INSA de Lyon
Computer Sciences Department
Software Engineering & UML

GL-UML Lab
AirWatcher

Autors:
Melisse Cochet
Saad ElGhissassi
Jassir Habba
Sekyu Lee
Simon Perret

Professors:
Mrs. Laforest
Mr. Hasan

April 5, 2024

Contents

Introduction

In this lab series, we embark on the development of 'AirWatcher', an advanced software application commissioned by a government agency for environmental protection. The application is pivotal in analyzing data from a comprehensive network of air quality sensors, ensuring meticulous surveillance over an extensive territory. AirWatcher's capabilities extend beyond mere data collection; it is designed to identify and troubleshoot malfunctioning sensors, synthesize data into actionable insights, and deliver real-time air quality evaluations.

Central to AirWatcher's operation is the government agency, the system's primary user, charged with the critical task of regional air quality oversight. The agency leverages AirWatcher for continuous monitoring, sensor management, and detailed analytical reporting to inform policy-making and public engagement. Furthermore, the system plays a strategic role in assessing the influence of air purifiers provided by third-party companies, gauging their efficacy in improving environmental conditions.

Providers of these air purifiers, integral users of AirWatcher, utilize the software to track the performance of their products, manage operational data, and make informed decisions based on environmental feedback. These collaborations enrich the agency's understanding of intervention impacts and contribute to a cohesive environmental management strategy.

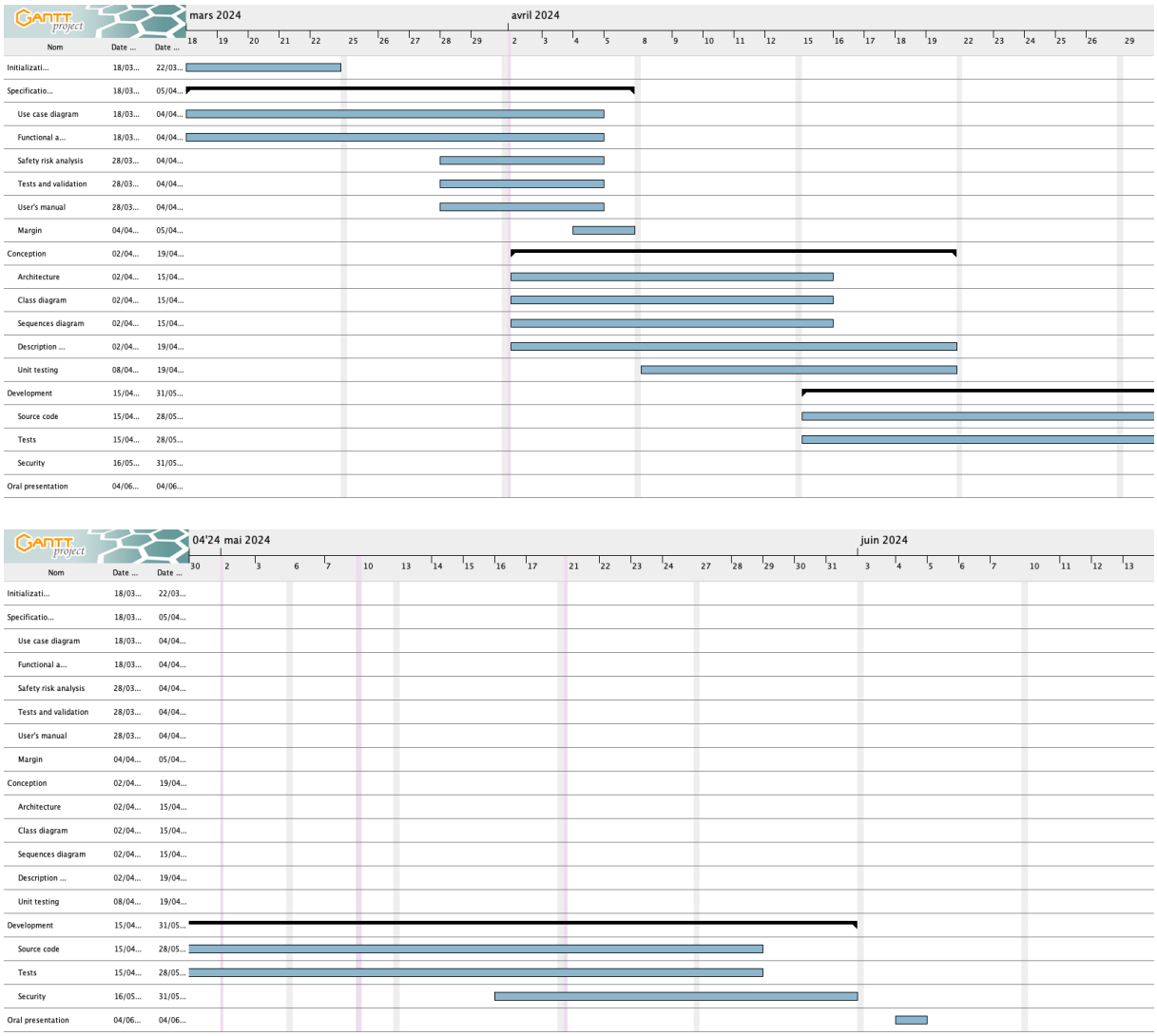
The application also empowers private individuals, volunteer contributors to the sensor network, to share their localized air quality data. Their engagement is facilitated by a points system that rewards contributions, providing a personal stake in environmental health. In return, they receive insights into local air quality and feedback on sensor reliability, fostering a community-driven approach to environmental monitoring.

