

TrackMe: Acceptance testing document

Software Engineer 2 - 2018/2019

Riccardo Poiani, Mattia Tibaldi, Tang-Tang Zhou
Politecnico di Milano

Version 1.0

January 20, 2019

Contents

1	Introduction and group reference	3
2	Acronyms	3
3	Installation setup	3
3.1	Setup acceptance testing	3
4	Acceptance tests	4
4.1	Data for help tests	5
4.1.1	Goal 1	5
4.1.2	Goal 2	6
4.1.3	Goal 3	7
4.1.4	Goal 4	8
4.1.5	Other tests	8
4.2	Automated SOS tests	10
4.3	Conclusion	11
5	Additional comments	12

1 Introduction and group reference

In this document, the acceptance testing performed on the TrackMe project of the following people is presented and explained.

Name of the authors: Avila, Schiatti and Viridi.

Link to the repository: <https://github.com/lauricdd/AvilaSchiattiViridi>.

Document considered for the acceptance testing:

- Design document 2, present in the delivery folder
- Requirements analysis and specification document 2, present in the delivery folder
- Implementation and testing document, present in the delivery folder

2 Acronyms

- Gx: goal number x. The enumeration follows the one present in the second version of the RASD document
- Dx: domain assumption number x. The enumeration follows the one present in the second version of the RASD document
- UCx: use case number x. The enumeration follows the one present in the second version of the RASD document
- RASD: requirements analysis and specification document
- DD: design document
- ASOS: automated sos

3 Installation setup

For installing and running the project, the section 7.1 of the implementation document has been followed. A comment on this could be that the instructions presented are not much detailed and consists of only three steps. Furthermore, one may notice that having performed this initial release with docker, forces a user having a windows non-pro/ultimate version to install a virtual machine with an operative system that supports docker. However, this is considered to be reasonable, but some comments on this not-so-exceptional case would have been welcome.

However, the docker environment allows to setup the whole system really fast, and without any problem encountered.

3.1 Setup acceptance testing

The requirement to launch the JMeter acceptance testing are:

1. The server of the other team running: needs Docker and Docker Compose
2. JMeter

For each test inside the DeliveryFolder/tests we open it on JMeter:

1. After opening JMeter GUI we can run the tests by clicking the green "play" button at the top bar.
2. To see the results, you need to check for each Thread Group the View Results Tree and Assertion Results listeners.
3. From the View Results you can see the response and request of each HTTP requests inside that Thread Group while for the Assertion Results you can just see if there is an error in the assertions done. You should not find any error in Assertion Results. But in View Results there could be some red requests because the response code is different from 20X.

It is important to note that all JMeter testing uses an UUID for creating users or third parties to have a random ssn or taxCode. In such a way the JMeter test can be repeated every time without resetting the server. It is very unlikely that an UUID generated equal to a previous one.

4 Acceptance tests

The approach to acceptance testing, for both data for help and automated SOS, has been conducted in the following way. First of all, it has been decided to test all the possible use cases that are described in the second version of the requirements and analysis document (of course, only the scenarios that regards the implemented requirements and goals, mentioned in the implementation and document testing, have been considered).

The exceptions of the various use cases have been considered too, since the system should behave properly, also, in these conditions.

Secondly, the website has been tested manually, by exploring the UI and its possibilities.

The tests that regards the user cases have been performed with JMeter. The source code is present in the delivery folder. Before starting the testing phase it is necessary to know the structure of the JSON files that are sent to the server. To do this we used one of the JMeter feature: HTTP(S) Test Script Recorder which can be used to record all the requests a web application is making to the server. As the name suggests, it will capture only the HTTP(s) requests. To make it work, it is necessary to set a Web Proxy on the browser to localhost:port (e.g. 8888 the standard port of the Test Script Recorder) and start the recording. After this it is possible to navigate on the browser and every request will be recorded and added on the "Recorder Controller" (look RecordingTemplate.jmx file)

The cases are presented in the next two subsections, each of them, first exposes the tests performed on the use cases divided by goal. After that, the manual tests on the UI are presented.

Finally, at the end of this chapter, it is possible to find some brief conclusions.

4.1 Data for help tests

4.1.1 Goal 1

Goal 1 is "The individual could allow (or refuse) Data4Help to use their data".

