TrackMe: RASD
Software Engineer 2 - 2018/2019

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

Version 1.2

January 12, 2019

Contents

1 Introduction

1.1 Purpose

The TrackMe system is designed as a software application for monitoring the position and the health of the user.
The application provides three services that are described below.
The first one, Data4Help, is thought for all people that want to keep under control their health during the day or who want to always know the position and health status of a particular person in the world. Indeed, with this service a third party user can send a request to access data of some specific user, by means of his social security number: if the receiver agrees, it is possible to see the data regarding the request. The service supports the registration of individual who, by signing in, agrees that the company TrackMe acquires their data, which will be used anonymously by third parties for making statistics on groups of people.
The second one is called AutomatedSOS service. It is thought for people that have serious health problems and, in case of illnesses (i.e. some parameters observed below the threshold), the system contacts an hospital.
Finally, the Track4Run service is also available. It is developed for organizers of sport events that want to monitor the runners in a race. The service allows organizers to set up a run, participants to enroll in the run, and spectators to see the position of all runners during the run on a map.

1.1.1 Goals

The goals can be distinguished into two families: the former regarding the users, and the latter regarding the third part customers.
The ones regarding the subscribed users, are the followings:

[G1] Allow a user to access its own data

[G2] Allow a user to contribute to data sharing by providing information about his location and health status

[G3] Once the health parameters of a user have been observed below the threshold for the first time after one hour, an ambulance is sent to the user location.

[G4] The time experienced between the moment in which the health parameters of a subscribed user are observed below the threshold and the time in which the emergency point is contacted is equal or less than 5 seconds

[G5] Allow a user to participate in a run managed by third parties, as an athlete, if all starting condition are satisfied

[G6] Allow spectators (i.e. user) to see on real-time the "correct" positions of all athletes taking part in a run, with at most 15 meters of radius error

[G7] The maximum time to accept an individual request from any third party is 30 days; after that, the request will expire

[G8] Allow a user to accept or refuse a request from third parties

[G9] Allow a user to block requests made by a specific third party and all the pending requests will be refused. This action is possible only when the user has already refused one request made by the customer that he is intending to block.

[G10] Allow spectators and runners to see the leaderboard, when a run is completed

The goals of the project, regarding the third part customers, are the followings:

[G11] Allow organizers (i.e. third parties) to set up a run, by defining its name, its path, date, start time, expiration date, and the minimum number of participants

[G12] Allow a third party to access data specified in a request if the user accepts the request or if he accepted one or more requests from the same third party that provided access to the same data

[G13] Allow a third party to access statistical and anonymized data if and only if the number of individual involved is greater than 1000. This is satisfied after the request is approved

[G14] Allow a third party to subscribe to non-existing data. They will have access to them, after the data is generated.

1.2 Scope

As already mentioned, the basic Data4Help service allows to monitor the position and the health status of individuals. When a user registers to the service he accepts the application's contract, that permits the acquirement of his data from his device. The information, once received, is stored. Data4Help provides different types of diagnostic procedures: heartbeat, blood pressure and blood oxygen saturation levels.
Periodically, the user's device sends data to TrackMe, that saves them into the system.
The people, who are probably most interested in this service, are whoever has a particular attention toward the health of himself, their family, or their close friends. For instance, this application allows parents to monitor their children, when they are unable to stay with them. Moreover, Data4Help permits to the users to constantly see their health status in order to be conscious of their condition. Indeed, this will keep patients regularly updated on their progress and will provide proactive measures for a better health control. In order to allow a third party customer to see the status of an appointed individual, he needs to send him a request of sharing data by means of the form provided by the system. The receiver, obviously, can accept or reject the request according to the sender and the attached reason. If the receiver accepts the demand, the requesting customer can see his data, which is related to the date of its generation into the system. Third party customers can also demand for anonymized data.
Notes, also, that a request can expire: the user has to accept (if he wants to) before the expiration date.

Also, with AutomatedSOS service, when the health parameters of a user go below the standards, the system, within 5 seconds, contacts an emergency room. The ambulance is managed by the owner of the vehicle, and it intervenes within the timing defined by the State. TrackMe, with this feature, hopes to help hospitals and private specialists to save lives.

For using the Track4Run service, the organizers need to post a race event on the application, that contains a description and a timetable. Then, all the interested people (i.e. runners) must sign up to the race. Notes that while performing the sign up to TrackMe, a runner accepts to share his data during the competition. During the event, the application will automatically monitor the runners, and spectators will be able to follow the race through the system. Notes that the possibility of observing the runners' position is allowed only when the competition is taking place.

Summarizing, all the features can be classified into two categories: world phenomena and shared phenomena. The most relevant non-shared world phenomena are:

1. The dispatch of an ambulance
2. Athlete is running
3. A user is ill
4. Organizers set up a run
5. Athlete shows up in a competition
6. Emergency room operator provides help
7. A user is hospitalized
8. An emergency call is redirected to the nearest hospital
9. A race terminates

While the shared phenomena are:

1. Sign up and log in into the application
2. User watches a competition
3. Third party customers send requests to access data
4. Organizers post the competition
5. The automated emergency call
6. Athlete subscribes / unsubscribes to a race
7. Athlete participates in a run
8. User accepts / declines a request
9. User blocks third party customer
10. User's position is updated

11. User's health status is updated
12. Third party customer access requested data
13. A competition is canceled
14. Organizer signals that a race is closed (i.e. terminated)

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- user: a person who has registered on the system (it is equivalent to subscribed user).
- athlete: a user who has subscribed to a race and intends to participate as a runner.
- spectator: a user who wants to watch the runners' position during a race
- third party customer: a person or company that wants to access data
- organizer: a third party customer who can set up a run
- individual requests: requests that third party customers can send that allow, if permissions are granted, to access the requested data of a specific user
- aggregated requests: requests that third party customers can send that allow, if permissions are granted, to access aggregated and statistical data on a certain group of user
- health parameter: a value regarding healthcare status of a specific individual that can be heartbeat, blood pressure or blood oxygen saturation levels
- threshold: a critical value regarding health parameters, which is unique for every user
- starting condition of a run: minimum number of participants
- race's overlapping path: a path is considered to be overlapping with that of another race, if they share at least a piece of path
- autonomous call: a call that starts automatically and in which a text-to-speech API is used to communicate with the other end point. Furthermore, a voice recognition API is used to retrieve a response. For better specifying, in the AutomatedSOS service, the autonomous call contacts the hospital and communicate the health parameters and the user location, and waits the a confirmation (e.g. "OK") from the emergency room operator

1.3.2 Acronyms

- RASD - Requirement Analysis and Specification Document
- NFC - Near Field Communication
- GPS - Global Positioning System
- API - Application Program Interface
- DSS - Data Storage System
- SSL - Secure Sockets Layer
- TLS - Transport Layer Security
- HTTPS - Hypertext Transfer Protocol Secure

1.3.3 Abbreviations

Gi - Goal number i, where i is a number

Ri - Requirement number i, where i is a number

1.4 Revision history

- v1.0: The document regards analysis of requirements before starting other processes such as design or implementation.
- v1.1: The document has undergone some changes after analyzing the design document:
  - the requirement R45 has been added, otherwise third party could not manage the race.
  - the domain assumption D6 has been changed slightly.
  - a new background application to be designed has been added in the product perspective and then, also, a communication interface for this has been added.
  - new use case about view list of managed race has been added to the use case diagram
  - name changing in the state of the request state diagram
  - changing every word about "as soon as" in "after", because "as soon as" was a requirement too strict (ambiguity issue)
  - the domain assumption D5 has been changed slightly
  - the requirement R16 has been changed slightly
  - the domain assumption D7 has been deleted because it was not really a domain assumption
  - two abbreviations have been added
  - in the product functions a section about future data in data requests from third party customers is moved in subscriptions to new data reformulated with a new phrase.

  - the goal G9 has been extended and the same modification is propagated to the requirements
  - the requirement R46 has been added regarding the extension of G9
  - change text caption under figure 15 since android is too specific at this abstraction level
  - the requirement R6 has been deleted because it is not necessary
  - the reliability has been slightly changed due to a unspecific non-functional requirement
  - the availability has been explained better
  - the portability has been changed due to incoherence with the design document

- v.1.2: The implementation is partially finished and there are some changes due to this:
  - the domain assumption D8 in the functional requirements section was different from the D8 of domain assumption section, therefore, the two has been merged into the D8 of the functional requirements one.
  - G9 has been better specified
  - R29 has been modified in order to reach the new G9
  - R5 has been removed since it is now considered to be wrong
  - the calls service in the software interfaces has been modified since it was imprecise

1.5 Reference documents

This RASD refers to different documents:

- IEEE standard on requirement engineer
- Slides about the courses of software engineer 2
- old RASD documents such as: "the RASD to be analyzed", RASD of 2015/16 and 2016/17.
- Wikipedia page about Regulatory Compliance

1.6 Document structure

The document structure follows what is defined on the "IEEE standard on requirement engineering":

- The first section introduces the project, its goals and its world and shared phenomena. Then, it specifies special keywords to try to resolve some ambiguity that might appear to the reader.

- The second section of this document tries to specify better the system to be implemented. First, it states better what the product to be is and what its core functions are. Furthermore, in this section, it is defined who this application is designed for and the domain assumptions regarding the world environment.
- The third section is more about requirements. But before specifying which requirement needs to be implemented, it states what are the interfaces of the application (i.e. software, hardware, communication) are; then for a better understanding of the use case, it shows a list of possible scenarios and their corresponding use case. And only after these, requirements are listed (functional and non-functional requirements).
- The fourth section is all about Alloy. It is listed all the code written to model the system application and maybe some comments about it to define it better.
- The fifth section shows more or less the effort spent by each author of this document.