By incorporating stringent security measures, AirWatcher ensures data integrity is paramount, allowing for the isolation and correction of unreliable inputs. The application is underpinned by a commitment to performance, with algorithms optimized for swift and precise execution, thus enhancing the agency's responsiveness to environmental changes.

AirWatcher stands as a beacon of environmental stewardship, a testament to the synergy between software engineering excellence and ecological conservation. The ensuing documentation maps out the development journey, detailing our structured approach to fulfilling the agency's ambitious vision for a comprehensive air quality monitoring system.

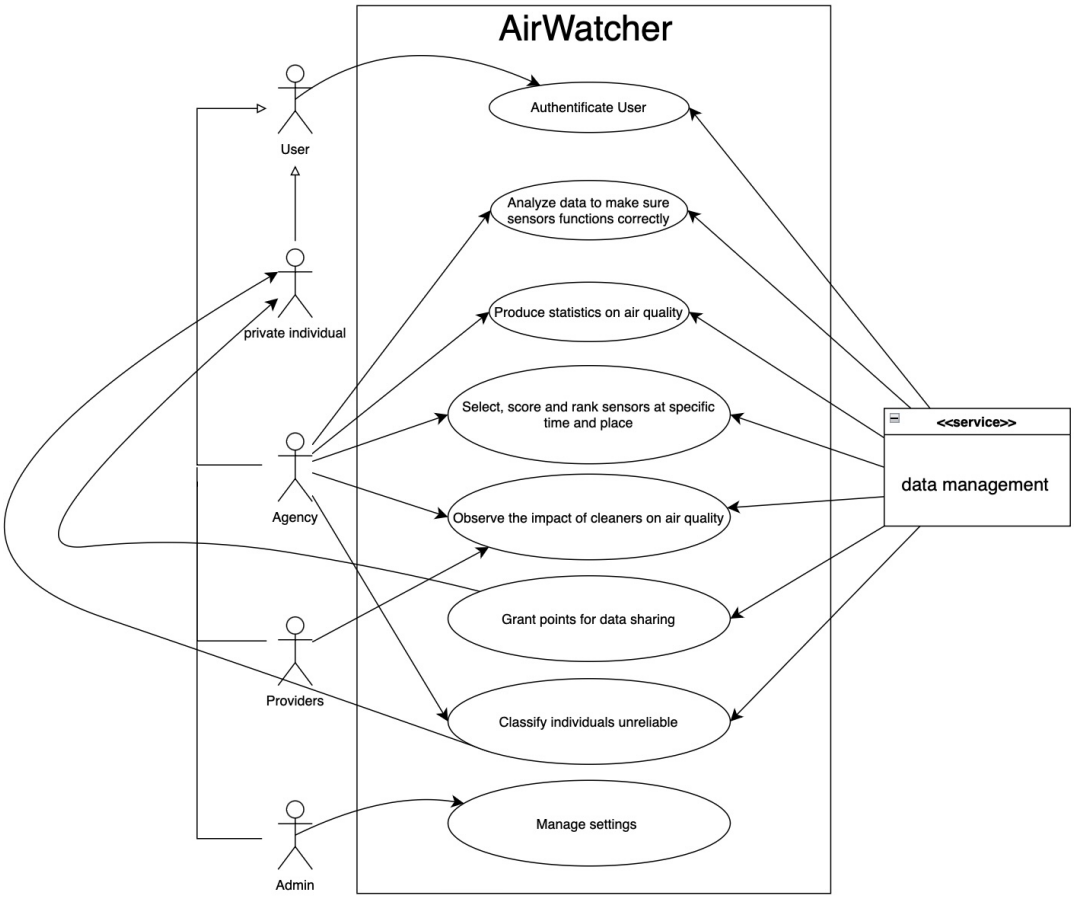
1 Requirements specification

1.1 Gantt diagram



Name	begin	end
Initialization documents	18/03/2024	22/03/2024
Specification of requirements	18/03/2024	05/04/2024
Use case diagram	18/03/2024	04/04/2024
Functional and non-functional requirements	18/03/2024	04/04/2024
Safety risk analysis	28/03/2024	04/04/2024
Tests and validation	28/03/2024	04/04/2024
User's manual	28/03/2024	04/04/2024
Conception	02/04/2024	19/04/2024
Architecture	02/04/2024	15/04/2024
Class diagram	02/04/2024	15/04/2024
Sequences diagram	02/04/2024	15/04/2024
Description and pseudo-code	02/04/2024	19/04/2024
Unit testing	08/04/2024	19/04/2024
Development	15/04/2024	31/05/2024
Source code	15/04/2024	28/05/2024
Tests	15/04/2024	28/05/2024
Security	16/05/2024	28/05/2024
Oral presentation	04/06/2024	04/06/2024

1.2 Use case diagram



1.3 Functional/non-functional requirements and test

1.3.1 Functional requirements