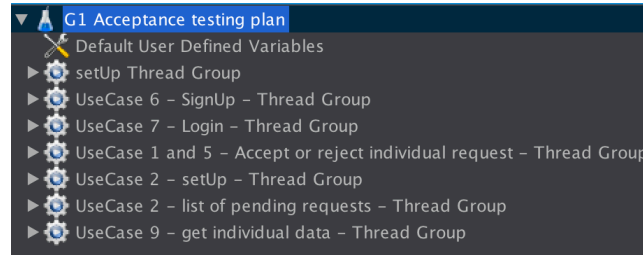


Figure 1: JMeter G1 Acceptance Testing

The uses cases that concerns G1 are: UC1, UC2, UC3, UC4, UC5, UC6, UC7 and UC9. These have been retrieved by looking at the requirement traceability matrix in the RASD.

- UC1 regards the scenario in which a user accepts or refuses a request. The successful case has been tested (no exception is present) by using some requirements that involves the third party registration and the possibility to send individual request. In particular, after having created a new third party customer, and a new user; a request from that customer to the user is made. The user retrieves the list of pending requests (that contains only that element) and approves/rejects it.
- UC2 involves the fact that a logged individual can access his pending requests. For testing it, a great number of third party customers has been registered: they all send a request to the same user. The user retrieves all the pending requests and a check on the number of pending request is performed: the behaviour is correct. The exceptional case, requires the use of the front end: in particular, it is said that if no pending requests are available, then a message should been shown to the user: however, no message is displayed.
- UC3 and UC4 regards sending notification to the third party when a user approve or reject an individual request. By recording the HTTP Request after changing a request to approved or rejected, this does not invoke any HTTP request. Therefore, we suppose that they are not working properly since it is not testable.
- UC5 regards the acceptance and the refusing of requests. In particular, they are similar to UC1, except for the fact that they are analyzed from a third party customer point of view. For this reason, they have been tested together with UC1. In particular, after that the request is approved, we have checked that in the subscriptions of the third party there exists one on a SSN that is equal to the user that accepted the request.

- UC6 involves the sign up of users and third parties. The successful case has been tested and also the exception cases in which SSN, email and taxCode are already present in the system.
The result is the following: userId and accessToken returned by the server are correct. In particular, they have been used to check if the user was registered by performing an HTTP request to retrieve the user's information.
The exception about checking if fields are empty or not has been checked with manual testing since this is something that is implemented in the front end.
- UC7 consists in logging users and third parties into the system. The normal event flow and the cases of wrong credential has been tested. The same check adopted in UC6 has been used here.
The exception that involves missing password or email, has been checked with manual testing since this is something that is implemented in the front end.
- UC9 regards the access to individual data. This test has also been done with JMeter using third parties and individuals which are generated from the folder src/seeder created by the other team. In particular the Delta Dore Italia third party has been used to access individual data regarding the user Andrea Esposito. Since data are autogenerated, we do not know how many data it has, but generally, after starting the server it should immediately send a data of Andrea Esposito, therefore in the test we check if it exists at least one data.

4.1.2 Goal 2

Goal 2 is : "the third-party company should be able to access data of a specific individual".

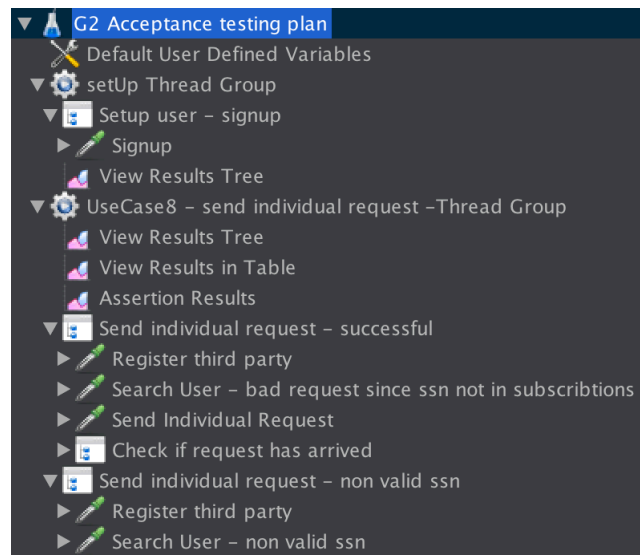


Figure 2: JMeter G2 Acceptance Testing

The uses cases that regards G2 are: UC6, UC7, UC8, UC9, UC32 and UC33.

- UC6, UC7 and UC9 were already presented above, while discussing G1.
- UC8 analyses the case in which a third party sends a request.
To test it, basically, the idea is searching for the user SSN. If it is found, it returns a bad request (this seems strange, but it is the correct chain-flow) with a specific message "Should send a request to the individual to access his data". This happens because the third party customer is not subscribed to the data of that user, and therefore a request has to be sent. Thus, after that, it is possible to send a request: to check if things work properly, we have to be sure that the user has received a new pending request. In the end the result is correct and the exceptions have been tested successfully.
- UC32 and UC33 regards the receive of notification of approved/rejected individual requests by third party or AutomatedSOS. For the same reason of UC3 and UC4, they are not working properly for us since they are not testable.

