

Politecnico di Milano

AA 2020-2021

Geoinformatics Engineering

**Software Engineering for Geoinformatics Project**

**XXXXXXXXX**

**RASD**

Requirement Analysis and Specification Document

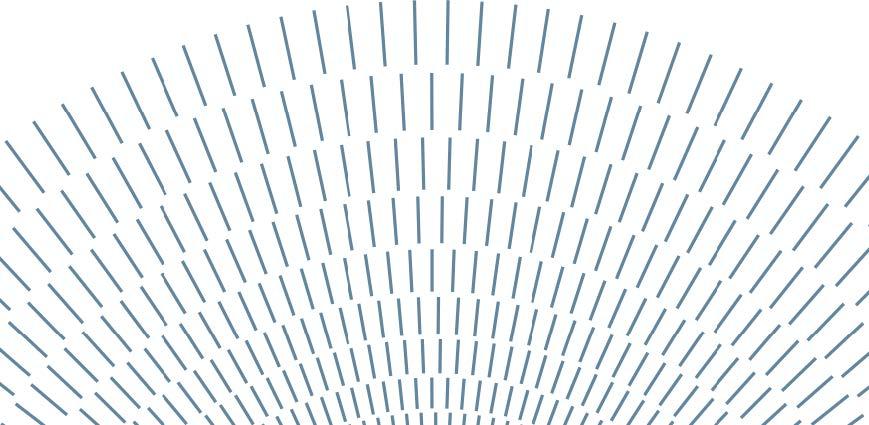
version 1.0 - 11/4/2021

By:

Gao Jiawei

Ding Guangyu

Ymeri Ernesa

Song Xiangyang

[**1 Introduction**](#_55ms5ma4r29h) **3**

[1.1 Background](#_5cqq76g1ty2t) 3

[1.2 Project Scope](#_kqqz5vgdi1fr) 3

[1.3 Project Goals](#_b1l906z66dyp) 3

[1.4 Domain Analysis](#_tjqrrw8d7soy) 4

[**2 Project Database**](#_93x9bm2klrmh) **5**

[2.1 EpiCollect5](#_7p1ojq908xld) 5

[2.2 PostgreSQL](#_ej4cbpa4sgg6) 6

[2.3 Dataset](#_w3djpgh424w9) 6

[**3 Software Structure //did not finish**](#_vy52s2ck9cm3) **7**

[**4 Use Cases**](#_lcthr1dz1sbt) **8**

[4.1 Use Cases Description](#_4v4v7sgu6wov) 8

[4.1.1 Register a user account](#_kxgg53dnu72r) 8

[4.1.2 Login](#_xlbm2lfw4u3e) 8

[4.1.3 Logout](#_mzfk9qpjwkfp) 8

[4.1.4 Query Data](#_85mfz0gnutyl) 9

[4.1.5 Update Data](#_xgkjie97jjwz) 9

[4.1.6 Delete Data](#_r6qma31nz0sv) 9

[4.1.7 View Data](#_it76k2ni6goi) 10

[4.1.8 Download Data](#_zc2r37o8nj4) 10

[4.1.9 Leave Comments](#_9xkabmcdnqmn) 10

[4.1.10 Modify Comments](#_o3flhodlam9y) 11

[4.1.11 Delete Comments](#_a32s1kwhpgm1) 11

[4.2 Use case diagrams](#_gjx0oku4rhtm) 12

[4.2.1 Visitor](#_pco6eulkk1si) 12

[4.2.2 User //did not finish](#_ytczke8zq476) 12

[**5 Efford Spent**](#_sdc2vdtfsu9e) **13**

[**6 Revision History**](#_70wwhxykb1ev) **13**

[**7 Reference**](#_e6k1iyvsaexg) **13**

This is the RASD document of group 1.

The project of our group is called “Web-Based Geospatial Database Application on Urban Environmental Investigation in San Silvestre”.

Our app is named iUrban

# 1 Introduction

## 1.1 Background

## 1.2 Project Scope

The scope of the project XXXX, accomplished by a group of Geoinformatics students in Politecnico di Milano, is to design a web-based application, serving as a platform which not only displays the geographical contents in a map-based view, but also allows users to both access or retrieve existing data on urban environmental factors (UEF in the following context) and contribute to the database of UEF in a given location through interactive mapping tools.

