III. Test Plan

III.1. Unitary Tests

### III.1.1. UserController functions

### LoadCSV functions

These tests will verify if the files passed as parameters in the functions loadCsv are correctly loaded. Not only the files’ names have to be correct but also the data loaded. We can check if the all data are correctly loaded by comparing the size of the users list and the sum of each files’ size.

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.loadCSV(string userFile , string providerFile, string governmentFile, string passwordFile) | users providers government password | true |
| Additional Test: Verification of the obtained user list after loading the different files | userController.users.size()  (sum of all users ; individual : user.csv , providers …) | 6 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.loadCSV(string userFile , string providerFile, string governmentFile, string passwordFile) | users providers government loop | false |

### Authentification

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.authenticate(string login, string pass) | Provider0  provider0 | User object with identifier provider0 and cleaner object identifier Cleaner0 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.authenticate(string login, string pass) | Provider5  provider0 | Null |

### GetIndividualUsers

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getIndividualUsers() | void | Vector of IndividualUsers objects with identifiers : User0 and User1 |

### GetProviders

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getProviders() | void | Vector of Providers objects with identifiers : Provider0 and Provider1 |

### GetPrivilege

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getPrivilege(string identifier) | User0 | INDIVIDUAL |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getPrivilege(string identifier) | Provider0 | PROVIDER |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getPrivilege(string identifier) | Government0 | GOVERNMENT |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| userController.getPrivilege(string identifier) | User8 | null |

### III.1.2. SensorController functions

### GetSensors

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.getSensors() | void | Vector of Sensor objects with identifiers : Sensor0 and Sensor1 |

### GetSensor

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.getSensors(string identifier) | Sensor0 | Sensor object with identifiers : Sensor0, latitude: 44 and longitude : -1 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.getSensors(string identifier) | Sensor101 | null |

### Malfunctionning analysis (à modifier en function des algos)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.malfunctioningAnalysis(string sensorId) | Sensor0 |  |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. malfunctioningAnalysis(string sensorId) | Sensor101 | null |

### MeanAir Quality

For the mean air quality, we take the sensors included in the surface, and then we do a mean for the measurements according to their attribute. Finally we take each mean that we link to the corresponding Atmo index.

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. meanAirQuality(double latitude, double longitude, double radius, time\_t start, time\_t stop) | 45  -2  5  01/01/2019 12:00:00 01/01/2019 12:00:00 | 3.5  O3 : good  NO2 : good  S02 : very good  PM10 : mediocre |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. meanAirQuality(double latitude, double longitude, double radius, time\_t start, time\_t stop) | 45  -2  5  01/01/2019 12:00:00  15/15/2025 12:00:00 | null |

### Compare Sensors

To compare sensor we first compare the similarity between each measurements with the same attributes by computing a percentage. Then we do the mean of the results.

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. compareSensors(string sensorId) | Sensor0 | A map with Sensor object and a double value (similarity 0 to 1) :  Sensor1 and 0.82 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. compareSensors(string sensorId) | Sensor101 | null |

### Air Quality

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. airQuality(double latitude, double longitude, time\_t time) | 45  -2  01/01/2019 12:00:00 | 3.25  O3 : very good  NO2 : good  S02 : very good  PM10 : mediocre |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. compareSensors(string sensorId) | 45  -2  15/15/2025 12:00:00 | null |

### LoadCSV

(A changer je vois pas trop les additional tests a faire pour loadCSV de cleaner et sensor controller)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.loadCSV(string sensorFile, string measurementFile, string attributeFile) | sensors\_test measurements\_test  attributes | true |
| Additional Test: Verification of the obtained sensors list after loading the different files | sensorController.sensors.size() | 2 |
| Additional Test: Verification of the obtained measurements list after loading the different files | In a for loop :  Sum += sensorController.sensors.get(i).measurements.size() | 8 |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController.loadCSV string sensorFile, string measurementFile, string attributeFile) | airCleaners | false |

### III.1.3. CleanerController functions

### LoadCSV

(A changer je vois pas trop les additional tests a faire pour loadCSV de cleaner et sensor controller)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| cleanerController.loadCSV(string file) | cleaners | true |
| Additional Test: Verification of the obtained cleaners list after loading the different files | cleanerController.cleaners.size() | 2 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| cleanerController.loadCSV(string file) | airCleaners | false |

### Compute Cleaner Statistics

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| cleanerController.computeStatistics(Cleaner cleaner) | Cleaner with identifier Cleaner0 | null |