2 Overall Description

2.1 Product perspective

TrackMe is a system which has to be designed as a completely new platform. It can be divided into two parts: a system of software application designed for the stakeholders (i.e. users, third parties) and a core system interacting with the application via Internet.

The former one is intended to be a mobile application which requires, necessarily, other devices to work as expected by the functionality defined in the Product functions section. For instance, a possible gear to interact with the mobile application is a smartwatch, which helps, mainly, the application to gather information about the user's health data. Therefore, a background application for the devices collecting data has to be designed to send collected data to the mobile application. Another essential requirement for the application is to have a stable connection to the Internet; without this obligation, the core system cannot collect information about the user. The requirement regarding the network connection is the only one mandatory for, also, the third part customers.

The core system has to provide a central connection for every user. The most important object is the data. Overall, the TrackMe system is designed to share data and information about users. Therefore, a Data Storage System (DSS) is necessary and enables the core application to be able to save data and share it when asked. For instance, the DSS can be a database that can be accessed through standard interfaces. Another essential requirement for the system is to safeguard the data collected; during the connection between the mobile application and the core system, TrackMe has to guarantee that nobody is tracing their data. Therefore, it is necessary to use some sort of strong network security, such as SSL/TLS protocols. Further the functions of AutomatedSOS and Track4Run are satisfied by using APIs of other companies (e.g. the one needed for maps and voice recognition).

Regarding the environment of the TrackMe system, the following diagram (Figure: 1) is provided to describe better the domain model adopted:

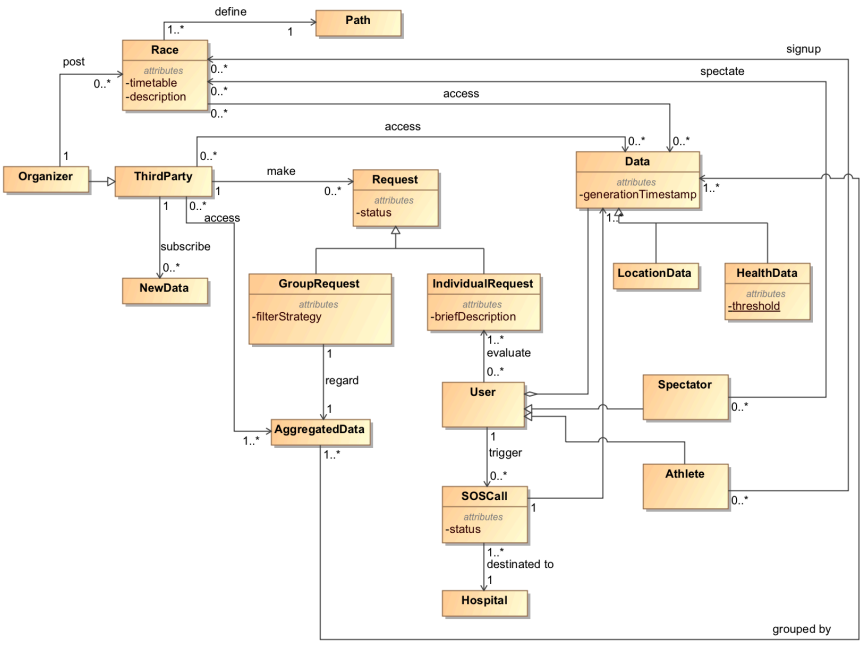

Figure 1: Class diagram of the environment

This diagram specifies the interaction with the actors and the objects of the world. The core point of the environment is the data, but what should be analyzed are its entry and exit points (i.e. usage):

1. Request: an entry point essential to share the data;
2. SOSCall: a very important feature for unhealthy people;
3. Race: necessary for runners.

2.1.1 Request perspective

Since people's data are very confidential information, a request has to be asked if someone desires it. Therefore, the design of how requests should work is essential. To give a better understanding of this, the following state diagram (Figure: 2) describes the possible states of a request:

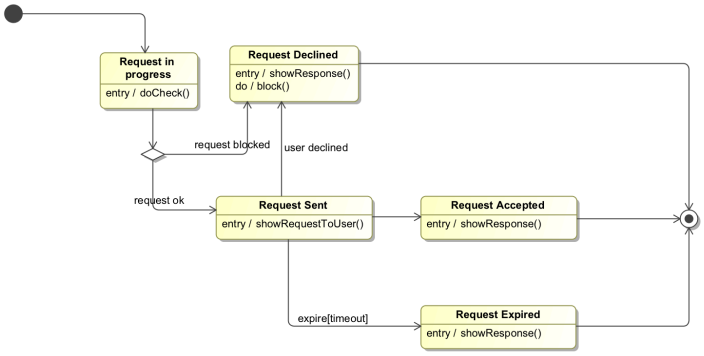

Figure 2: State diagram of a request

2.1.2 SOSCall perspective

For unhealthy people, a latency in a help call is a problem of life and death. Therefore, a better description of these calls is crucial. The following state diagram (Figure: 3) describes the possible states of a SOSCall:

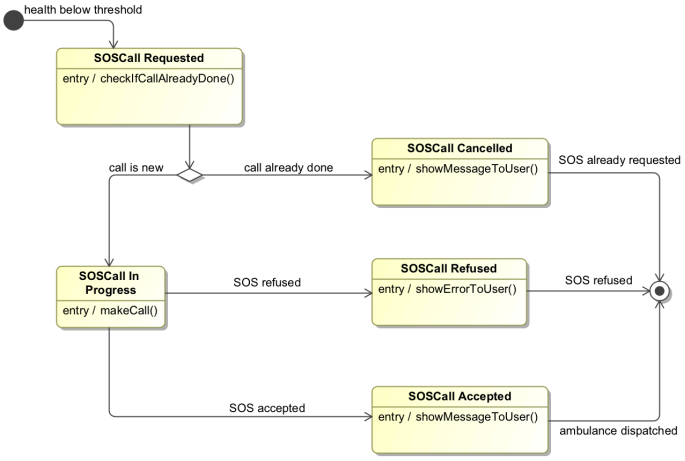

Figure 3: State diagram of a SOSCall

2.1.3 Race perspective

For someone, running is something that they cannot live without; since spectators can watch the runners' position every time, during a race, it is important to describe better how a race works for improving and understanding the problem of data privatization. Thus, the following state diagram (Figure: 4) is shown:

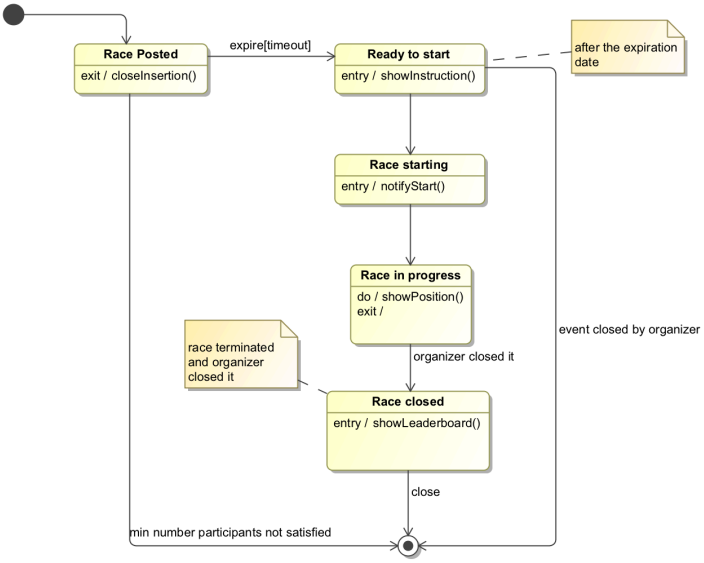

Figure 4: State diagram of a race

2.2 Product functions

The major functions of the projects can be divided and explained into four more specific aspects that are listed below:

2.2.1 Monitoring user's location and health status

Users subscribed to the application have agreed to being constantly monitored: in particular their locations and health statuses are kept under control. The information collection process continuously receives updated data from the users.

2.2.2 Data requests from third party customers

Third party customers can send two types of requests: individual requests and aggregated requests. In order to perform the former, the petitioner must provide the social security number of the individual and a brief motivation: after he accepts (if he does), the access will be eventually granted to the customer. Individual requests regard any subset of the stored information of the specified user. More specifically, this last point means that third parties can ask, also, for information that will be generated, for instance, during the next month.
Note that, in the case in which a request will be pending for 30 days, it will expire and the user will no longer be able to accept it.
Furthermore, it is also possible for an user to block, and thus prevent, requests that he receives from a certain third party customer: in this case the system will abort the demanding process at the beginning, during a brief analysis.
Aggregated requests involves, instead, anonymized set of people registered in the application. This access will be granted automatically by the system in the case in which the dimension of the group is greater than 1000.

2.2.3 Subscriptions to new data

Third party customers are allowed to subscribe to aggregated data and individual data that still does not exists: the access will be granted after the data is generated. Furthermore, for individual data it is possible to define a well-defined and limited period of time specified in the request.
For instance, a petitioner could ask for the health statuses of the next month that belongs to old people (e.g. age is greater than 68) that lives in Milan.
The same constraints on anonymization (i.e. dimension of the group is greater than 1000) is applied here as well.

2.2.4 Automated calls of SOS help

When the health parameters of an user are detected below a certain threshold, an automated call to the nearest hospital is performed by the user device within 5 seconds.
The call is handled in an totally autonomous way: the hospital can accept or declines a certain request of help.
Furthermore, the call can be requested only every minute and won't be performed in case another call from the same user has been already accepted in the previous hour: this will prevent call flooding toward the hospital.