Sensor Functionality Analysis

Inputs Concentrations of O3, SO2, NO2, PM10 from measurements.csv; sensor coordinates from sensors.csv.

Outputs List of malfunctioning sensors

Process Compares sensor data against the five geographically closest sensors, accepting a maximum deviation of 10% per indicator.

Tests
- Detect a sensor providing one incorrect concentration.
- Detect a sensor providing entirely incorrect concentrations.

Air Quality Statistics Production

Objective Generate air quality statistics for a specified time and location.

Inputs Time (moment or period), Location (coordinates), Sensor data (measurements.csv), Optional radius (default 10 km).

Outputs ATMO index value, air quality level for the area.

Process Calculates average concentrations to measure air quality based on the ATMO index.

Tests
- Verify the lowest quality level is displayed when measurements are between quality levels.

- Expand search radius automatically if no sensors are found in the initial area.
- Notify the user if no measurements are available for the specified time.

Sensor Similarity Scoring and Ranking

Objective Score and rank sensors based on similarity to a selected sensor during a specified time.

Inputs Sensor ID (user-selected), Time period (user-selected).

Outputs Selected sensor's score, top 10 similar sensors with scores and distances.

Process Displays sensors with the closest air quality readings, within a 10% similarity tolerance, ranking further sensors higher in case of equal scores.

Tests
- Handle fewer than 10 sensors within the similarity score threshold.
- Address identical scores and distance ranking.