4.1.3 Goal 3

Goal 3 is "the third-party company should be able to access anonymized data of groups of individuals under certain constraints".

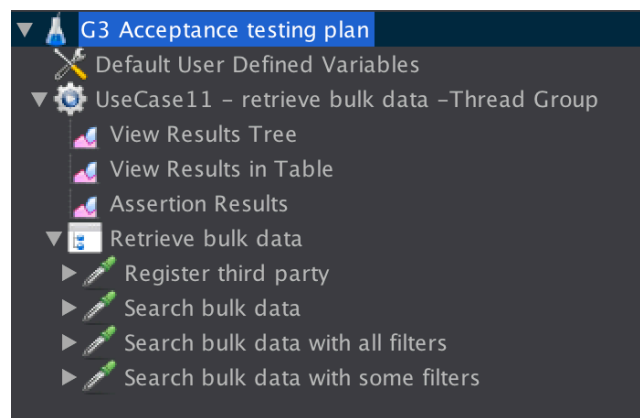


Figure 3: JMeter G3 Acceptance Testing

The uses cases that involves G3 are: UC6, UC7, UC11 and UC12.

- As already mentioned, the first two use cases are already tested above during the test of G1.
- UC11 and UC12 regards the access to bulk data. During the test of UC11 and UC12, it is impossible to understand when there is an error message. In fact by doing the manual test, even 3 blocks of anonymized data are returned and no error message is shown. This is in conflict with what is expressed in the description of the use case of the RASD document. Indeed, it is mentioned that "if there are less than 1000 individuals in the

request of data, an error is returned": however, this does not happen. To test the normal flow, however, one should note that to have many data into the system, it is necessary to leave the server up: the more the time passes, the more the pieces of data available. In UC11, we test the response code of the HTTP request to access the bulk data: the behaviour is correct. This is due to the fact that there is no possibility of inserting data, and, therefore, it is not possible to know which and how many pieces of data are available, in order to do more specific controls.

4.1.4 Goal 4

Goal 4 is "the third-party company could subscribe to get new data related to specific individuals or previously saved search".

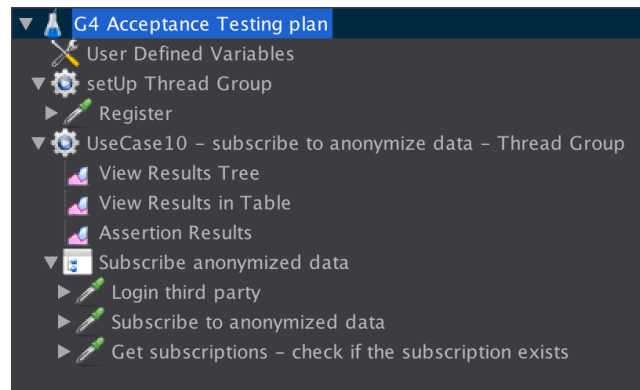


Figure 4: JMeter G4 Acceptance Testing

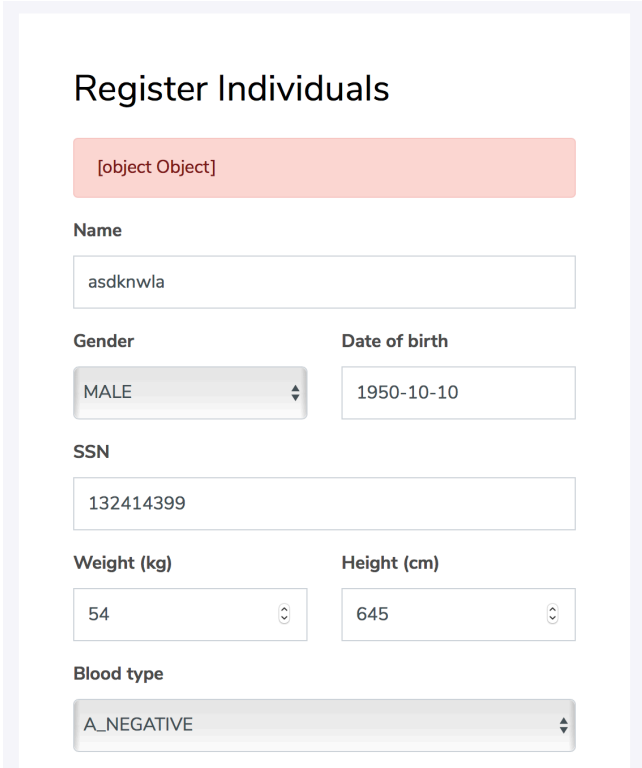
The uses cases that concerns G4 are: UC6, UC7, UC9, UC10, UC11, UC12, UC32, UC33 and UC34.