2.2.5 Run organization

Organizers of run events, are able to select a date for a run, the starting time, to define its path, to specify an expiration date for the subscriptions, and also to choose a minimum number of participants. Since a race is set up, athletes can subscribe and unsubscribe to the run, according to their will and preferences.
When a race is taking place, the athletes' information is monitored: their position can be seen on a map by users.
An event may not be able to start (i.e. not enough runners are present): in this case all the participants are notified.
When a run is completed the organizer will set the race's state to closed and the leaderboard will be announced; furthermore the position information regarding racers will no longer be available to the spectators.

2.3 User characteristics

The following actors are the users of this applications:

- User: this is a person who is successfully registered to TrackMe: his personal information will be collected, and he can accept or refuse requests of accessing his private data. Furthermore, he is provided with an autonomous SOS help, can watch positions of runners during a race, and can enroll and participate in running events
- Third party customers: these are clients whose purpose is to access anonymized data and individual data according to their needings. Furthermore, they can also organize running events

2.4 Assumptions, dependecies and constraints

2.4.1 Domain assumptions

[D1] User's smartphones are equipped with GPS sensor

[D2] Users own devices with sensors to acquire data information about health status

[D3] Health data collected by the devices are correct

[D4] There is at least an external service of trusted companies which provides the possibility to the user to view detailed maps

[D5] There is at least an external service of trusted companies which provides the possibility to make calls

[D6] There is at least a messaging protocol of trusted companies which provides the possibility to transfer data from devices containing sensors to acquire health data, to smartphones

- ~~[[D7]] Everyone has a method to access for the first time to services offered by TrackMe~~

[D8] Hospitals always accept new SOSCall.

[D9] If SOSCall are accepted, then an ambulance is sent to the location mentioned by the call

[D10] Organizers set up races using only paths that are walkable

[D11] When a user's phone GPS is set on high precision, then it provides the right position with at most a radius error ranged from 0 to 10 meters

[D12] Athlete participating in a run are equipped with a device sharing GPS position set on high precision

[D13] Hospitals are always reachable through help calls

[D14] Every user enrolled in a run are able to participate in the run as athletes if all starting conditions are satisfied

3 Specific Requirements

3.1 External interface requirements

3.1.1 User interfaces

This section puts focus on the relation between the user and the mobile application. For better understanding the set of functionality of the application and the interaction with the user, beneath there are some mockups that represent a basic idea of what the mobile app will look like.

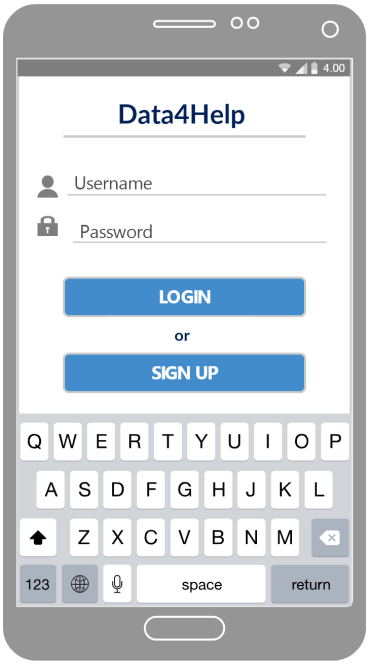

Figure 5: Mock up: Login.

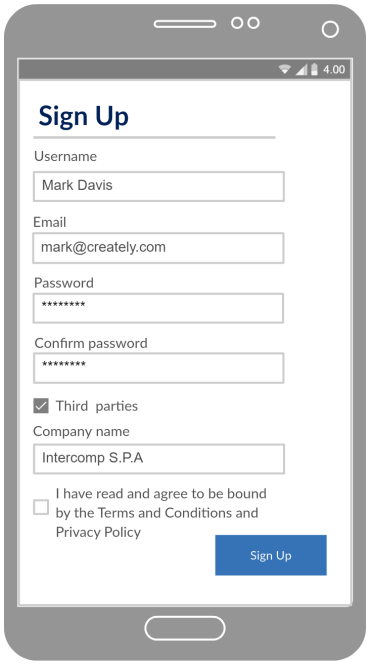

Figure 6: Mock up: Sign up.

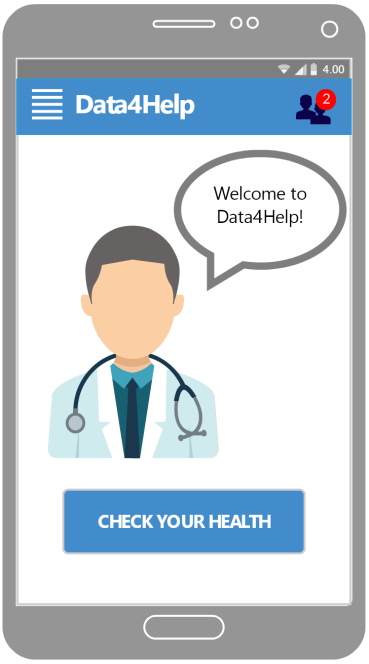

Figure 7: Mock up: Home.

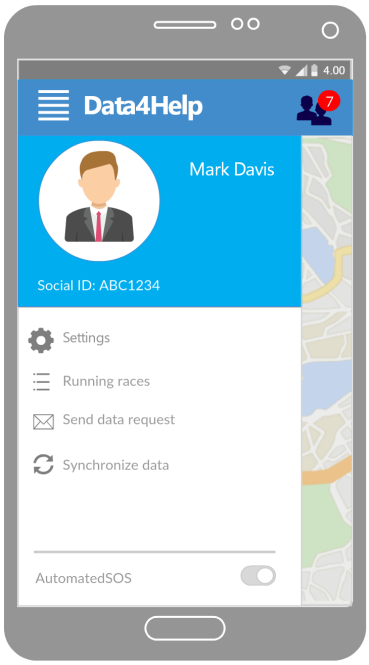

Figure 8: Mock up: Main menu.

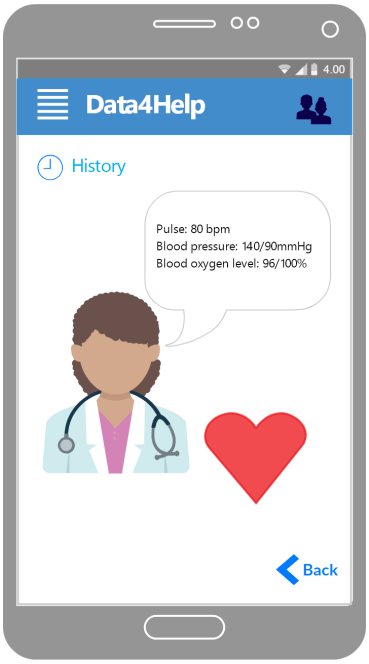

Figure 9: Mock up: Health status.

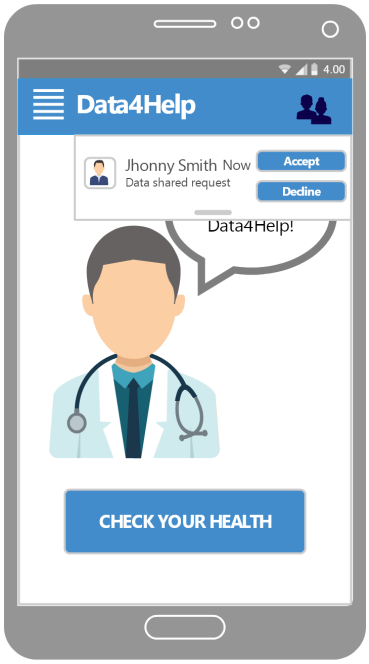

Figure 10: Mock up: Manage the request.

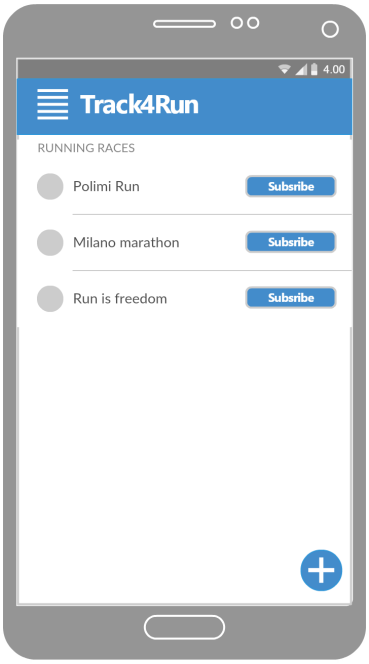

Figure 11: Mock up: List of available races.

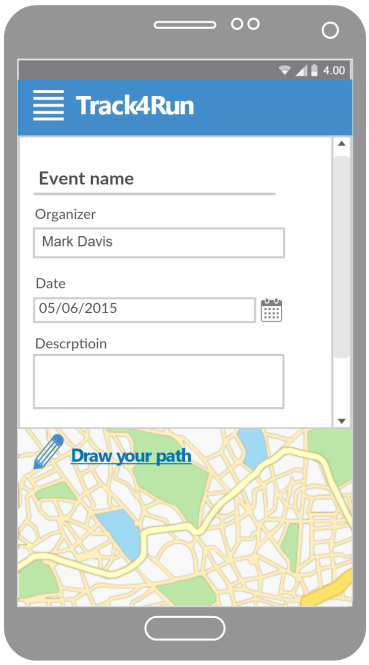

Figure 12: Mock up: Add some events.

