Requirement Analysis and Specification Document

PRESENTATION OF AIR POLLUTION DATA USING AN INTERACTIVE WEB MAP

Authors
Ahmed Abdalgader Ahmed Eisaa
Alba Lunner
Evalyn Horemans
Leonard Hökby
Mostafa Mahmoud

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Ahmed Abdalgader Ahmed Eisaa, Alba Lunner, Evalyn Horemans, Leonard Hökby, Mostafa Mahmoud
Version:	2.0
Date:	Date: June 6, 2022
Copyright:	Copyright © 2022, A.A.E.L.M – All rights reserved

Revision history

Version	**Date**	**Change**
version 2.0	6th of June, 2022	Second submitted version

Table of Contents

Introduction

Purpose

In a time of information saturation and global climate crisis, there is a need for distributed accessible high-quality information. This, combined with the increasing propensity for the individual to find their own specific information sources tailored to their own needs, means that tools are needed which allow individuals to cross-reference the information they receive using reliable and objective sources. Such tools would allow them to substantiate concerns in negotiations with governing bodies and other stakeholders as agents of democracy. The goal of our web application is to act as one of these tools, specifically concerning access to air quality measurements in locations across Italy.

Scope

The application to be developed, therefore, aims to be a personal tool that provides easy access to air pollution data in order to empower citizens both in their personal lives and as members of a democratic community. The application is not a source in and of itself, but rather a visualization tool for existing and established data sources. The first section takes a closer look at who the intended users are, followed by scenarios in which a user might turn to the application. Going into more depth, the phenomena around the application and its use cases are explained along with hard requirements to which the app will be developed.

The use cases and functionalities discussed here apply to the first version of the application. It should be noted that for the second version, to be released in June 2022, new elements such as a login feature are planned.

Definitions

“The web app” or “app” - The application that is being developed
“Use case” - An example of user interaction with the system
“User” - A person using the website that satisfies the description under the “ user characteristics” section.
“Bookmark” - Saving a snapshot of the information presented at a location on the map
JSON - JavaScript Object Notation
JS - javascript
df - Dataframe, a pandas dataframe object
gdf - geodataframe, a geopandas dataframe object
RASD - Requirements Analysis and Specification Document
easy later access.
“OpenAQ” - An organization providing open air quality data through API (OpenAQ, n.d)
“API” - Application Programming Interface

User Characteristics

The target user profile is a non-expert and may be an individual user or a user community. The user(s) want to quantify the health risk of air pollution specific to one geographic location or a series of geographic locations, either statically at a point in time or as the evolution of this risk over a period of time. This user lacks the technical skills to extract and present the data, and may have minimal knowledge of the concepts. The user(s) would like to apply their findings to a case that is personal or local in nature, allowing for a user community to support their work and keep track of changes occurring in the data.
Sufficiently precise solutions are not readily available to support their projects, and the user needs access to a new solution which will be relevant to their project.

Analysis of Phenomena

Phenomena	World/Shared/Machine	Controlled by:
Some important weather/climate event happens in the real world	World	World
Some anthropocentric event happens in the real world (discrete)	World	World
Some anthropocentric trend is observable	World	World

New data (open source) is uploaded to the source data	World	World
Someone is interested in knowing air pollution values for a time and location	World	World
Someone executes a search with parameters via the web app	Shared	World
The web app queries the OpenAQ API using input variables	Shared	Machine
The web app returns values for a set of parameters via the map view or dashboard view.	Shared	Machine
Someone locates a point on the base map using the web app interface	Shared	World

Scenarios

Scenario 1 – Spatial query empowering behaviour modifications

An individual user wants to contextualize air pollution of a given city compared to surrounding areas, other specific urban areas, or a reference value.

For example, identifying areas for outdoors activity or travel destinations with lower air pollution. The desired impact of this is empowering users to make decisions and behavioural changes.

Scenario 2 – Temporal query tracking an event

An individual user or user community wants to track air pollution levels for a given area across time using historical data.

For example, the evolution of air quality in a wildfire event spanning pre-wildfire conditions, the impact of the fire, and subsequent dissipation of particulates. The desired impact is increased health and safety as a result of distributed decision-making, i.e. local stake-holders acting on behalf of their own communities.