Air Cleaner Impact Observation

Objective Assess the impact of air cleaners on air quality.

Inputs Cleaner data (cleaners.csv), Sensor locations (sensors.csv).

Outputs The furthest sensor within the impact zone, improvement levels in percentages.

Process Compares data from the three closest sensors during and outside cleaner operation to calculate improvement.

Tests
- Identify cleaners with no measurable impact on air quality.
- Provide feedback when data is insufficient for assessment.

Data Sharing Points Allocation

Objective Reward private individuals for sharing sensor data.

Inputs User information (users.csv), Data usage instances.

Outputs Points awarded, total points per user.

Process Tracks sensor data usage and allocates 10 points per instance to the contributing user.

Tests
- Verify functionality with varied data usage scenarios.
- Ensure no points are allocated for non-private sensors.

Individual Reliability Classification

Objective Determine the reliability of data from private sensors.

Inputs User and sensor IDs (users.csv), Sensor data (measurements.csv).

Outputs Updated reliability status in measurements.csv and users.csv.

Process Compares user sensor data against neighboring sensors to detect anomalies beyond a 20% tolerance.

Tests
- Validate with both reliable and unreliable sensor data.
- Ensure user reliability status is accurately updated and communicated.

Dataset Context

- Total Sensors: 100 sensors, mapped across France.
- Measurements: Daily readings of O3, SO2, NO2, PM10, taken simultaneously.
- User Types: Two private users, one honest, one dishonest.
- Air Cleaners: Two, with differing effectiveness.
- Task: Identify dishonest users, effective air cleaners, and impact zones through AirWatcher functionalities.

1.3.2 Non-functional requirements

Performance Testing

Requirement : The application must process transactions within 2 seconds (actually, we might review this value) under normal operational conditions.

Testing method: Implementing load testing to simulate a large number of users accessing the application simultaneously to ensure that the application maintains its performance benchmarks, such as processing speed and response times. The testing will cover various scenarios, including normal, peak, and exceptional load conditions to assess the application's behavior under stress.

Reliability Testing

Requirement : The application should demonstrate a high degree of reliability, with a defined uptime guarantee.

Testing Method: Utilizing uptime tracking software to continuously monitor the availability of the application. Automated testing scripts can be employed to regularly send requests to the system to ensure it's operational. Additionally, reliability can be measured by the Mean Time Between Failures (MTBF) where the system is observed over time to identify the average interval between failures.

Security Testing

Requirement : AirWatcher must adhere to stringent security protocols to protect against unauthorized access and ensure data integrity. Users must authenticate through a secure login process, potentially integrating with existing agency credentials.

Testing Method: Conducting security audits (c.f : Analysis of Security Risks). Regularly update and patch the system in response to newly discovered vulnerabilities.

Usability Testing

Requirement : The system should be user-friendly, allowing users to perform functions with minimal training and effort. The console-based interface must be intuitive, requiring no more than 1 hour of training for new users to perform basic functions.

Testing Method: Conduct user experience (UX) testing sessions where participants perform typical tasks using the system. These sessions will be monitored to identify any usability issues.

Compliance Testing

Requirement : he application must comply with all relevant environmental and data protection legislation.

Testing Method: Conducting compliance audits against the latest standards and regulations. This involves reviewing the system's data handling, storage, and processing against legislative requirements and ensure that all personal data is managed according to privacy laws such as GDPR.

Portability Testing

Requirement : The application should be easily portable to different systems with minimal adjustments. AirWatcher should be adaptable for potential deployment on different server platforms without requiring significant changes.

Testing Method: Testing the application on different operating systems and hardware configurations to assess its adaptability.

Organizational Compliance Testing

Requirement : The development process should align with organizational standards and guidelines.

Testing Method: Implementing code reviews and documentation inspections to ensure compliance with prescribed practices.