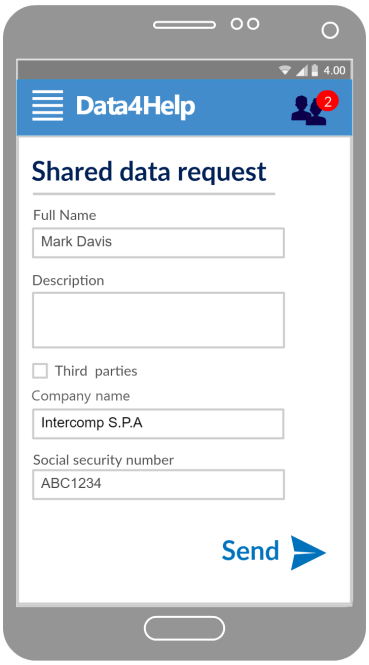

Figure 13: Mock up: Shared data request form.

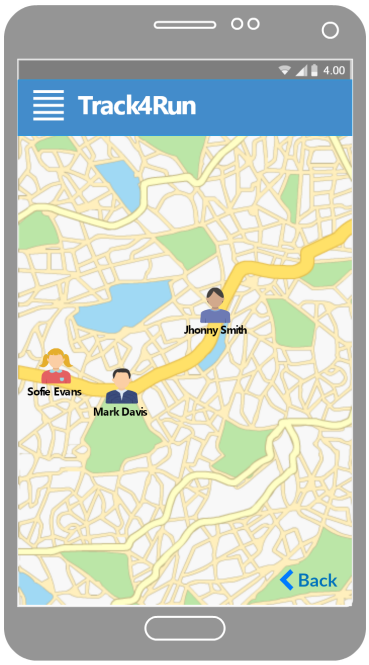

Figure 14: Mock up: Competition view.

3.1.2 Hardware interfaces

To exploit the set of functionality of TrackMe services is mandatory to have two distinct hardware components:

1. First, it is necessary to own a smartphone which is used to interact with the mobile application;

2. Second, it is also essential to have a device provided with at least one NFC sensor: this has the ability to acquire heartbeat, blood pressure and blood oxygen saturation levels data.

The latter, other than data acquisition, is also able to communicate with the smartphone in order to send the data acquired.

The smartphone must also have a GPS system to provide the position, a

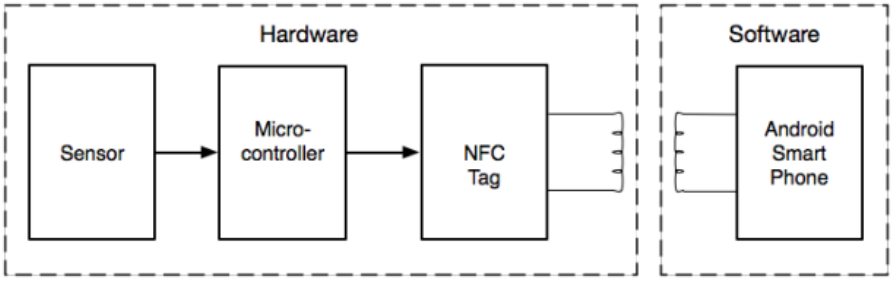

Figure 15: Example of System Architecture.

monitor to allow, during the race, to see the athletes on the path, and finally the phone must be able to call the emergency room number.

3.1.3 Software interfaces

The application uses external services to reduce the product complexity:

1. Race map
   The application requires the usage of maps to define a path for a race and to see the position of the athletes during the competition. One option here is using Google Maps API which lets customize maps with owner content and imagery.

2. Call service
   A call service will be used to perform the autonomous call: a VoIP API is here necessary. An example of this could be to use the one provided by Twilio.

3. GPS Data service
   The GPS Data as a Service API allows developers to store, handle, and manage GPS data in the system. It also analyzes GPS data in real time so that application can use it to provide the position during the call to the emergency room operator and during the running race.

4. Automated call
   Automated call service allows, to AutomatedSOS, to call the emergency number and to speak with the operator. For instance, an option to recognize the answer provided by the emergency room operator could be the Google Cloud Speech API, that enables the AutomatedSOS feature to convert audio to text by applying neural network models in an easy to use API.

5. Text-to-speech
   The Text-to-Speech API is ideal for any application that plays audio of human speech to users. It allows the application to convert arbitrary strings, words, and sentences into the sound of a person speaking the same things. For example, in this case the API is used to read a formatted text message during the call to the emergency number. The message provides all the information that usually are required by the health staff. One possible API for this interface can be Google Text-to-speech.

3.1.4 Communication interfaces

The project needs several communication interfaces:

1. A communication interface to read/write data from the NFC tag. Fortunately, this already exists and it is called NFC protocol.

2. A communication interface to send and receive data over the network, in order to interact with the TrackMe system. One possible solution is to use the protocol HTTPS, that guarantees data security during upload and download operations.

3. A communication interface to perform calls.

4. A communication interface that exploits the GPS technology

5. A communication interface between mobile application and the NFC device to transfer the collected data

3.2 Scenarios

3.2.1 Scenario 1

The company Canpari, whose business regards the production of natural and herbal medicines, needs data on the health statuses, during last year, of the inhabitants of Milan. This will allow them to evaluate the project of opening their first shop in the city.
Therefore, with their Data4Help account, the marketing office compiles and sends an aggregated request for accessing the needed information. Once the demand is received, the system analyzes it: since there are, of course, more than a thousand of user subscribed in Milan, the requests is approved and the access will be eventually granted to Canpari.

3.2.2 Scenario 2

The company Tibaldi&Zhou has set up a shocking outdoor art show that will take place in the following two months in Piazza Duomo. Therefore, it is interested in having access to data regarding the health statuses of users that visit this location when they are close to the exhibition. This is necessary in order to monitor the public feedback thanks to the usage of heart rate of possible spectators.
In order to accomplish this purpose, they subscribe, using Data4Help, to the mentioned above future data: when the two months will pass by, the system will evaluate the anonymization constraint. Since there will have been, of course, more than 1000 user that pass from Piazza Duomo, the access will be provided to the company.

3.2.3 Scenario 3

Jacob is preparing for a marathon and, while training, he always keeps with him all the equipment necessary to monitor his health. Unfortunately, yesterday he had an hearth attack while running, but his device was able to detect the health parameters below the threshold and have suddenly called the nearest hospital. Once the call was picked up, the device has communicated autonomously to the emergency room operator the observed health status and the location of the user, in order to request for an ambulance. Since, the operator understood the dangerous situation, he has accepted the demand and sent an ambulance to the location.

3.2.4 Scenario 4

The company Like wants to organize a public run in Milan and it assigns this task to Mattia, who is the chief of the public relations office.
Therefore, Mattia opens the TrackMe's application and fills the form for submitting a new run event. More specifically, he defines the name of the run, its path, the date and the time in which the run will start, and a closure date for the subscriptions. Furthermore, he also specifies 10 as the minimum number of people that needs to be present.

3.2.5 Scenario 5

Tang-Tang and his 9 friends are amateur runners that want to join a run in order to get some fun. Therefore, they decide to enroll, before the expiration date, in a run, whose minimum number of athletes is 10, organized by Abibas on the 5th of November.
When the expiration date arrives, the system checks the constraints on the number of runners. Since this constraint is satisfied, they wait for the day of the event: at this point they race and, at the end of the competition, the leaderboard is available.

3.2.6 Scenario 6

Harry, who use the TrackMe application daily, keeps receiving everyday individual requests from an Indian company and he continues to refuse.

Finally, one day, he decides to block requests from that company: he won't see any spam demand anymore and the access is, of course, not granted.

3.2.7 Scenario 7

Zhao, who is very concerned about his health status, pays a private company to analyze all of his health status data.
In order to accomplish this, he receives an individual request from them and he accepts it: after he does, the company will have eventually access to the needed information.

3.2.8 Scenario 8

Giulio is very worried about his old father, Franco, and, therefore, he convinces him to use the TrackMe application: in this way, he can send him a request to access his data, and keep the situation under control.
Franco, according to the deal, registers to the service: he provides credentials, a username, its social security number and a password. The system checks that an account associated with identification information and social security number does not exist: since this constraint is satisfied, the registration happens without any problem.

3.2.9 Scenario 9

Matteo, who is already a TrackMe user, has recently bought a new phone. Therefore, since he wants to keep using the application, he needs to perform the login.
To reach this goal, he provides his username and password: since he types both of them correctly, the access to the application is performed.

3.3 Functional requirements

The functional requirements will be listed with different way:

- Some will be called core requirements, which is necessary to satisfy more or less every goals in a simple way (e.g. registration, log-in, ...)
- Other requirements are listed together with the goal which they satisfy

3.3.1 Core requirements

The following requirements are, principally, about identifying a specific person on the system:

[R1] Allow a person to register into the system as a user by providing a username, a password, his credentials, his social security number and a consensus on the agreement

[R2] Allow a person or company to register into the system as a third party by providing an e-mail, a password, its credentials and a consensus on the agreement

[R3] A person cannot register as a user by inserting a username that has already been used in another successful registration process

[R4] A person cannot register as user by using the same social security number used in another successful registration process

- ~~[[R5]] A person cannot register as user by using the same credential used in another successful registration process~~
- ~~[R6] A person or company cannot register as third party into the system by using the same credentials used in another successful process~~

[R7] A person or company cannot register as third party by using the same e-mail used in another successful registration process

After the registration, another essential functionality of the system is to permit log-in:

[R8] Allow a person to log into the system as a user only if he has already registered as such

[R9] Allow a person or company to log into the system as a third party only if he has already registered as such

3.3.2 Goal reaching requirements

In this section, each requirement is necessary to satisfy specific goals of the system:

[G1] Allow a user to access its own data

[R10] If a user is logged-in, he is able to view its own data

[G2] Allow a user to contribute to data sharing by providing information about his location and health status

[D1] User's smartphones are equipped with GPS sensor

[D2] Users own devices with sensors to acquire data information about health status

[D3] Health data collected by the devices are correct

[D6] There is at least a messaging protocol of trusted companies which provides the possibility to transfer data from devices containing sensors to acquire health data, to smartphones

[R11] Allow a user to send its data to the system automatically when it is generated

[G3 & G4] Once the health parameters of a user have been observed below the threshold for the first time after one hour, an ambulance is sent to the user location. The time experienced between the moment in which the health parameters of a subscribed user are observed below the threshold and the time in which the emergency point is contacted is equal or less than 5 seconds

[D5] There is at least an external service of trusted companies which provides the possibility to make calls

[D8] Hospitals always accept new SOSCall.