Scenario 3 – Integrating trends with local initiatives

A representative of a user community wants to identify trends to spread information on positive or negative air quality phenomena.

For example, a member of the media informing constituents and mobilizing policy makers. In its spatial form, the desired impact may be increased understanding of air quality levels and their importance for the local population. In the temporal form, the desired impact may be observing improvements following investments.

Use Cases

Participating actors – Individual user, Interactive web app, OpenAQ API
Entry condition – Always True; The web app can always query the air pollution values for a geographic location in OpenAQ API.

U1: Search for location

Flow of events

1.1 The user enters the start point, a default orientation of the app (centered on a default point location).
1.2 The user selects a geographic location to the app using the standard city name.
1.3 The Interactive web app sends a request for data to the OpenAQ API to acquire the correct information.
1.4 The Interactive web map receives the data and the coordinates are used to place a marker on the map.
1.5 The user is able to see the searched location on the Interactive web app as a marker.

Exit condition: The user sees the updated map window with their selected city.

Exception handling: No city will appear in the dropdown that corresponds to wrong inputs.

Special requirement: The search string will be correctly spelled and in English. The city has to be a city with air pollution data from OpenAQ in Italy.

U2: Get historical data on specific air pollution particles

Flow of events -
2.1 The user clicks on a marker marking a location on a map on the Interactive web app.
2.2 The program requests data from the OpenAQ API.
2.3 The OpenAQ API sends queried data to the program.
2.4 Data processing is performed to send the correct data to the Interactive web app.
2.5 The user switches from map view to Dashboard view to see information on historical data.
2.6 The user selects one particle of interest on the dashboard graph showing historical air pollution information.

Exit condition: The user sees the updated dashboard window with the latest data on the selected particle.

Exception handling: It will only be possible to choose from predefined air pollution particles.

Special requirement: The city must be a city with air pollution data from OpenAQ in Italy.

U3: Change time interval for air pollution data

Flow of events -
3.1 The user clicks on a marker marking a location on a map on the Interactive web app.

3.2 The program requests data from the OpenAQ API.
3.3 The OpenAQ API sends quired data to the program.
3.4 Data processing is performed to send correct time intervals to the Interactive web app.
3.5 The user switches from map view to Dashboard view to see information on historical data.
3.6 The user selects one of the predefined time intervals of interest on the dashboard graph.

Exit condition: The user sees the updated dashboard window with their selected time interval.

Exception handling: It will not be possible to select any time interval, only predefined time intervals if one year and one month.

Special requirement: The city must be a city with air pollution data from OpenAQ in Italy.

U4: Compare current and historical data from two locations
Flow of events -
4.1 User selects location 1 on Interactive web app
4.2 The program requests data from the OpenAQ API.
4.3 The OpenAQ API sends queried data to the program.
4.4 User can expand the popup for the selected location to view current data for location 1.
4.5 User switches to Dashboard view and sees historical data for location 1.
4.6 User selects location 2 on Interactive web app, and steps 4.2 - 4.5 are repeated for location 2.

Exit condition: The user sees the updated dashboard window with their second selected location.

Exception handling: It will not be possible to see data from two locations at the same dashboard view.

Special requirement: Both locations must be cities with air pollution data from OpenAQ in Italy.

U5: Register as a User
Flow of events -
5.1 The user clicks on the user profile icon to open the dropdown showing account actions.
5.2 The user selects “Register” from the dropdown.
5.3 A popup appears.
5.4 The user can fill the dialog box using the fields “Username” and “Password”. Each input box is highlighted when selected by the user. The password input is masked.
5.5 The user submits their input using the “Register” button in the popup.
5.6 The user input is sent to the database for storage. The password is hashed.
5.7 The user is prompted to login with their new credentials.

Exit condition: The user sees the updated views (map/dashboard) “logged in as”, with their username

Exception handling: If an input field is left empty, the labels “Username” or “Password” will indicate a required field by changing to “Username (needed)”/” Password (needed)” accompanied by a colour change. If the user already has an account, a message will be provided to indicate non unique input and the user can navigate to the correct dialogue form using the clickable support “Already have an account? Login here!”

Special requirement: The user cannot have an existing account under the same credentials.

