POLITECNICO
MILANO 1863

# Test Report

## PRESENTATION OF AIR POLLUTION DATA USING AN INTERACTIVE WEB MAP

Authors

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

| | |
|---|---|
| **Deliverable:** | TR |
| **Title:** | Test Report |
| **Authors:** | Ahmed Abdalgader Ahmed Eisaa, Alba Lunner, Evalyn Horemans, Leonard Hökby, Mostafa Mahmoud |
| **Version:** | 1.0 |
| **Date:** | Date: June 7, 2022 |
| **Copyright:** | Copyright © 2022, A.A.E.L.M – All rights reserved |

**Revision history**

| Version | Date | Change |
|---|---|---|
| version 1.0 | 7th of June, 2022 | First submitted version |

## Table of Contents

# Test Report

This document presents the results of the tests outlined in *Design Document and Test Plan: Presentation of Air Pollution Using an Interactive Web Map* (Version 2.0). Testing includes: unit testing (on the individual component level), integration testing (testing the interaction of the components), and system testing (testing the complete application).

## Unit Testing: Flask

Unit testing was performed in each branch in GitHub before committing, pushing, pull requests and merging. The tests were carried out according to the agile method throughout the development process. A unit test is considered successful if the code ran and returned correct output. Table 1 below describes the endpoints used in the project and how they are formatted.

Table 1: Endpoints of Open AQ used

| Open AQ endpoint | Description | Querying function | Use in flask endpoint |
|---|---|---|---|
| v2/locations | Provides a list of all locations | getLocations() | /api/locations |
| v2/latest | Provides the latest measurements from all locations | getLatest()<br>getCity(city) | /api/latest<br>/api/cities/<city> |
| v2/measurements | Provides a list of measurements | queryByDay(startDay, endDay, city) | /api/month/<city><br>/api/year/<city> |
| v2/cities | Provides a list of cities within the plattform | getCityNames() | /api/cities/[1] |

[1] The data from getCityNames is used to check the input for all endpoints that accept city as input, so in fact, it is used also for those three endpoints.

## Tests for Endpoints without Inputs

Table 2: Test for endpoints without inputs

| Endpoint | Test | Hypothesis on system: | Expected outcome | Initial state of System | Actual output: | Pass/Fail |
|---|---|---|---|---|---|---|
| api/locations | Send request through browser | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| api/latest | Send request through browser | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| api/cities | Send request through browser | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| api/anythingelse | Send request through browser | Flask server turned on and listening to localhost:5000 | 404 not found error | Flask server turned on and listening to localhost:5000 | 404 not found error | Pass |

## Tests for Endpoints with City as Input

Table 3: Test for Endpoints with City as input

| Endpoint | Test | Hypothesis on system: | Expected outcome | Initial state of System: | Actual output: | Pass/Fail |
|---|---|---|---|---|---|---|

|  |  |  |  |  |  |  |
|---|---|---|---|---|---|---|
| */<city> | Send request through browser with existing city | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| */<City> | Send request through browser | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| */<CITY> | Send request through browser | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Flask server turned on and listening to localhost:5000 | JSON string matching format in table 1 returned | Pass |
| */<notacity> | Send request through browser | Flask server turned on and listening to localhost:5000 | 400 bad request error, “city notacity doesn’t exist in database” | Flask server turned on and listening to localhost:5000 | 400 bad request error, “city notacity doesn’t exist in database” | Pass |

## Unit Tests for POST endpoints with Payload

Table 4: Test for POST endpoints with payload

| Endpoint | Payload | Test | Hypothesis on system: | Expected outcome | Initial state of System: | Actual output: | Pass/Fail |
|---|---|---|---|---|---|---|---|

