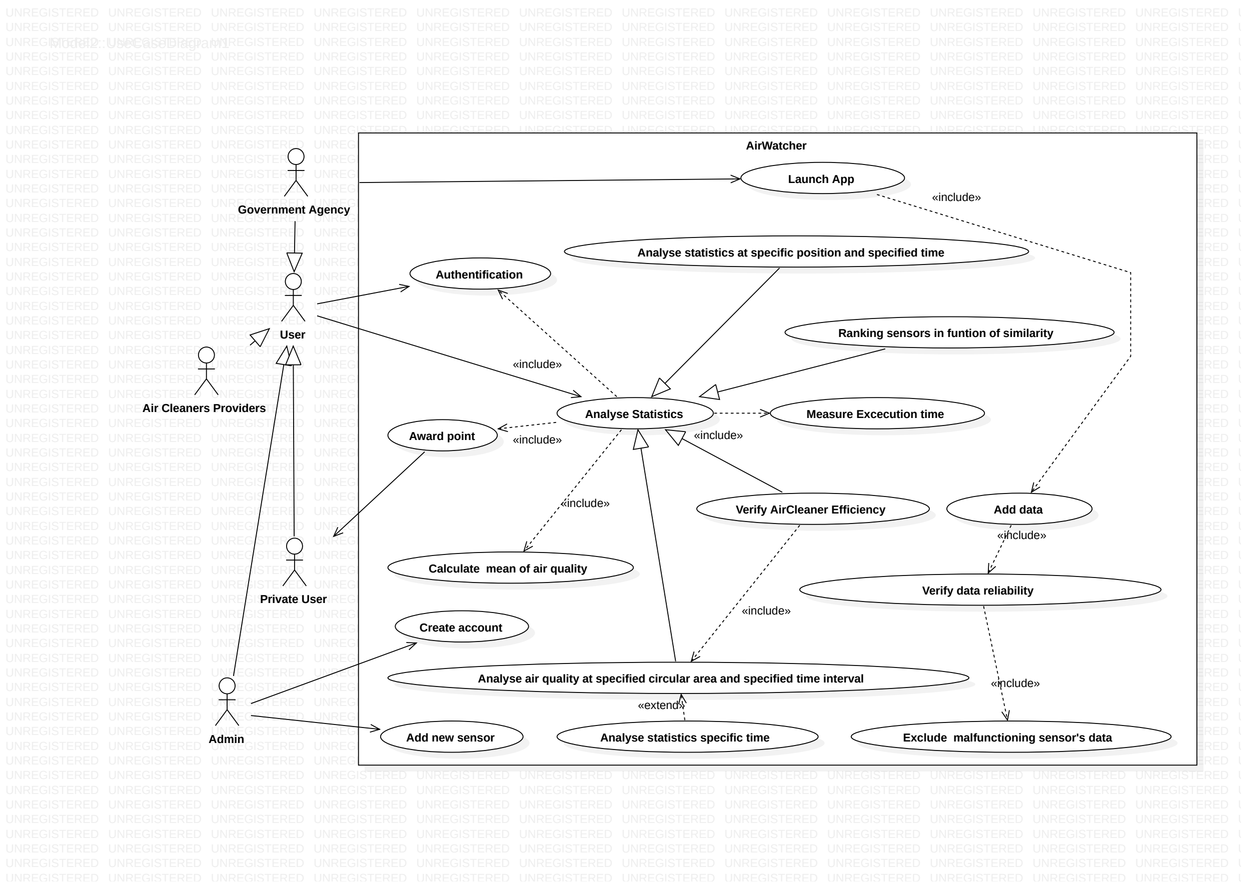


Figure 4.1: Use Case Diagram

4.1 User Features

4.1.1 UC-1: Log In

Description and Priority

This feature allows a user to authenticate himself. It has a low priority level.

Stimulus/Response Sequences

1. The system asks for the login of the user;
2. The user refers its login;
3. The system asks for the password of the user;
4. The user refers its password;
5. The system login the user.

