INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

UML PROJECT : *Air WATCHER* System

Members of the group : Ines CHEBBI, Tom SANCHEZ, Yassine TAHRASTE, Omar TKITO, Asmae TOUITI

Introduction

AirWatcher is an application designed to monitor air quality using data collected from sensors installed across a territory. It allows users to analyze the measurements, verify whether a sensor is functioning correctly, and perform statistical analysis of air quality in a defined area.

The application operates solely via the command line and uses CSV files to store sensor data. AirWatcher offers several features, such as comparing data from different sensors, estimating air quality in areas without sensors, and evaluating the impact of industrial air purifiers.

AirWatcher will primarily be used by government agencies, companies providing air cleaners, and private individuals contributing data through their sensors. The application helps identify sensor issues and track the impact of actions taken to improve air quality.

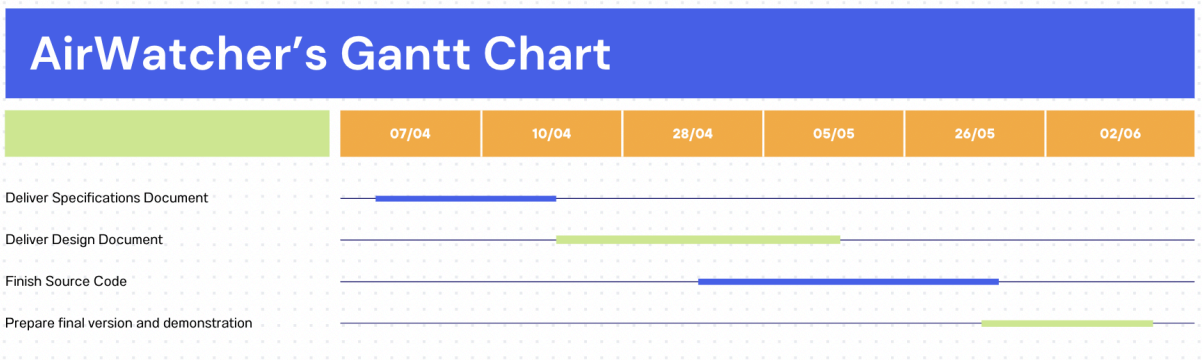
Planning

1. The teams

This project is carried out by two teams:

- 1 team of two: CHEBBI Ines and TAHARASTE Yassine.
- 1 team of three: TOUITI Asmae, TKITO Omar, and SANCHEZ Tom.

2. Gantt chart



I) Functional requirements

1. **AirWatcher is an application that shall be able to monitor air quality in an area equipped with sensors.**
   • It shall read sensor data from CSV files stored locally on a central server.
   • These sensors shall measure several air pollution-related attributes (O_3, NO_2, PM10, and SO_2).

2. **AirWatcher shall allow users to verify whether a specific sensor is functioning properly.**
   • By analyzing the evolution of its readings over time, the application shall detect whether the sensor is faulty or providing inconsistent data.
   • This feature shall help the government agency maintain a reliable sensor network.

3. **AirWatcher shall provide statistical analysis tools to evaluate air quality in a specified geographical area.**
   • The user defines a circular area on which the application shall perform various statistical analyses.
   • This shall include calculating average air quality at a given time or over a given period.

4. **AirWatcher shall allow comparison of a selected sensor's data with that of other sensors to detect similarities.**
   • The user selects a sensor and an analysis period.
   • The application shall score and rank other sensors based on the similarity of their readings to those of the reference sensor.

5. **AirWatcher shall provide an estimate of air quality at a given geographic location, even in the absence of a sensor.**
   • The estimate shall be based on measurements from the nearest available sensors at that time.

6. **AirWatcher shall enable analysis of the impact of industrial air purifiers on surrounding air quality.**
   • The user selects a cleaner and observes its impact on air quality.

• The application shall provide information on the observed improvement and the potentially affected area.

7. **AirWatcher shall automatically award points to private users whose sensor data are used in queries.**
   • In order to encourage individuals to contribute to air quality measurements, one point shall be awarded each time a private sensor's data is used.

8. **AirWatcher must identify malicious private users.**
   • Private sensors that provide erroneous or falsified data must be detected.
   • Associated private users must be considered unreliable and all their data will be ignored in future queries (the user no longer earns points).