1.4 Safety risk analysis

System : AirWatcher					
Asset	**Vulnerability**	**Attack**	**Risk**	**Impact level**	**Countermeasures**
Recovered air condition data	Inadequate password policies	Brute-force attack	Unauthorized account access	High	Enforce strong password policies and regular rotations
Data transfer to databases	Lack of secure transmission	Man-in-the-middle attack	Data interception and manipulation	High	Implement SSL/TLS encryption and digital certificates
Sensor and user data in the database	Insufficient access controls	Unauthorized access	Data leakage or unauthorized modification	Medium	Apply principle of least privilege, encrypt data at rest, and monitor for unusual access patterns
System operability	Single point of failure (server)	Power outage	System downtime and potential data loss	High	Deploy backup power solutions and redundant systems
Sensor data integrity	Malicious data tampering by users	Data corruption	Ingestion of false data	High	Implement anomaly detection systems and user behavior analysis
Unencrypted data	Unsecured data storage and transmission	System intrusion	Uncompensated data exposure	Medium	Utilize end-to-end encryption for data transmission and storage
Input data validation	Lack of input sanitization	Injection attacks	Storage and processing of erroneous data	High	Enforce strict input validation and sanitization processes

High
- Very costly loss of major tangible assets or resources
- Significant violation of, or harm or impediment to, an organization's mission, reputation, or interest

Medium
- Costly loss of tangible assets or resources
- Violation of, or harm or impediment to, an organization's mission, reputation, or interest

Low
- Loss of some tangible assets or resources
- A noticeable effect on an organization's mission, reputation, or interest

1.5 User's manual

1.5.1 Introduction

Welcome to the AirWatcher console interface, where air quality data is analyzed, and sensor information is managed. This manual will guide you through using AirWatcher based on your role - whether you're part of the Agency, a Provider, or a Private Individual.

Getting started

1.5.2 Installation and Compilation

Before launching the AirWatcher application, ensure you have compiled the source code using the provided Makefile. Follow these steps :

- Open the Terminal. Navigate to your terminal or command prompt application on your system.
- Navigate to the Application Directory. Use the cd command to change the directory to the location where the AirWatcher source code is stored. For example:

  ```
  cd path/to/AirWatcher
  ```

- Compile the Application. Run the make command to compile the application:

  ```
  make
  ```

  This command will use the Makefile to compile the source code into an executable.

- Run the Application. Once the compilation is successful, start the AirWatcher application by running:

  ```
  ./AirWatcher
  ```

1.5.3 Role Declaration

Upon starting AirWatcher, you will first declare your user status. Please enter your role when prompted:

```
Enter your role (Agency/Provider/Private Individual): [YourRole]
```

1.5.4 Authentication

After declaring your role, you will be prompted to authenticate:

```
Username: [YourUsername]
Password: [YourPassword]
```

1.5.5 Role-Based Main Menu

Once authenticated, you will access a main menu tailored to your role with functionalities specific to your privileges and responsibilities.

```
AGENCY MENU
Analyze Sensor Data
View Air Quality Statistics
Observe Cleaner Impact
Manage User Points and Classifications
System Settings
Log Out
```

```
PROVIDER MENU
View Cleaner Performance
Update Cleaner Information
View Air Quality Reports
Log Out
```

```
PRIVATE INDIVIDUAL MENU
View Personal Sensor Data
Check Points and Rewards Status
Log Out
```

Select the number corresponding to the action you want to take and press Enter.

1.5.6 Navigating the Interface

Returning to the Previous Menu
To return to the previous menu at any time, type:

```
back
```

Clearing the Screen
To clear the console screen of all previous commands and outputs, type:

```
clear
```

Viewing Help
For a list of available commands or help with a specific command, type:

```
help
```

Using Your Role-Specific Functions
After selecting an option from the main menu, simply follow the on-screen prompts to execute the desired function.

Exiting the Application
To exit AirWatcher safely, select the 'Log Out' option from your main menu.

Support and Assistance
For additional help or to report issues, contact the AirWatcher support team at support@airwatcher.gov.