POLITECNICO
MILANO 1863

Software Engineering II project

Academic Year: 2018/2019

TRACKME

Acceptance Test Document

Version 1.0 - 20/01/2019

Davide Rutigliano - 903616

Claudio Ferrante - 903417

Davide Matta - 920349

Contents

1 Scope and Purpose

This document is intended for analyzing *TrackMe* software application and specifications through analysis of test cases extracted from requirements explained in documents presented.

2 Project Analyzed

Authors:

- Lorenzo Francesco;
- Negri Virginia;
- Molteni Luca.

Repository: https://github.com/iPhra/LorenzoMolteniNegri

To perform the acceptance on test cases we used all the documents provided by authors: *Requirement Analysis and Specification*, *Design* and *Implementation and Test* documents. The latter has been primarily used for installation and setup instructions; first two instead have been used for extracting test cases in order to proceed with the analysis.

3 Installation Setup

3.1 Server Installation

Back-end installation was straightforward. However, database installation wasn't.

We had to ask further instructions to setup the database and explicitly asked developers to get a database dump: it was assumed the customer is able (and want) to recreate the database from scratch following the tables described in the document.

Moreover, it was not explained anywhere how to set server's database configuration for running tests (i.e. modifying parameters in *Backend/NodeJS/config/test.json*).

3.2 Client Installation

Client installation was straightforward and quite simple in case you have a Mac. A problem found is that in the installation instruction is not stated anywhere that you should have OSx Mojave and XCode 10.1 (latest version) in order to run the application.

4 Acceptance Test Case

In this section, we will provide the analysis of the acceptance of the test cases we have extract from the documents provided.

Below there is a list of functionalities extracted from requirements, their description and if they correctly works and has been tested. At all, all functionalities seems to have been tested.

Most tests has been done properly and most part of functionalities are tested both in case of success and in case of failure (i.e. bad data and bad/unauthenticated requests).

Registration: The Registration functionalities works properly. This feature has been tested analysing all the different cases: wrong password, email, FC, birth-date and sex or wrong IVA, company name and description.

Moreover, it has been tested also that users cannot register with the same parameters twice.

Login: The Login works correctly. Cases as try to login as a not-existing user or to login providing a wrong password have been tested. Furthermore, user whose account has not been activated cannot log into the system.

Profile Data Management: The application allows users, both individuals and third-parties, to manage correctly their profile. In particular, they can retrieve and update their private settings. A possible limit we have found is that it is not possible to edit the submitted email.

Upload Data: The system allows single users to upload new data correctly and to retrieve statistics about their imported data. Furthermore, it checks that wrong data are not present with respect to data type, timestamps and values.

Make Requests: The application allows Third-Parties to make in a correct way individual or group requests. In particular for what concerns individual requests, it does not allow to send a request to a not-existing user or to an user with an other request pending, coming from the same Third-Party. Moreover, the system checks requests with wrong subscription or duration value and that groups requests contains correct parameters.

Manage Requests: The system allows users to accept/deny requests coming from Third-Parties, in a proper way. Failure cases such as wrong Request identifier has been tested.

View Data of Accepted Requests: The application allows Third-parties to correctly download the data of the users that have accept their requests. The system checks that the requests have been accepted and that the requests exist. In the case of Group Requests the system permits in a correct way to Third-Parties to download their data. Moreover, it checks that the parameters constraints match.

Automated Sos: The sos service has been poorly tested. We was expecting at least a test to check that "if AutomatedSOS is enabled and the System detects that a users heart rate is below or above the critical threshold for his age, an ambulance is called" and another to check that "AutomatedSOS can be enabled only if the user grants permission to make emergency phone calls", that are respectively requirements 33 and 34. Furthermore, because we does not have an iPhone, we tested the application on an emulator and at least within the emulator, nothing happens when data are under thresholds. In conclusion, we were either not able to test this functionality nor to check that it works because it was not tested.

5 Additional Points

5.1 Documents

- Standards compliance, found on RASD2 at page 73, does not seems to make sense: std. **D0-178C** is the primary document by which certification authorities such as FAA, EASA and Transport Canada approve all commercial software-based aerospace systems. We asked the development team an explanation, and they confirmed that it was indeed that standard: their system has been described as a Level D risk level which, according to online resources corresponds to *"Failure slightly reduces the safety margin or slightly increases crew workload. Examples might include causing passenger inconvenience or a routine flight plan change."*;

- The alloy model was poorly done. There are some error distributed all over the model which can be easily seen; for instance at Page 85 in Figure 35 of RASD2: multiple users have the same FC and email, that is clearly wrong;

- Requirements 14, 15 and 16 (respectively: *"The S2B allows Single Users to visualize their historical data using Time Series/statistical operator/multiple timespans"*, concerning the case of visualizing data can be unified in just one requirement in order to avoid redundancy;

- All documents are in general too much verbose and presents a lot of redundancies; for instance, presenting all the screenshot of the application distributed within the three documents or for example the repetition of the database schema and er diagram in design and implementation documents.

5.2 Software

- Automated Sos functionality does not have anything of automated at all. The system will check if health parameters are under thresholds and if yes, when you open the application (i.e. data are reloaded into the app) it will ask if you (a person in fear of his life) want to call the emergency number; then, it will work as if you called the emergency number by yourself. Furthermore, it was clearly stated into customer request: *"AutomatedSOS sends to the location of the customer an ambulance, guaranteeing a reaction time of less than 5 seconds from the time the parameters are below the threshold"*;

- iOS only was not the best choice for a prototype. That is because if you don't have a mac with Mojave and Xcode 10.1, you can not test the front-end. Furthermore, the specification document says: *"The implementation of the S2B as a mobile application ensures itself portability with regards to the user, since a porting from Android to iOS, and vice versa, would be quite natural."*: porting a native iOS application (Swift) to Android (Java) would not be quite natural.

6 Effort Spent

- **Davide Rutigliano: 12h**
- **Davide Matta: 12h**
- **Claudio Ferrante: 12h**