9. **AirWatcher shall measure the execution time of each algorithm to help assess and optimize performance.**
   • When a user runs a query or analysis (such as averaging air quality, comparing sensors), the application shall measure how long the operation takes to complete.
   • The execution time is reported in milliseconds and shall be used by developers or administrators to evaluate the efficiency of the system.

II) Non functional requirements

Quality Requirements	**Usability**	• **Console-based UI**: The system must offer a text-based, command-line interface for interacting with the application. • **Role-based UI tailoring**: The interface should adjust to the user's role (e.g., government agent, provider, private individual) by showing relevant features.
	Reliability	• **Error handling**: The system should be robust to invalid data entries (e.g., corrupted or out-of-range sensor values). • **Fault tolerance**: Malicious or faulty user sensors must be detectable, and their data excluded without crashing the application. • **Data integrity & Access control** : The system must ensure that users cannot alter the CSV data files—data is read-only. Modifying the CSV files by users is prohibited.
	Performance	• **Response time**: The duration of algorithm execution must be measurable in milliseconds to allow performance benchmarking. • **Throughput**: All analysis algorithms (e.g., sensor analysis, mean air quality calculation, sensor similarity comparison) must be optimized for performance. • **Accuracy** : The system must be able to handle a large volume of data from many sensors and measurements without significant degradation in performance.
	Supportability	• **Local execution**: The application must run locally on the central server, with no remote access or network-based execution. All data (CSV files) must be read from the server's local storage.

		• **Separated architecture :** The system must be developed using a modular architecture (separation of concerns: user management, data management, sensors, etc.) to ease debugging and future updates.
Constraint Requirements	**Implementation**	• All related software associated with the application will be written using C++. • All the data used will be stored in CSV files. • L'application n'aura pas d'interface. Les interactions seront menées en ligne de commande.

III) Analysis of security risks

System	AirWatcher
Attackers	**Malicious user (with a private sensor)**: has access to the system but intends to manipulate or falsify data. **External attacker (hacker, cybercriminal, etc.)**: seeks to intercept data or compromise the system's integrity. **Unauthorized access**: users/attackers trying to access data without permission (e.g., accessing a user's personal information).
Assets	**Air quality data** (sensor measurements). **Users' personal information** (email, password). **Points**. **Results from statistical analyses**.
Vulnerabilities	Weak passwords. Transmission and storage of data without encryption. Errors in algorithms.

Attack	Risks	Countermeasures
– The attacker guesses or steals a weak password to log into the application.	**- (High)** The attacker gains unauthorized access to sensor and user data. **- (Medium)** The attacker manipulates the points system (possibility of influencing the points earned by other accounts	• Passwords are validated and weak ones are rejected: implement a strong password policy (minimum length, special characters). • Force periodic password updates.

	through precise queries).	• Account lockout after multiple failed attempts • Implement two-factor authentication. • Continuous monitoring of suspicious login attempts and brute force detection.
– The user attempts to access unauthorized data (e.g. private users should not have access to cleaner data).	**- (Medium**) Unauthorized disclosure of restricted data.	• Role-based access control : restrict access to data to authorized roles only.
– The attacker injects malicious data into the system.	- **(High)** The attacker directly modifies sensor data, e.g., alters air pollution readings or falsifies statistical analysis results. **- (High)** A malicious private individual corrupts his sensor in order to supply erroneous data. **- (Medium)** The attacker manipulates the point system by directly adding or removing points from a user. - **(Medium)** Service disruption/interruption (data overload).	• Restrict data access to read-only (no deletion/modification for users). • Detect private sensors that provide erroneous or falsified data, and ignore their data in future queries.
– The user attempts to manipulate the point system (e.g., fraud or abuse).	- **(Medium)** The user attempts to exploit or abuse the point system (via bots or repeated requests).	• Use CAPTCHA to verify that actions are performed by humans. • Monitor point activity for anomalies or abuse patterns.