By using this application, users can find quantitative or qualitative data descriptions about the urban environment on the website, as well as instructions on how to contribute to the database. The permission of user-uploaded data is intended to make the application volunteer-based, so that the database can be continuously augmented. Nonetheless, the data uploaded by users must be supervised by the database managers or follow a predetermined regulation.

## 1.3 Project Goals

Firstly, the application of the project XXXX enables users to visualize location-specific UEF data on a web page based on a base map.

Secondly, users can use the UEF data to perform certain data analysis, such as filtering, buffering, etc., to inspire their decisions such as choosing the positions of their houses.

Thirdly, users can leave a comment on (...) , where the application acts as a forum for users to communicate.

Finally, users can find descriptions and guidance on how to extract the selected data arbitrarily and acquire information on how to collect and upload the data in order to keep the database up to date and enlarged gradually.

## 1.4 Domain Analysis

The web application of XXXX is aimed at providing the environmental factors in the city for house buyers. And meanwhile municipalities could apply it on the urban health or urban economy thesis, for instances, to analyze the relationship between traffic and noise and make city park planning. It could also be conducive for research on urban environments.

This web application will be run online by the users, without interaction with other software or hardware.

Generally, our product is an application serving as a platform where users could monitor, update and exchange UEF information as references to help them make decisions according to their needs. To preserve the relevance of the website, log-in is required in order to access all the functions. As web managers, we have the privilege to remove any comment or user-uploaded data.

The representation of raw data in the web application will be performed by python scripts, and the libraries needed are as following: shapely, pandas, geopandas for the representation of data; psycopg2 for SQL and also Flask for web implementation.

# 

# 2 Project Database

The data for this project, including information about point positioning, will be retrieved from EpiCollect5, pre-processed and copied to a PostgreSQL database with a given frequency. The web app will then interact with DBMS and perform CRUD operations on the PostgreSQL database. The advantages for storing the data in a PostgreSQL database as opposed to fetch them directly from Epicollect5 include: i) to enable verification, pre-processing and storing of consistent data; ii) to ensure availability of data, decoupling our web application from EpiCollect5; iii) to reduce the risk of data loss; iv) to improve performance; v) to leverage DBMS capabilities and, in particular, the interface between Python and PostgreSQL.

## 2.1 EpiCollect5

In this project, we have chosen the dataset “[SAN SILVESTRE GEOGRAPHY IA 2020](https://five.epicollect.net/project/san-silvestre-geography-ia-2020)” from the website [Epicollect5](https://five.epicollect.net/project/san-silvestre-geography-ia-2020). With 451 entries uploaded in the year of 2020, this dataset collects both the environmental and artificial geographical data in the city San Silvestre, such as temperature, humidity, wind direction noise level and traffic count. Through processing and exposing the above data, users could choose an optimal location for residential, commercial, official areas as well as public service facilities.

Here is an example of a data entry:



A data entry contains the following attributes:

1. The name of the surveyor
2. Location: geodetic coordinates (Latitude, Longitude) in WGS 84 reference system. Coordinates are expressed in degrees, generated automatically by Epicollect5.
3. Date of entry: DD/MM/YYYY
4. Time of entry: hh:mm:ss
5. Average Noise Level:
6. Average Light Intensity
7. Wind direction
8. Wind Speed (Beaufort Scale)
9. Cloud Cover:
10. Cloud Type:
11. Visibility
12. Traffic Count
13. Temperature
14. Humidity
15. Note of anything which could cause an anomaly
16. Air pollution: number of particles (PM only):

## 2.2 PostgreSQL

## 2.3 Dataset

# 

# 3 Software Structure //did not finish

# 

# 4 Use Cases

## 4.1 Use Cases Description

### 4.1.1 Register a user account

|  |  |
| --- | --- |
| Name | Register a user account |
| Actors | Visitor |
| Entry Condition | The user has opened the application through a browser |
| Flow of Events | 1. Open the application  2. Click on “Sign Up” button  3. Fill all the mandatory fields and provide the necessary information  4. Click on “Confirm” button  5. The system saves the data |
| Exit Condition | The user has successfully registered and now he’s able to use the application |
| Exceptions | 1. The information that the user is used to register have already signed up, such as e-mail and phone number  2. The user didn’t fill all of the mandatory fields with valid data |

### 4.1.2 Login

|  |  |
| --- | --- |
| Name | Login |
| Actors | User |
| Entry Condition | The user is previously successfully signed up |
| Flow of Events | 1. The user opens the application through a browser  2. The user enters her/his credentials in the “account” and “Password” fields of the home page of the app  3. The user clicks on the “Login” button  4. The user successfully login and the system automatically redirects to the function page |
| Exit Condition | The user is successfully redirected to the function page |
| Exceptions | 1. The user enters invalid Account  2. The user enters invalid Password |