[D9] If SOSCall are accepted, then an ambulance is sent to the location mentioned by the call

[D13] Hospitals are always reachable through help calls.

[R12] When a user's health parameters has been observed below the threshold, an SOSCall is requested within 5 seconds

[R13] All the automated SOS call are performed with devices of users whose health parameters are observed below a certain threshold

[R14] An SOSCall can be requested only every minute

[R15] An SOSCall is blocked if a previous one has already been accepted within one hour

[R16] An SOSCall is implemented as an automated call by using an external API

[R17] During an SOSCall, the GPS is set on high-precision

[G5] Allow a user to participate in a run managed by third parties, as an athlete, if all starting conditions are satisfied.

[D14] Every user enrolled in a run are able to participate in the run as athletes if all starting conditions are satisfied

[R18] Allow a user to view a list of available runs, i.e. those that are still waiting to start

[R19] Allow a user to enroll in a run, after choosing it from the list, only before the expiration date by specifying his nickname

[R20] Allow a user to unsubscribe from an enrolled run only before the expiration date

[R21] If, after the expiration date, the number of participants is less than the minimum number defined by the organizer, then it is impossible to start the run.

[R22] If the run cannot start due to minimum number of participants unsatisfied, then the enrolled runners are notified.

[G6] Allow spectators (i.e. user) to see on real-time the "correct" positions of all athletes taking part in a run, with at most 15 meters of radius error

[D4] There is at least an external service of trusted companies which provides the possibility to the user to view detailed maps

[D11] When a user's phone GPS is set on high precision, then it provides the right position with at most a radius error ranged from 0 to 10 meters

[D12] Athlete participating in a run are equipped with a device sharing GPS position set on high precision

[R23] Every athlete participating on a run, only on this specific occasion, shares continuously (i.e. each ten seconds) its position through a device

[R24] Allow a spectator to see on a map the real-time position of every athlete in a specific run

[G7] The maximum time to accept an individual request from any third party is 30 days; after that, the request will expire

[R25] Once the time elapsed from sending request is greater than 30 days, then the request will be deleted from the system

[G8 & G9] Allow a user to accept or refuse a request from third parties. Allow a user to block requests made by a specific third party and all the pending requests will be refused: this action is possible only when the user has already refused one request made by the customer that he is intending to block.

[R26] Allow a user to receive individual requests about data sharing from third parties

[R27] Allow a user to view a list of pending requests

[R28] Allow a user to accept or refuse a request from the list of pending request

[R29] Allow a user to block requests from a defined third party, after having refused a request made by the customer involved in the block.

[R46] When a request it is blocked, the other requests sent by the same Third party to the same user are refused.

[G10] Allow spectators and runners to see the leaderboard, when a run is completed

[R30] Allow an organizer to close the run (when it is terminated)

[R31] After a run is closed, the leaderboard is shown to the spectators and runners

[R32] After a day is elapsed from the date of the race, if the run is not closed the application will automatically close it.

[G11] Allow organizers (i.e. third parties) to set up a run, by defining its name, its path, date, start time, expiration date, and the minimum number of participants

[D4] There is at least an external service of trusted companies which provides the possibility to the user to view detailed maps

[D10] The service which shows the map of the world offers only paths that are feasible.

[R33] Allow a third party to see a map of the world with feasible paths

[R34] Allow a third party to publish a race by providing an unique name, a feasible and a non-overlapping path (non-overlapping with other races of the same date), a date, a start time, a brief description, an expiration date for subscription and a minimum number of participants

[R45] Allow a third party to manage its run by giving him a list of its managed race

[G12] Allow a third party to access data specified in a request if the user accepts the request or if he accepted one or more requests from the same third party that provided access to the same data

- [R35] If an individual request is accepted, then the third party who has made the request can access the data specified in the request
- [R36] For each piece of individual data accessible by a third part customer, exists an accepted request regarding it, performed by the same third party
- [R37] Allow a user to accept or refuse request given by third parties
- [R38] Allow a third party to send individual requests to users by providing the user's social security number and a brief motivation

[G13] Allow a third party to access statistical and anonymized data if and only if the number of individual involved is greater than 1000. This is satisfied after the request is approved

- [R39] A group request is accepted if the aggregated data specified in the request is accessible to the third party who performed the demand
- [R40] Group requests are accepted if and only if the number of user involved is greater than 1000
- [R41] Aggregated data is accessible to a third party if an accepted aggregated data that request that data exists
- [R42] Allow a third party to send group request to the system regarding data about many users

[G14] Allow a third party to subscribe to non-existing data. They will have access to them, after the data is generated.

- [R43] Allow a third party to express a data request on future data
- [R44] A third party can have access to non-existing aggregated data regarding future information if and only if the number of people involved will be greater than 1000

3.3.3 Use Case

Here follows the use case diagram:

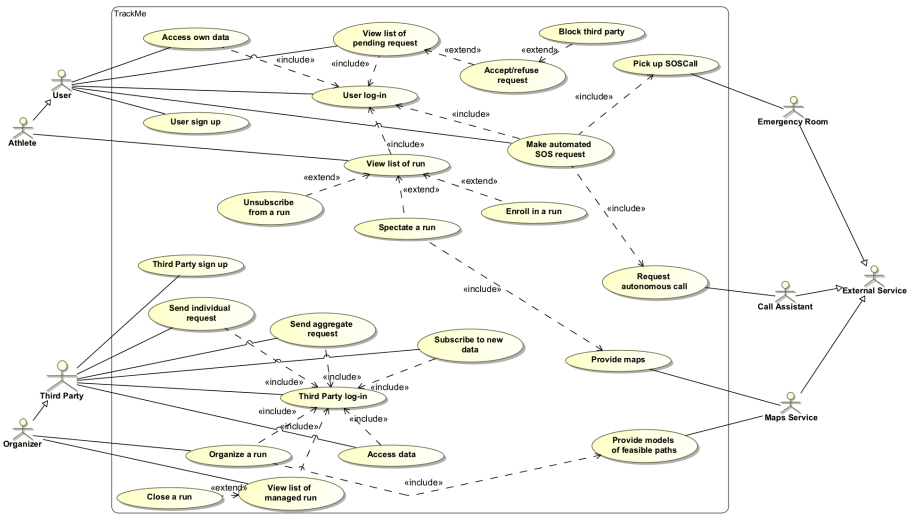

Figure 16: Use case diagram

Beneath, the use case is analyzed:

Name	Send aggregated request
Actor	Third party customer
Entry conditions	Third party customer is logged in
Event flow	1. Third party customer compiles a form specifying the aggregated data that he wants to access 2. Third party customer sends the request 3. The system analyses the request and provides to the user the access to the data
Exit conditions	The third party user can access the requested data
Exceptions	If the request involves less or equal than 1000 distinct user, the access is not provided to the third party that gets a notification

Name	Subscribe to new data
Actor	Third party customer
Entry conditions	Third party customer is logged in
Event flow	1. Third party customer compiles a form specifying the future aggregated data that he wants to access, when they will be available 2. Third party customer sends the request 3. The system waits for the moment in which all the future data requested will be generated 4. The system analyses the request and provides to the user the access to the data
Exit conditions	The third party user can access the requested data
Exceptions	If the request involves less or equal than 1000 distinct user, the access is not provided to the third party that gets a notification

Name	Make autonomous SOS call
Actor	User, emergency room operator
Entry conditions	User health parameters detected below the threshold for the first time in an hour
Event flow	1. The mobile application calls autonomously the emergency number 2. The emergency room operator picks the call up 3. The mobile applications communicates the health parameters and the user location to the emergency room operator, thus requesting for assistance 4. The emergency room operator accepts the request 5. The emergency room operator sends an ambulance to the user location
Exit conditions	An ambulance is sent to the user location
Exceptions	No exceptions

Name	Organize run
Actor	Third party customer
Entry conditions	The third party customer is logged in
Event flow	1. The third party customer sets up a run: he provides the name, the path, the date, the starting time, a closure date for the subscriptions and the minimum number of participants 2. The third party customer sends all the mentioned above information to the system 3. The system adds the run to the list of the available races
Exit conditions	The race has been successfully added to the list of the available run
Exceptions	1. The name is already used by another run 2. Another run is already been specified for the same date and two paths are overlapping All the exceptions are handled in the same way: the race is not added to the list and the third party user gets notified of the unsuccessful operation

Name	Enroll in a run
Actor	Athlete
Entry conditions	The user is logged in and there is at least a run in the list of the available race
Event flow	1. The athlete accesses the list of available run 2. The athlete selects the race in which he wants to enroll 3. The athlete sends the request for joining the run to the system 4. The system receives the requests and enroll the athlete in the race that he has specified
Exit conditions	The athlete has been successfully enrolled to the race
Exceptions	1. The athlete is already enrolled to the race that he specifies in the request 2. The athlete is already been enrolled to a race in the same date of the run specified in the request All the exceptions are handled in the same way: the athlete is notified that the enrollment was unsuccessful, by providing a brief motivation. Of course, the subscription process is aborted