		• Add rate limiting or logic checks to detect repeated or automated actions
– The attacker intercepts network communications between users and the server to steal sensitive data (email, password, cleaners data, …).	**- (High)** The attacker gains unauthorized access to sensor and user data.	• Encrypt sensitive data (email, password, …). • Use a valid and up-to-date SSL certificate. • Monitor network connections to detect interception attempts.
– The attacker uses malicious code, such as a virus, worm, or trojan, to infect the system or gain unauthorized access.	- **(High)** The attacker gains unauthorized access to sensor and user data. **- (High)** The attacker deletes crucial air quality data. **- (Medium)** Service disruption/interruption.	• Use antivirus software and firewalls to protect the system from malware. • Regularly update all systems and software to patch security vulnerabilities. • Raise user awareness about cybersecurity to avoid downloading malicious files and phishing attacks.

IV) Validation Tests

Requirement	Tests	Expectation
AirWatcher shall read sensor data from CSV files stored locally on a central server.	Provide a correctly formatted CSV file with valid data.	Check that the system successfully loads and parses the file.
	Provide a CSV file with invalid or corrupted entries or a non CSV file.	Check that the system logs an error or skips the faulty lines.
	Provide an empty CSV file.	Check that the system handles it without crashing and shows an appropriate message.
AirWatcher shall verify whether a specific sensor is functioning properly by analyzing the evolution of its readings.	Provide consistent, realistic readings over time for a sensor.	Check that the sensor is marked as functioning.
	Provide erroneous data.	Check that the sensor is flagged as potentially faulty.
	Provide missing or null values for a period.	Check that the system detects irregularity and raises a warning.
AirWatcher shall provide statistical analysis tools to evaluate air quality in a specified geographical area.	Define a circular area with multiple sensors inside.	Check the correct average air quality calculation.

	Define an area with no sensors.	Check that the system displays a warning but still gives a coherent computation according to our triangulation technique
	Define a valid area and time range.	Check that statistics match expected values for given mock data.
AirWatcher shall allow comparison of a selected sensor's data with that of other sensors to detect similarities.	Provide one sensor with a known similar pattern to another.	Check that it is ranked highly in similarity.
	Provide multiple sensors with different patterns.	Check the correct ranking by similarity score.
	Provide no sensors for comparison.	Check system raises an exception.
AirWatcher shall provide an estimate of air quality at a given geographic location, even in the absence of a sensor.	Provide known surrounding sensor data.	Check that estimation is a weighted average based on distance.
	Provide inconsistent or sparse sensor data.	Check that the system returns estimation doesn't take this set of data into account in its calculation.
	Provide no nearby sensors.	Check that the system returns an appropriate response respecting the triangulation method we will have defined. The hypothesis made being that

		we will always have nearby sensors.
AirWatcher shall enable analysis of the impact of industrial air purifiers on surrounding air quality.	Define a purifier and provide before/after data in its vicinity.	Check that improvement is correctly calculated and displayed.
	Provide no change in air quality around the purifier.	Check that the system reports no significant impact.
	Select a purifier with no data.	Check that the system handles missing data properly, by returning an appropriate exception.
AirWatcher shall automatically award points to private users whose sensor data are used in queries.	Use a private individual sensor's data in an analysis.	Check that the user earns 1 point per function call.
	Run analysis with no private sensors involved.	Check no points are awarded.
	Reuse the same private sensor multiple times.	Check that the point counter increments correctly each time.
AirWatcher must identify malicious private users.	Provide a private sensor with unrealistic values (example : negative PM10).	Check that the sensor is flagged and ignored in future analyses.
	Use a flagged sensor in a query.	Check that no points are awarded, and the data is not used.

	Correct the sensor data and attempt to reuse it.	Check if the system allows re-validation or keeps it blocked. The system shall allow re-validation.
AirWatcher shall measure the execution time of each algorithm to help assess and optimize performance.	Run a basic query (e.g., average air quality in one zone).	Check that execution time is recorded in milliseconds.
	Run a complex similarity analysis.	Check that execution time reflects processing duration.
	Run an operation and cancel midway.	Check if partial timing is handled or discarded.

INSA INSTITUT NATIONAL DES SCIENCES APPLIQUÉES LYON

V) User Manual

TABLE OF CONTENTS

OBJECTIVE OF THE APPLICATION

AirWatcher is an application designed to analyze air quality.
This analysis is based on data collected from sensors.
AirWatcher is intended for professionals (such as government agencies and air purification system providers), but also for curious individuals.