U6: Login as a Registered User
Flow of events -
6.1 The user clicks on the user profile icon to open the dropdown showing account actions.
6.2 The user selects "Login" from the dropdown.
6.3 A popup appears.
6.4 The user can fill the dialog box using the fields "Username" and "Password". Each input box is highlighted when selected by the user. The password input is masked.
6.5 The user submits their input using the "Login" button in the popup.
6.6 The user input is sent to the database to compare against existing user records.
6.7 The database confirms if the user exists.
6.8 The database confirms if the username and password are a match.

Exit condition: The user sees the updated views (map/dashboard) "logged in as", with their username

Exception handling: If an input field is left empty, the labels "Username" or "Password" will indicate a required field by changing to "Username (needed)"/"Password (needed)" accompanied by a colour change. If the user does not exist, a message will be provided to indicate incorrect input and the user can navigate to the correct dialogue form using the clickable support "Already have an account? Login here!". If the password is not a match, a message will be provided to indicate incorrect input.

Special requirement: The user must have an existing account under the same credentials.

U7: Log Out
Flow of events -
7.1 The user clicks on the user profile icon to open the dropdown showing account actions.
7.2 The user selects "Login" from the dropdown.
7.3 A popup appears.
7.4 The user can fill the dialog box using the fields "Username" and "Password". Each input box is highlighted when selected by the user. The password input is masked.
7.5 The user submits their input using the "Login" button in the popup.
7.6 The user input is sent to the database to compare against existing user records.
7.7 The database confirms if the user exists.
7.8 The database confirms if the username and password are a match.

Exit condition: The user sees the updated views (map/dashboard) "logged in as", with their username

Exception handling: If an input field is left empty, the labels "Username" or "Password" will indicate a required field by changing to "Username (needed)"/"Password (needed)" accompanied by a colour change. If the user does not exist, a message will be provided to indicate incorrect input and the user can navigate to the correct dialogue form using the clickable support "Already have an account? Login here!". If the password is not a match, a message will be provided to indicate incorrect input.

Special requirement: The user must have an existing account under the same credentials.

Requirements

The product is a website proof-of-concept (poc), from now on called "web app" that should meet the following requirements:

Functional requirements

R1 Map
R1.1 The web app should present an interactive map which the user can zoom and pan. This map should contain both air-pollution data and a relevant background map.
R1.2 There should be a dropdown function that can find included cities based on their names.

R2 Dashboard
R2.1 The web app should present an interactive dashboard with which the user can interact. This dashboard should contain air-pollution data presented in different time intervals.
R2.2 There should be a dropdown function that can find included cities based on their names.

R3 Data
R3.1 The app should allow for comparison of data across time and space.
R3.2 Current data should refer to the most recent entry in the dataset.
R3.3 The data classification levels should be customizable.

R4 Login/Logout
R4.1 The app should allow user login based on username and password.
R4.2 The app should allow new users to register.
R4.2 The app should allow users to log out.

R5 Non-functional requirements
R5.1 The data should be displayed in a manner that is clear and easy to understand

R6 Technological requirements
R6.1 The web app should implement requests to a REST API in order to access the data used.
R6.2 The main backend implementation language must be Python.
R6.3 Flask is one of the technologies to be used in the stack

Domain assumptions

Users have a level of computer literacy which allows for comfortable navigation of standard browsers (i.e. Google and related functions such as Google maps). Familiarity with dropdown menus, zooming, and panning can be assumed.

It is assumed that the API providing the air pollution data is available 24/7 and that the data within the API is correct.

It is assumed that the user accesses the web app via a device powerful enough to support client-side rendering of the app and has an internet connection that is fast enough to support server-side rendering of the app.

References

D2.1 requirement specification and analysis - europa. (n.d.). Retrieved April 20, 2022, from https://cordis.europa.eu/docs/projects/cnect/3/609023/080/deliverables/001-MYWAYD21RequirementSpecificationAndAnalysiscnectddg1h520142537023.pdf

Environmental Protection Agency. (n.d.). EPA. Retrieved April 20, 2022, from https://www.epa.gov/enviroatlas/enviroatlas-use-cases

Use cases and requirements for standardizing web maps. (n.d.). Retrieved April 20, 2022, from https://maps4html.org/HTML-Map-Element-UseCases-Requirements/#use-case-single-location

Use cases. Copernicus. (n.d.). Retrieved April 20, 2022, from https://atmosphere.copernicus.eu/use-cases