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| /api/register | `{“username”: <String>, “password”: <String>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. The username does not exist in database | JSON string matching format in table 1 returned, new user registered in database | Viewing register popup. The username does not exist in the database | JSON string matching format in table 1 returned, new user registered in the database | Pass |
| /api/register | `{“username”: <String>, “password”: <String>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. The username already exist in database | JSON string matching format in table 1 returned, no change in database | Viewing login/register popup. The username already exists in database | JSON string matching format in table 1 returned, no change in database. Message provided to the user. | Pass |
| /api/register | `{“anything”: <Whatever>, “other”: <Something>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. | 400 bad request error. | Viewing login/register popup. Input is not valid | 400 bad request error. Message provided to the user. | Pass |
| /api/authent-icate | `{“username”: <String>, “password”: <String>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. Username and password is correct | JSON string matching format in table 1 returned | Viewing login pop-up. | JSON string matching format in table 1 returned. User is logged in and the username is displayed in the top right. | Pass |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| /api/authent-icate | `{"username": <String>, "password": <String>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. Username or password is incorrect | JSON string matching format in table 1 returned | Viewing login pop-up. | JSON string matching format in table 1 returned. User not logged in. Message provided to the user. | Pass |
| /api/authent-icate | `{"anything": <Whatever>, "other": <Something>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000 | 400 bad request error. | No authenticated user | 400 bad request error. | Pass |
| /api/logout | `{"username": <String>, "lastsearch": <String>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. Username and lastsearch is correct | JSON string matching format in table 2 returned. "lastsearch" added at user row in database | User is logged in and has searched for at least one city. | JSON string matching format in table 2 returned. "lastsearch" added at user row in database | Pass |
| /api/logout | `{"anything": <Whatever>, "other": <Something>}` | Send request from Nuxt app | Flask server turned on and listening to localhost:5000. | 400 bad request error. | No logged in user | 400 bad request error. | Pass |

## Unit Testing: Nuxt

To check the units in Nuxt, the input data listed in the console was checked. Then, the representation was checked to make sure that the correct data is being displayed. Finally, a test of the dynamicity and interactivity was carried out to check if all possible different inputs were correctly intertwined with the app. Inputs tested included various cities and particles. All Nuxt Unit tests were passed.

## Integration Tests

Integration tests were done continuously throughout the software development process. The software was divided into smaller units, branches, to be developed independently. The integration test was identical for all units. A finished piece of software was integrated to the main software with GitHub. Integration was made by committing changes in the code, then pushing them to the origin to make the code available for other group members. After pushing, a pull request was made so another group member could double check the code and approve it for merging. The finished piece of code was then merged into the main software. Integration testing was carried out by checking for conflicts and then running the main piece of software. An integration test was considered successful if no conflicts were present and the code could run without error.

## System Testing

System testing was carried out by running the software and checking on the web app interface that the implementations had correct functionality. Testing was done to make sure that all functionalities described by the use cases and requirements were fulfilled. System testing was integrated in Integration testing, the tests were done at the same occasion. Use cases affected by each function are noted in the following table, and a summary of use cases and their codes is seen below:

U1: Search for location
U2: Get current data on specific air pollution particles
U3: Change time interval for air pollution data
U4: Compare current data from two locations
U5: Register as a User
U6: Login as a Registered User
U7: Log Out
One condition for doing these tests is that the backend and frontend is turned on. Meaning that the Flask server is turned on and listening to localhost:5000 and the Nuxt app turned on and listening to localhost:3000.

Table 5: Integration and system tests on functions that are considered finished

| Function and Use Case | Test | Hypothesis on system | Expected outcome | Initial state of System: | Actual output: | Pass/Fail |
|---|---|---|---|---|---|---|
| Map U1 U2 | Navigate to the map page, refresh the page. | Set-up complete[2] | Map displays with markers | Flask server turned on and listening to localhost:5000. Nuxt app turned on and listening to localhost:3000. | Map displays with markers | Pass |
| Pop-up U1 U2 U3 | Navigate to the map page and click a number of markers | Set-up complete | Map displays with markers, when markers are clicked, they produce a pop-up with correct information | Backend and frontend is turned on. Map displayed with clickable markers. | Map displays with markers, when markers are clicked, they produce a pop-up with correct information | Pass |
| Map dropdown list U1 U4 | Navigate to the map page, choose a city from dropdown list | Set-up complete | Map displays, dropdown displays. Clicking a city in the dropdown causes the map to zoom to that city. | Backend and frontend is turned on. Collapsed dropdown is displayed and clickable. | Map displays, dropdown displays. Clicking a city in the dropdown causes the map to zoom to that city. | Pass |
| Contacts page | Navigate to the contacts page | Set-up complete | User is shown a list of the project members and their information | Backend and frontend is turned on. Contacts page accessible via | User is shown a list of the project members and their information | Pass |