Name	Block requests from a company
Actor	User
Entry conditions	The user is logged in and there is at least a request in the pending request list
Event flow	1. The user accesses the list of available requests 2. The user refuses a pending request 3. The user blocks the company that has sent the request 4. The system receives the user's decision and permanently blocks the company that has sent the request
Exit conditions	The company has been successfully blocked
Exceptions	No exceptions

Name	Accept request
Actor	User
Entry conditions	The user is logged in and there is at least a request in the pending request list
Event flow	1. The user accesses the list of available requests 2. The user selects the request which he wants to accept 3. The user accepts the request from a specific company 4. The system receives the acceptance and notifies both the third party and the user of the successful of the operation
Exit conditions	The data specified in the request is now accessible to the third party that has sent the request that has been accepted
Exceptions	No exceptions

Name	Sign up
Actor	User
Entry conditions	The user has not registered yet
Event flow	1. The user completes the registration's form and provides his information 2. The user sends the information to the system 3. The system checks that an account associated with the specified credentials does not exist 4. The system checks that an account associated with the specified social security number does not exist 5. The system checks that an account associated with the specified username does not exist 6. The system registers the user into the application
Exit conditions	The user is successfully been registered
Exceptions	The user provides non-valid data: credential or username or social security number has already been used in another successful registration process. In this case, the user is notified that the subscription is unsuccessful, by providing a brief motivation.

Name	Login
Actor	User
Entry conditions	The user has already sign up
Event flow	1. The user sends his username and password credentials through the application 2. The system checks the information provided by the user 3. The system provides access to the user to the TrackMe services
Exit conditions	The user is logged in the application
Exceptions	1. The user sends invalid username 2. The user sends invalid password All the exceptions are handled in the same way: the user is notified that the procedure was unsuccessful, by providing a brief motivation. The access to the services is not granted.

3.3.4 Sequence diagrams

To provide a better understanding of the interaction of some processes, the following diagrams are shown:

- Send individual request process

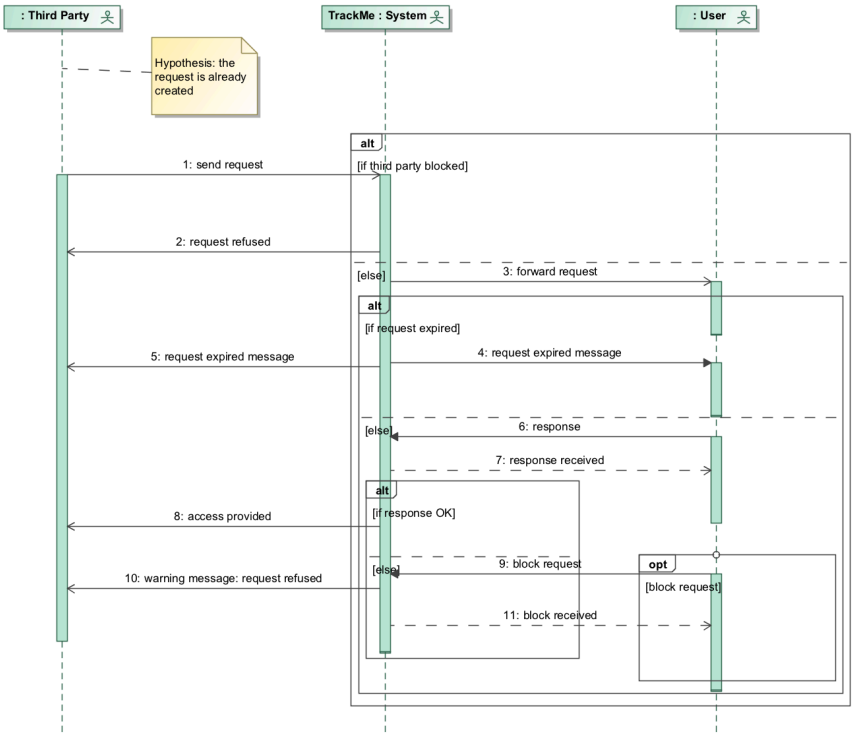

Figure 17: Sequence diagram about the process of sending a request

In this sequence diagram, it is assumed that the request is created correctly, since the most important things are the relationships with TrackMe and the other actors.

- SOSCall process: the sequence diagram is designed by assuming that the process starts when the application has detected a health parameter below threshold for the first time within one hour.

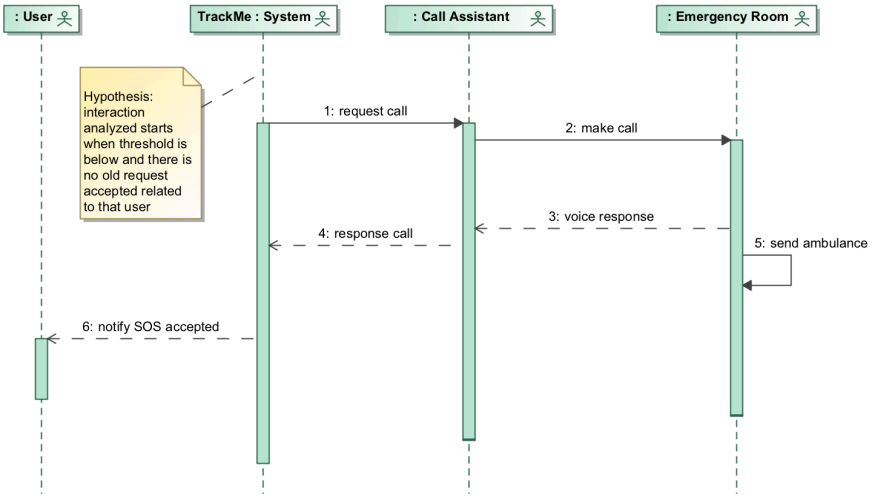

Figure 18: Sequence diagram about the process of an SOS call

- Publish race process: in this sequence diagram, it is shown how the organizer, i.e. third party, can publish a race to be seen by potential athletes that want to participate.

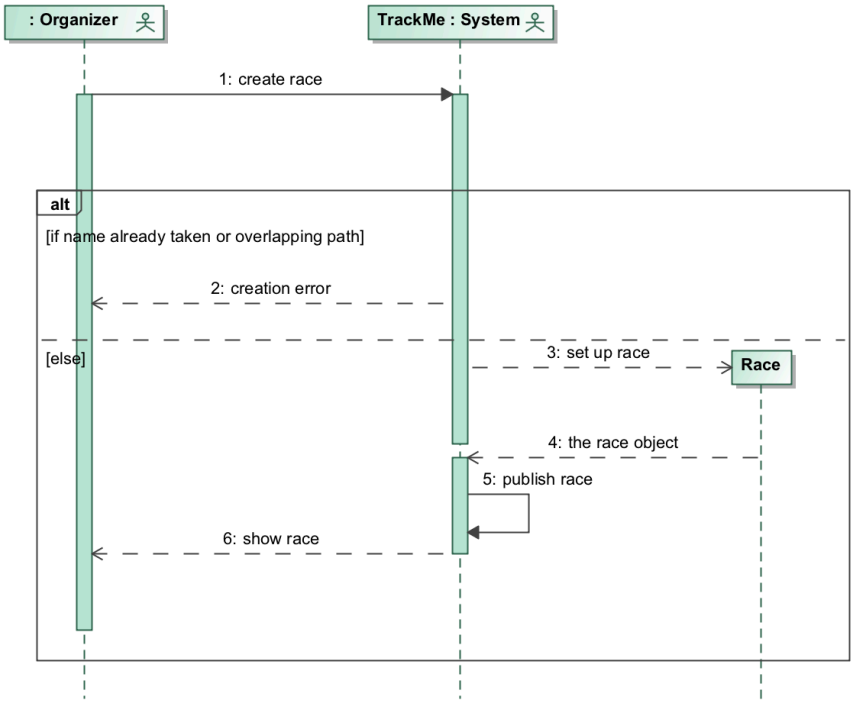

Figure 19: Sequence diagram about the process of how a race is published

3.4 Performance requirements

The system must be able to manage elevate number of users, because one of the scope is to provide access to anonymized data that involves more then 1000 users. Even if at the beginning a range of 50000-100000 users is estimated, the system has to be flexible and support scalability.
Furthermore, the AutomatedSOS service is critical for the life of people, therefore, the health status and user location detection have to be performed in a fast way: the time experienced between the observation of health status parameter below the threshold and the dispatch of an emergency call, has to be less or equal than 5 seconds.

3.5 Design constraints

3.5.1 Standards compliance

The privacy is one of the most important characteristic which the system has to implement. The agreement, shown at sign up operation for both user and third party, must be written in a way that are in compliance with the law. Of course, it also has to protect the privacy of people and what the application does, must not violate what is written in the agreement. Furthermore, the APIs, used to implement the functionality about maps, text-to-speech, voice recognition and so on, must respect the license defined by the external services. For instance, using an API which is open source is completely compliant with the law.

3.5.2 Hardware limitations

The hardware limitations of the application are strict for the user. Infact, he needs two devices, and the second one is not so wide-spread:

- A smartphone, for being connected to the TrackMe system. This allow him to accept or refuse requests from third party customer, enroll in available runs and spectate races. The smarthphone required, needs an Internet connection, used for communicating with the system.
  Furthermore, it needs GPS and NFC sensors.
- A device provided of at least a NFC sensor: this will collect health status of the user and will communicate with the smarthphone. The data detection and the communication must be performed in a fast way (see performance requirements)

For what concerns the hardware limitations of the third party customers, they are not strict at all. They only needs a smartphone that can access to the Internet in order to send requests and access information.