The available features include:

- Identifying malfunctioning sensors
- Monitoring the effects of air purifiers
- Comparing air quality across different areas
- Estimating air quality at a specific location

LAUNCHING THE APPLICATION :

Prerequisites:

The program airwatcher.exe must be installed.
The following files must be present in the data/ folder:

```
AirWatcher/
├── data/
│   ├── sensors.csv
│   ├── measurements.csv
│   ├── attributes.csv
│   ├── providers.csv
│   ├── cleaners.csv
│   └── users.csv
```

Interaction with the application is done via the console.

On LINUX systems:

- Open a terminal
- Type: ./airwatcher

On WINDOWS:

- Open PowerShell
- Type: airwatcher

HOW TO USE THE APPLICATION ?:

After launching the application, the following is displayed in the console:

```
Role:
```

If the user is a government agent, they will see the following menu:

```
Rôle : Agent gouvernemental
```

The following menu is displayed in the console:

```
===== AIRWATCHER – ADMIN MENU =====
   1. Check if a sensor is functioning correctly
   2. Calculate the average air quality in a geographic area
   3. Compare sensors with each other (similarity)
   4. Estimate air quality at a specific geographic point
   5. Analyze the effect of an air purifier
   6. Collect data from a specific sensor
   7. Measure the performance of the algorithms
   8. Exit
Enter your choice:
```

If the user is a provider of an air purifier, they will see the following menu:

```
Role: PROVIDER
```

The following menu is displayed in the console:

```
===== AIRWATCHER – PROVIDER MENU =====
   1. Analyze the effect of an air purifier
   2. Exit
Enter your choice:
```

If the user is a private user

```
Role : Citizen
```

The following menu is then displayed in the console:

```
===== AIRWATCHER – GUEST MENU =====
   1. View air quality in a geographic area
   2. Estimate air quality at a specific point
   3. Check my points
   4. Exit

Enter your choice:
```

Simply enter the number corresponding to the desired service and press Enter.

EXAMPLES OF FEATURE USAGES:

Check if a sensor is functioning correctly:

```
Enter the ID of the sensor to check:
```

Two cases are possible:

```
The sensor Sensor12 is not working properly. Its data will no longer be
taken into account.
```

OR

```
The sensor Sensor12 is working correctly.
```

Calculate air quality in a zone:

```
==== AIR QUALITY CALCULATION ====

Latitude : 45.76
Longitude : 4.84
Radius (in km) : 5
Start : 2024-04-01 00:00:00
End   : 2024-04-01 23:59:59
```

The sensors measure 4 types of data:

- Ozone concentration (O3)
- Nitrogen dioxide concentration (NO2)
  Sulfur dioxide concentration (SO2)
- Fine particulate concentration (PM10)

The result is the average of all sensor readings within the selected time range and circular area.

```
==== RESULTS ====

Average air quality in the zone
Center: 45.76, 4.84 -- Radius: 5 km
Period: 01/01/2024 00:00 → 01/01/2024 23:59

- O3 (Ozone)                 : 47.25 µg/m³
- NO2 (Nitrogen Dioxide)     : 31.10 µg/m³
- SO2 (Sulfur Dioxide)       : 5.90 µg/m³
- PM10 (Fine Particles)      : 18.45 µg/m³

→ ATMO Index = 3 (Moderate air quality)
```

Compare sensors with each other:

```
===== SENSOR COMPARISON =====

Enter the reference sensor ID: Sensor15
Enter the start date: 2025-04-01 00:00:00
Enter the end date  : 2025-04-03 23:59:59
```

We calculate a correlation score between the O3, NO2, SO2, and PM10 values of others censors, called **the similarity score.**

The 5 sensors with the highest similarity scores (closest to 100%) are displayed.

```
==== RESULTS ====

Comparison with sensor: Sensor15
Period: from 2025-04-01 00:00:00 to 2025-04-03 23:59:59

Ranking of most similar sensors:

| Rank | Sensor     | Similarity Score (%) |
|------|------------|----------------------|
| 1    | Sensor42   | 96.7 %               |
| 2    | Sensor08   | 94.5 %               |
| 3    | Sensor03   | 93.2 %               |
| 4    | Sensor27   | 91.9 %               |
| 5    | Sensor34   | 89.6 %               |
```