### 4.1.3 Logout

|  |  |
| --- | --- |
| Name | Logout |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Logout” button  4. The user successfully logout and the system automatically redirects to the login page |
| Exit Condition | The user is successfully redirected to the login page |
| Exceptions | Some system errors |

### 4.1.4 Query Data

|  |  |
| --- | --- |
| Name | Query Data |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Query” button  2. The user inputs or chooses some filters filed  3. The user click on the “Confirm” button  4. The system returns the query results |
| Exit Condition | The system has successfully returned the results |
| Exceptions | 1. There are something wrong for user’s inputs  2. There are something system wrong |

### 4.1.5 Update Data

|  |  |
| --- | --- |
| Name | Update Data |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Update” button  2. The user inserts or edits the field data  3. The user click on the “Confirm” button  4. The system returns the update results |
| Exit Condition | The system has returned the successful message |
| Exceptions | 1. There are something wrong for user’s inputs  2. There are something system wrong |

### 4.1.6 Delete Data

|  |  |
| --- | --- |
| Name | Delete Data |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Delete” button  2. The user chooses the data  3. The user click on the “Confirm” button  4. The system returns the delete results |
| Exit Condition | The system has returned the successful message |
| Exceptions | 1. The user does not choose the data  2. There are something system wrong |

### 4.1.7 View Data

|  |  |
| --- | --- |
| Name | View Data |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “View” button  2. The user clicks on the type of view she/he wants to visualize the data, such as, Chart, Graph or Map View  3. The user clicks the specific view Button and confirms  4. The system returns the result of the view option the user selected |
| Exit Condition | The system has successfully returned the results |
| Exceptions | There are something system wrong |

### 4.1.8 Download Data

|  |  |
| --- | --- |
| Name | Download Data |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Download” button  2. The user clicks on the “Confirm” button  3. The user chooses the location to download  4. The browser starts downloading |
| Exit Condition | The system has successfully download |
| Exceptions | There are something system wrong |

### 4.1.9 Leave Comments

|  |  |
| --- | --- |
| Name | Leave Comments |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Comments” button and opens the “Comments” panel  2. The user inputs her/his comments  3. The user click on the “Confirm” button  4. The system returns the comment results and reflash this web |
| Exit Condition | The system has reflashed this page and displayed the comments |
| Exceptions | 1. The user does not input the comments  2. There are something system wrong |

### 4.1.10 Modify Comments

|  |  |
| --- | --- |
| Name | Modify Comments |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Comments” button and opens the “Comments” panel  2. The user chooses her/his comments  3. The user click on the “Modify” button  4. The user inputs and edits the data  5. The user click on the “Confirm” button  6. The system returns the modify results  7. The system reflash this web |
| Exit Condition | The system has reflashed this page and displayed the comments |
| Exceptions | 1. The user does not do the modify operation  2. There are something system wrong |

### 4.1.11 Delete Comments

|  |  |
| --- | --- |
| Name | Delete Comments |
| Actors | User |
| Entry Condition | The user has previously successfully logged in |
| Flow of Events | 1. The user clicks on the “Comments” button and opens the “Comments” panel  2. The user chooses her/his comments  3. The user click on the “Delete” button  5. The user click on the “Confirm” button  6. The system returns the delete results  7. The system reflash this web |
| Exit Condition | The system has reflashed this page and displayed the comments |
| Exceptions | 1. The user does not do the delete operation  2. There are something system wrong |

## 4.2 Use case diagrams

### 4.2.1 Visitor



### 4.2.2 User //did not finish



# 5 Efford Spent

For this project, the development team is composed of 4 people. We use the Github to manage our project. Use the Google File to edit our RASD together. Use the zoom to discuss.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Date | Task | Gao | Ding | Ernesa | Song |
| 22/3/2021 | First Discussion of this Project | 2h | 2h | 2h | 2h |
| 28/3/2021 | Scope&Goals&Domain | 3h | 3h | 3h | 3h |
| 11/4/2021 | Use Cases | 4h | 4h | 4h | 4h |
|  |  |  |  |  |  |

# 6 Revision History

|  |  |
| --- | --- |
| Date | Modifications |
| 11/4/2021 | First Version |
|  |  |

# 7 Reference