3.6 Software system attributes

3.6.1 Reliability

The application has to work properly even in difficult condition such as bad requests from clients or when the system is under huge work load. In particular, the reliability of the AutomatedSOS service should be very high, because it could

be a matter of life and death. Data4Help and Track4Run, have more concessions in these terms.

3.6.2 Availability

The automatedSOS service should be available 99.99% of the time, due to the already mentioned reasons. This availability takes into consideration what the system can guarantee; basically problems such as user's connection or device's issues are excluded. Instead, Data4Help and Track4Run have to be available 99% of the time, because they are not considered critical features.

3.6.3 Security

The health statuses and the locations of the users are communicated toward the internet connection encrypted: since the communication toward the central system is not so critical, security constraints are considered more important than the speed of the communication (same for the communication between third party customers). Moreover, the communication between the device that collects the health statuses and the smartphone, is also encrypted to avoid malevolent action from other people.
Private data, such as account information, is stored also in the smartphone with high-security encryption. Collected data that has already been sent to the system, is not saved into the smartphone.
Data access is provided to third party customers with an high-secure communication protocol.

3.6.4 Maintainability

Since the application is very complex to implement, the system, in the future, will be maintained continuously. To ease this process, for instance, it is possible to select external services which are most trusted with respect to reliability and maintainability. Another essential feature to take in consideration is the reusability of the code, which is highly used. For instance, using an approach top-down during the design process could help or using some design patterns to facilitate code design.

3.6.5 Portability

The application is designed to reach more people as possible. Therefore, the application should be portable in the main smartphone operating systems; to achieve this goal it should be necessary to implement the system to be as portable as possible. For example writing code which is easy to traduce can help.

4 Formal Analysis using Alloy

In the alloy model, in order to be safer w.r.t. the requirements that have been stated in this document, critical aspects have been modeled. In particular, the following vital goals have been asserted:

[G3] Once the health parameters of a user have been observed below the threshold for the first time after one hour, an ambulance is sent to the user location

[G12] Allow a third party to access data specified in a request if the user accepts the request or if he accepted one or more requests from the same third party that provided access to the same data

[G13] Allow a third party to access statistical and anonymized data if and only if the number of individual involved is greater than 1000. This is satisfied after the request is approved

Note on the alloy model:

- Some trivial domain assumptions are written in order to prove the goals. However, they are not stated in the domain assumption chapter of this document
- The context of G3 has been modeled considering that a user uses the AutomatedSOS service at most once in an hour. This allows to prove the goals, without loss of generality
- For G13, the fact that the aggregated request is satisfied after it is approved, has not been proven: the part considered critical is the access to the demanded information
- The constraint on the number of people involved in an aggregated request is modified without loss of generality: the group dimension has to be greater than 5

```
/*
 title: TrackMe's Alloy
 date: 2018/19
 description: It defines the world and shared phenomena of the system
 author: Riccardo Poiani, Mattia Tibaldi, Tang-Tang Zhou
*/

/* Useful data structure */
abstract sig Bool{}
one sig True extends Bool{}
one sig False extends Bool{}

abstract sig RequestStatus{}
one sig RequestInProgress extends RequestStatus{}
one sig RequestAccepted extends RequestStatus{}
one sig RequestRefused extends RequestStatus{}
one sig RequestExpired extends RequestStatus{}

abstract sig CallStatus{}
one sig CallAccepted extends CallStatus{}

/* Actors */
abstract sig Actor{}

sig User extends Actor{
```

```
    userData: set Data,
    ambulancesProvided: set Ambulance
}

sig ThirdParty extends Actor{
    accessibleData: set Data,
    accessibleAggregatedData: set AggregatedData,
    requests: set Request
}

sig EmergencyRoom{
    ambulances: set Ambulance
}

/* Objects */
abstract sig Data{
}

sig LocationData extends Data{
}

sig HealthData extends Data{
    belowThreshold : one Bool
}

sig AggregatedData{
    regardingData: some Data
}

abstract sig Request{
    status: one RequestStatus
}

sig GroupRequest extends Request{
    aggregatedData: one AggregatedData
}

sig IndividualRequest extends Request{
    user: one User,
    requestedData: some Data
} {
    requestedData in user.userData
}

sig SOSCall{
    status: one CallStatus,
    destinatedTo: one EmergencyRoom,
    performedBy: one User
}

sig Ambulance{

}

/* Domain Assumptions */
fact DataBelongOnlyToOneUser {
    all d: Data | some u : User | d in u.userData and (no u1 : User | u ≠ u1
        ↪ and d in u1.userData)
}

fact RequestBelongOnlyToOneThirdParty{
    all r : Request | some tp : ThirdParty | r in tp.requests and (no tp1:
        ↪ ThirdParty | tp ≠ tp1 and r in tp1.requests)
}

fact AggregatedDataBelongAlwaysToSomeGroupRequest {
    all ad: AggregatedData | some gr : GroupRequest | gr.aggregatedData = ad
}

fact ThereIsAtLeastAnSOSExternalServiceWithAmbulance{
    some er: EmergencyRoom | #er.ambulances > 0
}

fact EmergencyRoomAcceptsNewSOSCall{
```

```
    //In this model, a SOSCall is performed only if it has not already
        ↪ performed in the last hour
    all sc: SOSCall | some er: EmergencyRoom | #er.ambulances > 0  and sc.
        ↪ destinatedTo = er and sc.status = CallAccepted
}

fact IfSOSCallAreAcceptedThanAnAmbulanceIsSent{
    all a: Ambulance |some u: User | a in u.ambulancesProvided iff (some sc:
        ↪ SOSCall | sc.status = CallAccepted and sc.performedBy = u)
}

fact ambulanceBelongToAtLeastAnEmergencyRoom{
    all a: Ambulance | some er: EmergencyRoom | a in er.ambulances
}

fact ForAllSOSCallAreProvidedOnlyOneAmbulance{
    all sc: SOSCall | #sc.performedBy.ambulancesProvided = 1
}

/* Help Function and Predicate */
fun getPeopleInvolved [d: set Data] : set User{
    userData.d
}

/* Requirements */

// Requirements for G3

/*
 * R12
 * When a user's health parameters has been observed below the threshold, an
     ↪ SOSCall is requested within 5 seconds
 */
pred parametersHasBeenBelowThenAnSOSCallIsRequested {
    all hd: HealthData |one sc: SOSCall | some u: User | sc.performedBy = u
        ↪ and  hd in u.userData and hd.belowThreshold = True
}

/*
 * R13
 * All the automated SOS call are performed with devices of users whose
     ↪ health parameters are observed below a certain threshold
 */
pred AllSOSCallArePerformedByUserWithDataBelowTheThreshold{
    all sc: SOSCall | some hd: HealthData | hd in sc.performedBy.userData and
        ↪  hd.belowThreshold = True
}

// Requirements for G12

/*
 * R35
 * If an individual request is accepted, then the third party who has made
     ↪ the request can access the data specified in the request
 */
pred individualRequestAcceptedIfDataAccessible {
    all r : IndividualRequest, tp : ThirdParty | r in tp.requests and r.
        ↪ status = RequestAccepted implies {
         r.requestedData in tp.accessibleData
    }
}


/*
 * R36
 * For each piece of individual data accessible by a third part customer,
     ↪ exists an accepted request regarding it, performed by the same third
     ↪  party
 */
pred individualDataAccessibleIfAnAcceptedRequestExist {
    all tp : ThirdParty,  d : Data | d in tp.accessibleData implies {
        some r : IndividualRequest | r in tp.requests and r.status =
            ↪ RequestAccepted and r.requestedData in d
    }
```

```
}

// Requirements for G13

/*
 * R39
 * A group request is accepted if the aggregated data specified in the
     ↪ request is accessible to the third party who performed the demand
 */
pred groupRequestAcceptedIfAggregatedDataAccessible {
    all r : GroupRequest, tp : ThirdParty | r in tp.requests and r.status =
        ↪ RequestAccepted implies {
        r.aggregatedData in tp.accessibleAggregatedData
    }
}

/*
 * R40
 * Group requests is accepted if and only if the number of user involved is
     ↪ greater than 1000
 *
 * Note: the number of people involved in the request must be greater than
     ↪ 1000 to be accepted by
 *          the system (here 1000 has been decreased for simplicity)
 */
pred groupRequestNumberOfPeopleInvolved {
    all gr : GroupRequest | #(getPeopleInvolved[gr.aggregatedData.
        ↪ regardingData]) > 5 implies {gr.status = RequestAccepted} else {
        ↪ gr.status = RequestRefused}
}


/*
 * R41
 * Aggregated data is accessible to a third party if an accepted aggregated
     ↪ data that request that data exists
 */
pred groupDataAccessibleIfAcceptedRequestExist {
    all tp : ThirdParty, d : AggregatedData | d in tp.
        ↪ accessibleAggregatedData implies {
        some r : GroupRequest | r in tp.requests and r.status =
            ↪ RequestAccepted and r.aggregatedData in d
    }
}


/*
 * Goals
 */

/*
 * G3
 * Once the health parameters of a user have been observed
 * below the threshold for the first time after one hour, an ambulance is
     ↪ sent to the user location.
 */
assert ambulanceIsProvidedAfterASOSCall {
    parametersHasBeenBelowThenAnSOSCallIsRequested  and
        ↪ AllSOSCallArePerformedByUserWithDataBelowTheThreshold implies{
        all hd: HealthData | some u: User | hd in u.userData and hd.
            ↪ belowThreshold = True implies{
            some a: Ambulance | a in u.ambulancesProvided
        }
    }
}

/*
 * G12:
 * Allow a third party to access data specified in a request if the user
     ↪ accepts the request or if he accepted one or more requests
 * from the same third party that provided access to the same data
 */
assert correctAccessToIndividualData {
    individualDataAccessibleIfAnAcceptedRequestExist  and
```

```
        ↪ individualRequestAcceptedIfDataAccessible implies {
        all r : IndividualRequest, tp : ThirdParty | r in tp.requests implies
            ↪  {
            r.status = RequestAccepted implies {r.requestedData in tp.
                ↪ accessibleData }
            and
            (r.status ≠ RequestAccepted and  no r2 : IndividualRequest | r2
                ↪ in tp.requests and r2.status = RequestAccepted and  some
                ↪ (r2.requestedData & r.requestedData))  implies {
            (r.requestedData not in tp.accessibleData)
            }
        }
    }
}

/*
 * G13:
 * Allow a third party to access statistical and anonymized data if and only
     ↪ if the number of individual involved is greater than 1000.
 * This is satisfied after the request is approved
 */
assert correctAccessToGroupData {
    groupRequestAcceptedIfAggregatedDataAccessible and
        ↪ groupRequestNumberOfPeopleInvolved and
        ↪ groupDataAccessibleIfAcceptedRequestExist implies {
        all r : GroupRequest, tp : ThirdParty | r in tp.requests implies {
            #(getPeopleInvolved[r.aggregatedData.regardingData]) >5 iff r.
                ↪ aggregatedData in tp.accessibleAggregatedData
                ↪
        }
    }
}

/* Show World Predicate */
pred showData4HelpWorld{
    individualRequestAcceptedIfDataAccessible and
        ↪ individualDataAccessibleIfAnAcceptedRequestExist and
        ↪ groupRequestNumberOfPeopleInvolved
    and groupRequestAcceptedIfAggregatedDataAccessible and
        ↪ groupDataAccessibleIfAcceptedRequestExist
}

pred showAutomatedSOSWorld{
    parametersHasBeenBelowThenAnSOSCallIsRequested and
        ↪ AllSOSCallArePerformedByUserWithDataBelowTheThreshold
}

check correctAccessToGroupData for 10
check correctAccessToIndividualData for 10
check ambulanceIsProvidedAfterASOSCall  for 10
run showAutomatedSOSWorld for 5 but 0 Request
run showData4HelpWorld for 5 but 1 EmergencyRoom, 1 Ambulance, 0 SOSCall
```