Analyze the effectiveness of an air purifier:

```
===== AIR PURIFIER ANALYSIS =====

Enter the purifier ID: Cleaner03
Analysis radius: 3 km
```

We compare pollutant concentrations before (1 week before) and after (1 week after) the purifier's installation:

```
==== RESULTS ====

Change in air quality in the area:
Sensors used: 5 sensors detected within the radius

|Pollutant|Average BEFORE (µg/m³)|Average AFTER (µg/m³)| Improvement (%)
|----------|----------------------|---------------------|---------
| O3       | 52.6                 | 45.1                | -14.3 %
|
| NO2      | 34.2                 | 28.7                | -16.1 %
|
| SO2      | 6.8                  | 5.0                 | -26.5 %
|
| PM10     | 24.5                 | 20.3                | -17.1 %
|

Conclusion:

ATMO Index BEFORE treatment: 4 (Moderate air quality)
ATMO Index AFTER treatment : 3 (Fair air quality)

The air quality in the area has improved after the installation of the
purifier.
```

Collect data from a specific sensor:

The threshold value is the limit (in µg/m³) not to be exceeded.

The result will show how many times this value was exceeded.

Note: enter 0 if threshold information is not relevant.

```
===== COLLECT DATA FROM A SPECIFIC SENSOR =====

Enter the sensor ID: Sensor12
Start date (YYYY-MM-DD HH:MM:SS): 2025-04-01 00:00:00
End date   (YYYY-MM-DD HH:MM:SS): 2025-04-03 23:59:59

Threshold O3   : 0
Threshold NO2  : 40
Threshold SO2  : 0
Threshold PM10 : 40
```

All data from the sensor between the start and end dates is then displayed in the console.

```
==== RESULTS ====

Sensor: Sensor12
Location: Latitude 45.7612, Longitude 4.8423
Period: from 2025-04-01 00:00:00 to 2025-04-03 23:59:59

Total number of measurements: 288

Average [O3]    = 45 µg/m³
Average [NO2]   = 30 µg/m³
Average [SO2]   = 5.9 µg/m³
Average [PM10]  = 17 µg/m³

Min [O3]    = 38.2 µg/m³  → 01/04/2025 03:00:00
Min [NO2]   = 22.5 µg/m³  → 01/04/2025 06:00:00
Min [SO2]   = 4.2 µg/m³   → 02/04/2025 02:00:00
Min [PM10]  = 12.0 µg/m³  → 01/04/2025 00:00:00

Max [O3]    = 52.6 µg/m³  → 01/04/2025 14:00:00
Max [NO2]   = 38.0 µg/m³  → 01/04/2025 19:00:00
Max [SO2]   = 7.8 µg/m³   → 02/04/2025 11:00:00
Max [PM10]  = 21.9 µg/m³  → 01/04/2025 15:00:00

Standard deviation [O3]    = 4.1 µg/m³
Standard deviation [NO2]   = 3.9 µg/m³
Standard deviation [SO2]   = 0.8 µg/m³
Standard deviation [PM10]  = 2.3 µg/m³

Exceedances [PM10 > 20 µg/m³] : 5 times
Exceedances [NO2 > 40 µg/m³]  : 0 times

ATMO Sub-indices based on average values:
- O3   = 45 µg/m³   → ATMO index: 3
- NO2  = 30 µg/m³   → ATMO index: 2
- SO2  = 5.9 µg/m³  → ATMO index: 1
- PM10 = 17 µg/m³   → ATMO index: 3
```

```
Most polluted day  : 02/04/2025 → ATMO = 3  (Moderate)
Cleanest day       : 02/04/2025 → ATMO = 1 (Very good)
```

Measuring the performance of algorithms

To help government agents identify potential improvements to the application, this menu allows them to evaluate the performance of the various algorithms.