Functional Requirements REQ-1

Use Case	Log in
Description	Allow a user to log in
Inputs	login, password
Precondition	Nobody is log in
Post-condition	User logged in
Output	User

Table 4.1: UC-1 Functional Requirements

4.1.2 UC-2: Calculate Mean of Air Quality at a specified circular area

Description and Priority

This feature permits to show the mean of air quality at a specified circular area. It has a high priority level.

Stimulus/Response Sequences

1. The system shows the different options of the menu;
2. The user choose the option 4;
3. The system ask for a circular area;
4. The user enters a specific area;
5. The system shows the mean of air quality at this position.

Functional Requirements REQ-2

Use Case	Calculate mean of air quality
Description	This functionality returns the mean of air quality in terms of inputs
Inputs	longitude, latitude, radius, begin, end
Precondition	user logged
Post-condition	
Output	The mean of air quality in the circle defined by a point (longitude and latitude) at a certain moment (between begin and end).

Table 4.2: UC-2 Functional Requirements

4.1.3 UC-3: Log Out

Description and Priority

This feature permits to logout the user. It has a low priority level.

Stimulus/Response Sequences

1. The system shows the different options of the menu;
2. The user choose the option 9;
3. The system logout the user.

Functional Requirements REQ-3

Use Case	Log out
Description	Allow a user to log out
Inputs	
Precondition	User logged in
Post-condition	
Output	

Table 4.3: UC-3 Functional Requirements

4.1.4 UC-4: Compare and rank all sensors with a specific one.

Description and Priority

This feature permits to rank sensors compared to a selected one. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a SensorId;
2. The user refers a SensorId;

3. The system compare all sensors to this one;
4. The system ranks all sensors. The sensor which is the closest to the referred is on top of the list. The farthest is the last one of the list;
5. The system print the list to the user.

Functional Requirements REQ-4

Use Case	Rank sensors compared to a specific one.
Description	Allows an user to rank every sensor compared to a specific one. This feature permits to identify areas with a similar air qualities.
Inputs	SensorID
Precondition	User logged in. The selected sensor must not be malfunctioning
Post-condition	
Output	A rank of sensors compared to the selected one.

Table 4.4: UC-4 Functional Requirements

4.1.5 UC-5: Analyse air quality at specific position and specified time.

Description and Priority

This feature permit to get the air quality at specific position and specified time. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a Longitude;
2. The user refers a longitude;
3. The system asks for a latitude;
4. The user refers a latitude;
5. The system asks for a begin timestamp;
6. The user refers a begin timestamp;
7. The system asks for a end timestamp;
8. The user refers a end timestamp;
9. The system calculates the air quality at the specified position and the specified time;
10. The system print the value.

Functional Requirements REQ-5

Use Case	Analyse air quality at specific position and specified time.
Description	Allow a user to analyse air quality at specific position and specified time. The application must return a value even if there is no sensor at the specified location.
Inputs	longitude, latitude, date
Precondition	
Post-condition	
Output	The air quality at the specified location and date.

Table 4.5: UC-5 Functional Requirements

4.1.6 UC-6: Analyse AirCleaner efficiency

Description and Priority

This feature allows the user to get stats about the air Cleaner efficiency such as the radius of AirCleaner impact. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a CleanerId;
2. The user refers a CleanerId;
3. The system calculates the size of area impacted by the AirCleaner. It uses cleaner's property to get begin timestamp and end timestamp;
4. The system print the radius of the AirCleaner Impact;

Functional Requirements REQ-6

Use Case	Analyse AirCleaner efficiency.
Description	Allow a user to analyse AirCleaner efficiency. The application analyse data at the beginning and at the end of AirCleaner operation.
Inputs	CleanerId
Precondition	
Post-condition	
Output	The radius of AirCleaner impact.

Table 4.6: UC-6 Functional Requirements

4.2 Government Agency Features

4.2.1 UC-7: Analyse data of specific sensor

Description and Priority

This feature allows the user to analyse data of a specific sensor and to know if the sensor is malfunctioning. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a SensorId;
2. The user refers a SensorId;
3. The system calculates and returns if the sensor is malfunctioning or not. It bases its calculations on the sensor's measurements.
4. The system print if the referred sensor is malfunctioning;

Functional Requirements REQ-7

Use Case	Analyse data of specific sensor
Description	Allow a user to analyse data of a specific sensor. If the user is a private user he can only analyse data of his own sensor.
Inputs	SensorID
Precondition	User logged in. If it's a private user, the sensor must be his.
Post-condition	
Output	If the sensor is malfunctioning.

