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

This method uses several other functions to produce the wanted result:

1. removeAllMeasurementsFromSensor(Measurement[] measurements, Sensor sensor)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. removeAllMeasurementsFromSensor(Measurement[] measurements, Sensor sensor) | getAllMeasurements() Sensor0 | Will give all the measurement except sensor0’s ones : Her we only have sensor1’s measurements |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. removeAllMeasurementsFromSensor(Measurement[] measurements, Sensor sensor) | getAllMeasurements() sensor101 | null |

1. FR5\_malfunctioningAnalysis(in sensorToCheck:Sensor): double

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR5\_malfunctioningAnalysis(Sensor sensorToCheck) | sensor0 | 0.08 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR5\_malfunctioningAnalysis(Sensor sensorToCheck) | sensor101 | null |

### MeanAir Quality

|  |  |  |
| --- | --- | --- |
| 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 | 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

This method uses several other functions to produce the wanted result:

1. FR7\_averageValue(Sensor sensor, Attibute targetAttribute, time\_t t1, time\_t t2)

Compute the average value of all its measurements for the given attribute

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR7\_averageValue(Sensor sensor, Attibute targetAttribute, time\_t t1, time\_t t2) | sensor0  03  01/01/2019 12:00:00  01/01/2019 12:00:00 | 50.25 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR7\_averageValue(Sensor sensor, Attibute targetAttribute, time\_t t1, time\_t t2) | sensor0  o2  01/01/2019 12:00:00  01/01/2019 12:00:00 | null |

1. FR7\_sensorComparison (Sensor sensorToCompare, timestamp t1, timestamp t2)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR7\_sensorComparison (Sensor sensorToCompare, timestamp t1, timestamp t2) | Sensor0  01/01/2019 12:00:00  01/01/2019 12:00:00 | A map with Sensor object and a double value (similarity 0 to 1) :  Sensor1 and 0.846 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR7\_sensorComparison (Sensor sensorToCompare, timestamp t1, timestamp t2) | Sensor101  01/01/2019 12:00:00  01/01/2019 12:00:00 | null |

### Air Quality

This method uses several other functions to produce the wanted result:

1. isGivenTimeInsideTimePeriod(time\_t start, time\_t : stop,time\_t: time): bool

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. isGivenTimeInsideTimePeriod(time\_t start, time\_t : stop,time\_t: time): bool | 01/01/2019 12:00:00  01/01/2019 12:00:00  01/01/2019 12:00:00 | true |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. isGivenTimeInsideTimePeriod(time\_t start, time\_t : stop,time\_t: time): bool | 01/01/2019 12:00:00  01/01/2020 12:00:00  01/01/2025 12:00:00 | null |

1. distanceBetweenPositions (double latitudeA, double longitude, double latitudeB, double longitudeB)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. distanceBetweenPositions (double latitudeA, double longitude, double latitudeB, double longitudeB) | 44  -1  45  -2 | 1.41 |

1. FR8\_qualityAttributes (double latitude, double longitude, time\_t time)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR8\_qualityAttributes (double latitude, double longitude, time\_t time) | 45  -2  01/01/2019 12:00:00 | Returns a Map<Attribute,double> with the Attribute object’s identifiers o3, No2, So2 and PM10 and their corresponding measurement:  O3 : 55.56  NO2 : 69.28  S02 : 38.56  PM10 : 47.39 |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR8\_qualityAttributes (double latitude, double longitude, time\_t time) | 45  -2  15/15/2025 12:00:00 | null |

1. FR8\_quality (double latitude, double longitude, time\_t time)

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR8\_quality (double latitude, double longitude, time\_t time) | 45  -2  01/01/2019 12:00:00 | String equal to “Mediocre” |

|  |  |  |
| --- | --- | --- |
| Tested Function | Input | Output |
| sensorController. FR8\_quality (double latitude, double longitude, time\_t time) | 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 |