By analyzing execution times and efficiency, they can detect possible optimizations and ensure the system remains responsive and scalable.

```
===== PERFORMANCE TEST MENU =====

Select one or more algorithms to test (separated by commas):

1. Average calculation in a zone
2. Point-based air quality estimation
3. Sensor comparison
4. Sensor functionality check
5. All algorithms
0. Back
Your choice: 1,3,4
```

The results provide insight into the execution times of the algorithms, allowing for a temporal performance analysis.

```
===== SELECTED ALGORITHM TESTS =====

>>> TEST 1: Average calculation in a zone <<<

Number of trials: 3
Radius: 5 km
Tested sensors: Sensor01, Sensor08, Sensor14
Period: 2025-04-01 → 2025-04-02
```

```
[Sensor01] Time: 121 ms - Measurements: 190
[Sensor08] Time: 109 ms - Measurements: 186
[Sensor14] Time: 132 ms - Measurements: 202

>>> TEST 3: Sensor comparison <<<

Number of trials: 3
Reference sensors: Sensor03, Sensor17, Sensor22

[Sensor03] Time: 305 ms - Compared with: 28
[Sensor17] Time: 289 ms - Compared with: 28
[Sensor22] Time: 316 ms - Compared with: 28

>>> TEST 4: Sensor functionality check <<<
Tested sensors: Sensor05, Sensor19, Sensor33

[Sensor05] Time: 48 ms - Result: OK
[Sensor19] Time: 51 ms - Result: Inconsistent values detected
[Sensor33] Time: 46 ms - Result: OK
```

Note:

The ATMO index is calculated based on the concentration levels of four major air pollutants: ozone (O_3), nitrogen dioxide (NO_2), sulfur dioxide (SO_2), and fine particules (PM10).

For each of these pollutants, a sub-index is determined according to a predefined scale of concentration thresholds. These thresholds are divided into 10 levels, ranging from 1 (very good air quality) to 10 (extremely poor air quality).

Once the sub-indices are calculated for all four pollutants, the final ATMO index is equal to the highest (worst) of the four sub-indices. This ensures that the pollutant with the greatest health impact determines the overall air quality score.

Calculus of the sub-index:

Indice ATMO[24,22]	O_3	SO_2	NO_2^-	PM_{10}	Niveau
1	0 à 29	0 à 39	0 à 29	0 à 6	Très bon
2	30 à 54	40 à 79	30 à 54	7 à 13	Très bon
3	55 à 79	80 à 119	55 à 84	14 à 20	Bon
4	80 à 104	120 à 159	85 à 109	21 à 27	Bon
5	105 à 129	160 à 199	110 à 134	28 à 34	Moyen
6	130 à 149	200 à 249	135 à 164	35 à 41	Médiocre
7	150 à 179	250 à 299	165 à 199	42 à 49	Médiocre
8	180 à 209	300 à 399	200 à 274	50 à 64	Mauvais
9	210 à 239	400 à 499	275 à 399	65 à 79	Mauvais
10	≥ 240	≥ 500	≥ 400	≥ 80	Très mauvais

NAVIGATION AND INTERACTION :

To facilitate the navigation of the user, they are few shortcuts:

- The user can return to the previous menu at any time by entering `0`.
- If an input is left empty, the system will prompt the user again.
- Entering an invalid format (e.g., wrong date format or unknown sensor ID) will display an error message and allow a retry.
- To quit the application, select the “Exit” option in the main menu.

ERROR HANDLING AND MESSAGES :

The application includes basic error handling. Common messages include:

- Sensor not found — The entered ID does not exist in the dataset.
- Invalid date format — Dates must be entered as YYYY-MM-DD HH:MM:SS.
- No data found in the selected area — No active sensors or measurements available for the selected parameters.
- File not found — One or more required CSV files are missing from the /data folder.

The application will not crash but instead guide the user back to a valid input.

KNOWN LIMITATIONS :

- The application is designed for local execution only (no network support).
- CSV files must not be modified during execution.
- The interface is available in a single language (English).
- All data is considered static during runtime (no real-time updates).