Table 4.7: UC-7 Functional Requirements

4.2.2 UC-8: Exclude malfunctioning sensors.

Description and Priority

This feature allows the user to exclude all malfunctioning sensor. It has a high level of priority.

Stimulus/Response Sequences

1. The system find all malfunctioning sensors using the feature from the U-C 6;
2. The system considers the found sensors as malfunctioning.

Functional Requirements REQ-8

Use Case	Exclude malfunctioning sensors.
Description	The governmental agency can exclude a malfunctioning sensor. Then they wouldn't be considered anymore for stats.
Inputs	SensorID
Precondition	Sensor is not soon considered as malfunctioning
Post-condition	Sensor is considered as malfunctioning
Output	

Table 4.8: UC-8 Functional Requirements

4.3 Private Individual Features

4.4 Administrator Features

4.4.1 UC-9: Create Account

Description and Priority

This feature allows the admin user to create an new use account. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a login;
2. The user refers a login;
3. The system asks for a password;
4. The user refers a password;
5. The system asks for a role;
6. The user refers a role;
7. The system adds a new user to the application using these informations.
8. The system prints the login, password and role.

Functional Requirements REQ-9

Functionality 3	Create account
Description	Allow an admin user to create an account
Inputs	login, password, role
Precondition	No account with this login and password already exists
Post-condition	Account created
Output	login, password, role

Table 4.9: UC-9 Functional Requirements

4.4.2 UC-10: Add new sensor

Description and Priority

This feature allow the admin user to add a new sensor in the application. It has a high level of priority.

Stimulus/Response Sequences

1. The system asks for a SensorId;
2. The user refers a SensorId;
3. The system asks for a longitude;
4. The user refers a longitude;
5. The system asks for a latitude;
6. The user refers a latitude;
7. The system adds a new sensor to the application using these informations.
8. The system prints the sensor.

Functional Requirements REQ-10

Functionality F4	Add new sensor
Description	Allow the admin user to add a new sensor to the application
Inputs	SensorId, longitude, latitude
Precondition	A sensor with the same ID does not exist
Post-condition	The sensor exists
Output	Sensor

Table 4.10: UC-10 Functional Requirements

Chapter 5

Other Nonfunctional Requirements

5.1 Performance Requirements

The government agency would like the algorithms (analyzing a sensor, calculating mean, comparison of sensors, etc) of AirWatcher to be efficient. They must be performing. The application must be able to measure their performance, through the duration of execution of an algorithm measured in milliseconds.

5.2 Safety Requirements

There is the possibility that a private individual may act maliciously and corrupt their sensor in order to provide false data. Therefore, the AirWatcher application will allow the government agency to analyze the data provided by a private individual's sensor and classify its behavior as reliable or unreliable. If a private individual is detected to provide unreliable data, their entire data will be marked as false and will be excluded from all further queries on the application. This will prevent the user from gaining any further points. Besides, the application is also required to verify the files format in order to always have well formed documents. By doing so, the application's functionalities are going to rely on data constructed deliberately to serve as inputs for AirWatcher. Furthermore, it is important to observe that the data reliability is not supported by AirWatcher.

5.3 Security Requirements

Like every other application, AirWatcher is likely to be attacked. Nonetheless, the risks an attack could present are limited. That is due to the fact that all of the applications functionalities are read only regarding it's data. Therefore we are able to conclude that the only risks presented are related to one unauthorised person reading confidential data. Even so, as the application is constructed in a way that it does not store any users' sensible data, in terms of security, the risks are limited.

System	AirWatcher Application
Assets	Sensors' data on air quality
Vulnerabilities	Weak Passwords
Attacks	Discovering one's password
Risks	Non authorised people having access to data and to the functionalities
Counter measures	Checking the passwords entropy in order not to allow users having a weak password

Table 5.1: Security Analysis

The application must allow the government to be able to thwart individuals who corrupt their sensor, to remedy it and thus guard against this type of behavior.