To give a better understanding of the world environment, the following images are shown:

- Data4Help world: it defines better how the world of the service Data4Help works

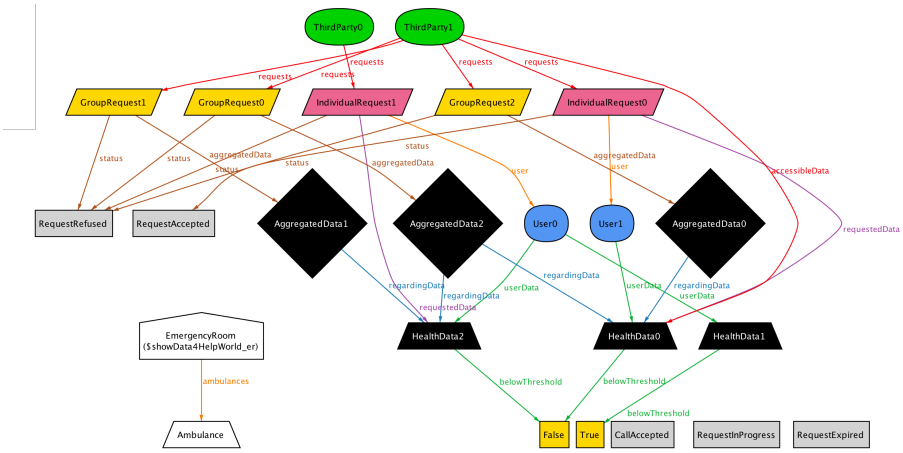

Figure 20: Data4Help World

- AutomatedSOS world: it defines better how the world of the service AutomatedSOS works

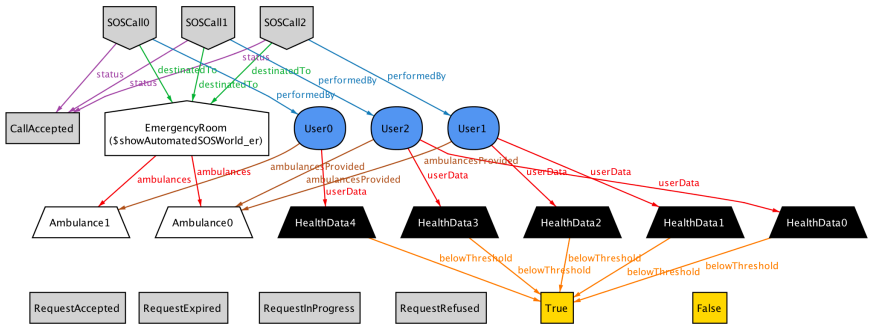

Figure 21: AutomatedSOS World

4.1 Alloy results

These are the results of the alloy model.

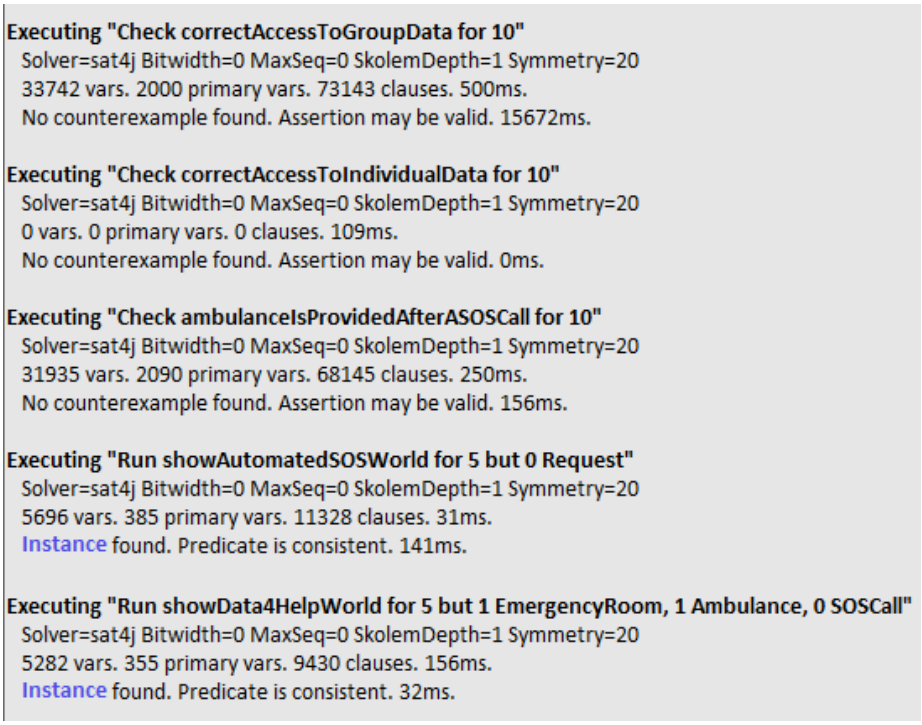

Figure 22: Alloy results

5 Effort Spent

5.1 Riccardo Poiani

Date	Task	Hours
15/10	Goal definitions	2.5
17/10	Document structure, goal definition, class diagram	2.5
18/10	state diagram, class diagram, purpose, hypothesis	1.5
19/10	Scope, purpose and state diagram	2
22/10	Product functions	2
23/10	Scenarios	2
24/10	Scenarios	1
25/10	Refactor all document (revising)	1
26/10	Refactor all document (revising)	4
27/10	Add design standards, performance, availability, reliability, security; Add use case diagram and sequence diagram	5
27/10	Alloy	2
28/10	Alloy; world and shared phenomena; use case; whole document revision; sequence diagram	7.5
29/10	Alloy; sequence diagram; fix requirement	3
01/11	Revise document	4
09/12	Document version 1.1	3
	Overall	43

5.2 Mattia Tibaldi

Date	Task	Hours
15/10	Goal definitions	1.5
17/10	Introduction	2
18/10	Scope and purpose	4
19/10	State diagram	1
22/10	Hardware interfaces	2
23/10	Software interfaces	3.5
24/10	User interfaces	3
25/10	User interfaces and communication interfaces	4
27/10	Revise document and use case	3.5
28/10	Alloy	4.5
29/10	Alloy and refactor document	4
01/11	Revise document	4
09/12	Document version 1.1	2
	Overall	39

5.3 Tang-Tang Zhou

Date	Task	Hours
15/10	Goal definitions	2.5
17/10	Document structure, goal definition, class diagram	2.5
18/10	state diagram, class diagram, purpose, hypothesis	1.5
20/10	Product Perspective, race diagram, request diagram and Alloy	4
21/10	Alloy	1.5
22/10	Fix goals, purpose and perspective	2
23/10	Domain asssumptions and requirements	2
24/10	Requirements, goals, assumptions	2
25/10	Refactor all document (revising)	2
26/10	Refactor all document (revising)	3
27/10	Fix requirements, add alloy, add use case and sequence diagram	5
27/10	Add mantainability, portability, definitions, structure and references	2
28/10	Revise document, fix diagrams, add sequence diagram to document, add effort spent	5
29/10	Fix diagrams, Add alloy worlds, fix requirements	2
31/10	sequence diagram	1
01/11	Revise document	3
24/11	Document version 1.1	3
09/12	Document version 1.1	3
	Overall	47