CREDITS :

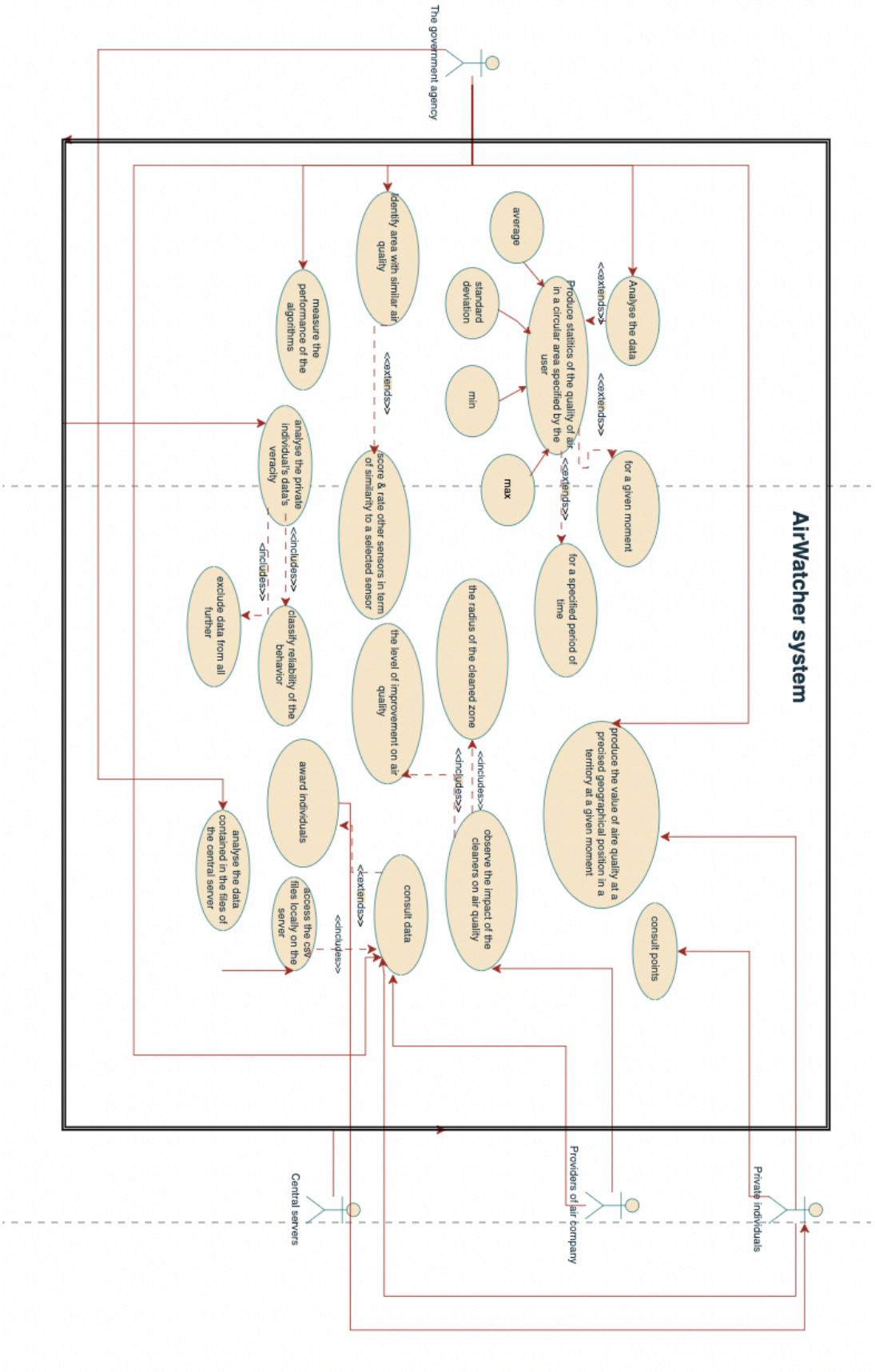
Project: AirWatcher

Team: INSA Lyon – 3IF – Group 3322 x Group ?

Version: 1.0

Last updated: April 2025

Contributors: Ines Chebbi, Tom Sanchez, Yassine Tahani, Omar Tkito, Asmae Touiti



USE CASE DIAGRAM : System AirWatcher

Actors identified through the explanatory text of the system :

-Government agency
- Particular individual
-Air cleaner provider
-Central server

A FAIRE:
-Rajouter un menu de connexion si particulier (check points)