TrackMe: Acceptance testing document
Software Engineer 2 - 2018/2019

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

Version 1.0
TODO LINKS HERE

January 19, 2019

Contents

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 Virdi.
Link to the repository: `https://github.com/lauricdd/AvilaSchiattiVirdi`.
Document considered for the acceptance testing:

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

2 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 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 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, due to the fact that an user may behave in non-predicted way, other cases, for some requirements have been considered.

The cases are presented in the next two subsections, each of which first exposes the tests performed on the use cases divided by goal. After that, special cases and more strange behaviours are considered.
Finally, at the end of this chapter, it is possible to find some brief conclusions.

3.1 Data for help tests

3.1.1 G1

The uses cases that concerns G1 are: UC1, UC2, UC3, UC4, UC5, UC6, UC7 and UC9.

For what concerns UC1, 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 and rejects it.

UC2 to be completed

UC3

UC4

UC5

UC6 involves the sign up of user and individual requests. The successful case has been tested and also the cases in which SSN and email 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 skipped since this is something that is implemented correctly 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 skipped since this is something that is implemented correctly in the front end.

3.1.2 G2

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

The first two use cases are already tested above during the test of G1.

For what concerns UC8 basically the idea is searching for the user SSN. If it is found it returns a bad request (i.i. this seems strange but it is the correct chainflow) with a specific message "Should send a request to the individual to access his data". Then it will send the request: to check if it works in the test and after it checks if the user received has a new pending request. The result of the test is correct, but the choice to use a bad request message as an answer to a correct operation is a bit strange. Also the exception 2 has been tested and it checks that an error message will be shown.

UC9 Test on this use case is skipped.

Instead UC32 and UC33 have not been implemented by the other team.

3.1.3 G3

The uses cases that concerns G3 are: UC6, UC7, UC11 and UC12.
As already mentions the first two use cases are just tested above during the test of G1.

During the test of UC11 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 that is expressed in the description of the use case of the RASD document. For checking the creation of new data into the system just leave the server up. The more time the server is up, the more data there are (i.i. every minute new data arrives), therefore we just check if the response is OK.

Since UC12 seems to equivalent to UC11, therefore results already checked.

3.1.4 G4

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

UC6, UC7, UC9, UC11, UC12, UC33, UC32 already discussed above.

UC10 consists in the subscription to new user data by third party customer. This case has been tested with the website. The subscribe is done when a checkbox is crossed during the search of the bulk data. Here in this test of jmeter, the HTTP method of post subscriptions has been done without doing the search of bulk data. And then to check if it works, we went to the subscription of the third party and see if there is the one that has been inserted (by checking subscription id).

UC34 has not been implemented by the other team.

3.1.5 Other tests

3.2 Automated SOS tests

Mystery of faith!!

3.3 Conclusion

4 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

- Even if is present a sort of domain assumptions regarding security (i.e. D2 TrackMe addresses data protection and integrity against possible attacks), it is not clear what this concerns. Therefore, since we considered important, as users, to don't access APIs that concerns other users, we have tested also this possibility.
  One should also note that password are saved in clear.
- 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
- The code of aos.js misses, for some reason, all sort of indentation