- UC6, UC7, UC9, UC11, UC12, UC33, UC32 are already discussed above
- UC10 consists in the subscription of a third party customer to new user data.
In the website, the subscribe is done when a checkbox is crossed during the search of bulk data. While testing with JMeter, the search has been skipped and only an HTTP request to directly subscribe has been done. To check if things work properly, the subscriptions of that third party are retrieved: since the subscription id is present, the result is correct.
- UC31 regards the send of individual data to third parties subscribed. This process has been discovered during the capture of HTTP Request by using JMeter. Therefore, we suppose that UC31 is working properly since other testing cannot be done due to random generation of data.

4.1.5 Other tests

As already mentioned, some other tests have been done, manual, by exploring the UI.

Indeed, once the application is installed, it is possible to test the correct functioning of the graphic interface. The site appears to be well structured in its parts and graphically pleasing. The test has found some graphic bugs that do not affect the correct achievement of goals. The operations that a user can do on the site are all fairly clear and easy to understand, sign of a correct structuring of the user interface. The only operation not explained is the registration of the third party, where it is required to enter a certificate, without explaining what it is. Another problem is when a user try to sign up with an e-mail that already exists into the system: an appropriate error message is not shown. (see the figure 5)



The image shows a web form titled "Register Individuals". At the top, there is a red rectangular box containing the text "[object Object]". Below this, the form contains several input fields: "Name" with the value "asdknwla", "Gender" with a dropdown menu showing "MALE", "Date of birth" with the value "1950-10-10", "SSN" with the value "132414399", "Weight (kg)" with the value "54", "Height (cm)" with the value "645", and "Blood type" with a dropdown menu showing "A_NEGATIVE".

Figure 5: UI Bug: [object Object]

We have also performed some tests that regards the registration of third party customer, that sends to the users previously created individual requests and group requests. Finally, back to the user profiles, we have checked that the requests have arrived and that the user can refuse or accept the requests received.

Other than UI tests, we have also performed a test to check if the public API of an individual can be accessed with the userId and accessToken of a third party and viceversa. Indeed, this operation cannot be done and a error message is returned: "The current user is not authorized to perform the operation" (this

test is present in OtherTest.jmx of JMeter).

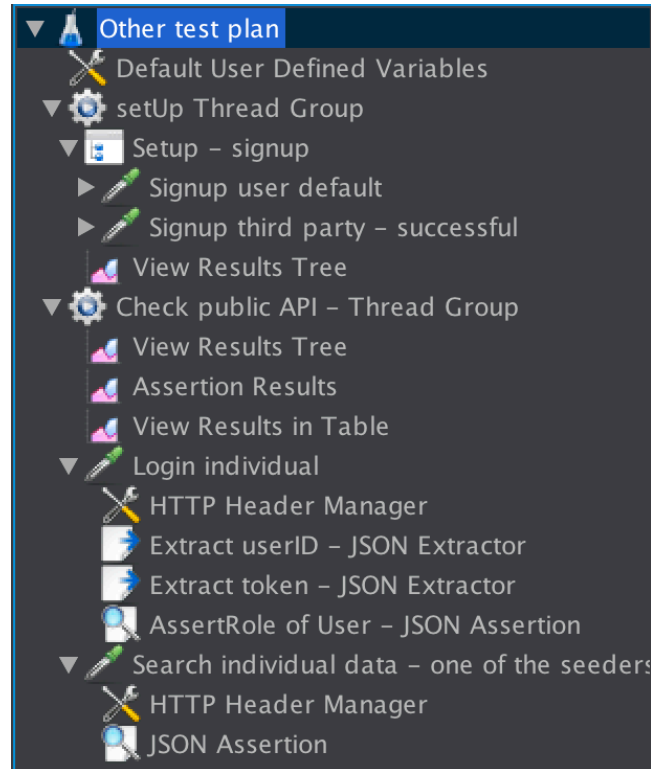


Figure 6: JMeter Other Tests

4.2 Automated SOS tests

As indicated by the other team, the Automated SOS service is based on: a schedule job executed every 24 hours to send request regarding the usage of the ASOS service to the elderly people (over 60 years old). To avoid waiting so much time, we did a little modification on the source code of the project. One class called "src/data4help/src/main/java/avila/schiatti/virdi/Main.java" has to be modified:

- Substitute the code

```
ASOSRequestScheduler.create()
```

with

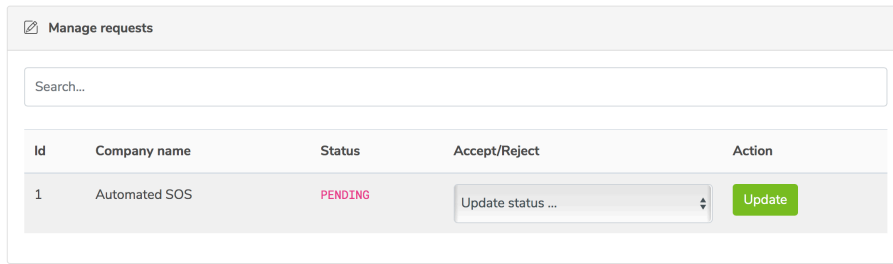
```
ASOSRequestScheduler.create().setPeriod(10).setTimeUnit(TimeUnit.SECONDS)
```

- Add the import of the class TimeUnit on the top of the class

```
import java.util.concurrent.TimeUnit;
```

After this modification, we still could not test the functionality of ASOS. Then after the other team fixed the bug, we tested it and we saw that the new user received a pending requests from Automated SOS. This test regarding UC13 has been done manually due to some unknown problem in JMeter (no new requests were found).

- UC13 regards the creation of ASOS Request to elderly people



Manage requests				
Search...				
Id	Company name	Status	Accept/Reject	Action
1	Automated SOS	PENDING	Update status ...	<button>Update</button>

Figure 7: UI Received ASOS requests

The other UC14, UC15 and UC16, regarding receiving health data, sending health data and contacting emergency point, cannot be checked since the send of data cannot be done manually.

4.3 Conclusion

The prototype has been developed until a good point, but it still contains some bugs. In particular, many of them regards the user interface and the website. For what concerns the single goals, we can say the following:

- It is possible to state that G1 has been reached. However there are some problems with the notification of third party when individual requests are approved or rejected.
- A same reasoning can be applied also to G2. Indeed, UC32 and UC33 present problems with the notifications
- G3 has not been fully reached. To be precise, there is the problem with the constraint on 1000 individuals while accessing bulk data
- G4 is achieved correctly
- For what concerns G5, it has been possible to test graphically only the fact that, after having registered a user older than 60 years old, and when 10 seconds are elapsed, a request arrives from ASOS, that is some kind of special third party customer.
As mentioned in the section regarding the test of G5, the other functionalities could not be tested.

In conclusion, the prototype is not ready to be deployed on the market due to previous problems.

5 Additional comments

Here it follows a list of consideration and comments, that we would like to point out:

- The implementation and testing document, in our opinion, misses some points. In particular, a discussion on how the frameworks adopted match, in some sense, the requirements and the goal of the project is not present at all: what is possible to find is only a list of advantages and disadvantages of the framework, and most of them are not related to the system developed, but very general
- Motivations regarding the choice of implemented features of automated SOS is not present at all
- In the section regarding the test, it is said that "tests were done by hand" This choice has not been motivated and, of course, it presents obvious disadvantages: it is necessary to repeat all the tests as soon as an update or fix is released
- Most of the code that we looked at, in particular the backend, misses most of the documentation. For very few methods, some sort of comments is present
- It is not clear why, for accepting a request, it is requested to use also the SSN: the token should be the only thing relevant, from a API design point of view
- No information regarding the coverage of the test is present in the documentation provided; in our opinion, this is an important information for the stakeholders.
- While searching for individual data, if a wrong SSN is inserted first, and then a correct one, the error message does not disappear.

Make new search

Individual data

Bulk data

The provided user id or ssn is not valid or null

SSN

Last search

```
{ "ssn": "787392090" }
```

Location (Lat, Long)	Heart Rate (bpm)	Systolic Blood Pressure (mmHg)	Diastolic Blood Pressure (mmHg)	Body Temperature (°C)	Blood O ₂ Saturation Level (%)
42.70006, 17.84854	161	133.387695	70.651145	39.661609	94

Figure 8: UI bug

Other mini bugs like this are present. However, they are not important since the UI is not considered to be the main purpose of the project. This one has been reported as an example, for the sake of completeness.

- On Safari there are some problem with the registration: first it is impossible load a certificate during the third party registration, in fact though the user manages to load the certificate, the system always returns an error message saying it is missing. Secondly, the date picker is not working.

Register Third Parties

Company name

Tax code

Phone

Certificate

11.LP-simplex-FOR-16.pdf

Certificate is required

Email

Figure 9: UI Certificate bug

- The site is not responsive and can't be accessed, for instance, from a smartphone or from device with a resolution that is too low