[2] Flask and Nuxt servers are running, database has been initialized

| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | sidebar and address. | | |
| Dashboard U1 U3 U4 | Navigate to the dashboard page and click on different parameters for the graphs | Set-up complete | Dashboard displays with two graphs. Graphs change when a parameter is clicked. | Backend and frontend is turned on. Dashboard with graphs is displayed, including dropdown to choose city and toggle to choose parameters. | Dashboard displays with two graphs. Graphs change when a parameter is clicked. | Pass |
| Dashboard dropdown list U1 U4 | Navigate to the dashboard and choose a city from the dropdown list. | Set-up complete | Dashboard displays as above, and data for a selected city are shown. | Backend and frontend is turned on. Collapsed dropdown is displayed and clickable. | Dashboard displays as above, and data for a selected city are shown. | Pass |
| Login U6 | Click the “User” icon on the top right corner and open the login dialog. Fill out username and password in the dialog | Set-up complete. User is not logged in Username and password are correct. | User is logged in and the username is displayed in the top right. | Backend and frontend is turned on. “User” icon is clickable and displays a pop-up window when clicked. | User is logged in and the username is displayed in the top right. | Pass |
| Login U6 | Open the login dialog. Fill out username and password in the dialog | Set-up complete. User is not logged in. Username or password are not correct. | User is presented with a “Authentication failed” message. | Backend and frontend is turned on. Login dialog is open and input fields are displayed. | User is presented with a “Authentication failed” message. | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| Login U6 | Open the login dialog. Fill either username or password in the dialog | Set-up complete. User is not logged in | User is presented with a “Both username and password are required” message. | Backend and frontend is turned on. Login dialog is open and input fields are displayed. | User is presented with a “Both username and password are required” message. | Pass |
| Login to register and back U5 U6 | Open the login dialog. Click the link to register. Click the link to login | Set-up complete. User is not logged in | User is moved to the register dialog and then back to the login dialog. | Backend and frontend is turned on. Login dialog is open and interactive elements are displayed. | User is moved to the register dialog and then back to the login dialog. | Pass |
| Register U5 | Click the “person” icon on the top right corner and open the register dialog. Fill out username and password | Set-up complete. User is not logged in. Username is not in the system | User is created in the database and user is presented with a “account creation successful” message | Backend and frontend is turned on. “Person” icon is clickable and register pop-up displays when clicked. | User is created in the database and user is presented with a “account creation successful” message | Pass |
| Register U5 | Open the register dialog. Fill out username and password | Set-up complete. User is not logged in. Username is already in the system | User is presented with a “Username is already in use” message | Backend and frontend is turned on. “Person” icon is clickable and register pop-up displays when clicked. | User is presented with a “Username is already in use” message | Pass |
| Register U5 | Open the register dialog. Fill either | Set-up complete. User is not logged | User is presented with a “Both username and password are required” | Backend and frontend is turned on.“Person” icon is | User is presented with a “Both username and | Pass |

| | | | | | | |
|---|---|---|---|---|---|---|
| | username or password in the dialog | in. | message. | clickable and register pop-up displays when clicked. | password are required” message. | |
| Logout U7 | Click the “person” icon and click “logout” | Set-up complete. User is logged in. User has selected a city. | Username no longer shows in the top right and the user is presented with a “logout successful” message. Last Search is added to the database. | Backend and frontend is turned on. “Person” icon displays logout as an option when clicked. | Username no longer shows in the top right and the user is presented with a “logout successful” message. Last Search is added to the database. | Pass |
| Logout U7 | Click the “person” icon and click “logout” | Set-up complete. User is logged in. User has not selected a city. | Username no longer shows in the top right and the user is presented with a “logout successful” message. No change to database. | Backend and frontend is turned on. “Person” icon displays logout as an option when clicked. | Username no longer shows in the top right and the user is presented with a “logout successful” message. No change to database. | Pass |
| Logout login U6 U7 | Login a user, search a city in the dashboard, logout. Reload page and log in again | Set-up complete. | On the second log in, the dashboard defaults to the city chosen before. | Backend and frontend is turned on. “Person” icon is clickable, log in pop-up and the log out option is displayed. | On the second log in, the dashboard defaults to the city chosen before. | Pass |
| Error page | Navigate to a non existing page by typing http://localhost:3000/anything | No page called …/anything has been implemented | A 404 page not found is shown. | Backend and frontend is turned on. | A 404 page not found is shown. | Pass |