



# POLITECNICO MILANO 1863

TrackMe  
Software Engineering 2 Project  
*RASD Document*

Stefano Martina, Alessandro Nichelini, Francesco Peressini

A.Y. 2018/2019

November 10, 2018

# Contents

<b>1</b>	<b>Introduction</b>	<b>4</b>
1.1	Purpose . . . . .	4
1.2	Scope . . . . .	5
1.2.1	Description of the problem . . . . .	5
1.2.2	World Phenomena . . . . .	5
1.2.3	Machine Phenomena . . . . .	5
1.2.4	Shared Phenomena . . . . .	5
1.2.5	Goals . . . . .	5
1.3	Definitions, Acronyms, Abbreviations . . . . .	7
1.3.1	Definitions . . . . .	7
1.3.2	Acronyms . . . . .	7
1.3.3	Abbreviations . . . . .	7
1.4	Revision history . . . . .	7
1.5	Document Structure . . . . .	7
<b>2</b>	<b>Overall description</b>	<b>8</b>
2.1	Product perspective . . . . .	8
2.1.1	Class Diagram . . . . .	8
2.1.2	State Diagram . . . . .	9
2.2	Product functions . . . . .	11
2.3	Users characteristics . . . . .	12
2.4	Assumptions, dependencies and constraints . . . . .	12
2.4.1	Domain assumptions . . . . .	12
<b>3</b>	<b>Specific Requirements</b>	<b>13</b>
3.1	External Interface Requirements . . . . .	13
3.1.1	User Interfaces . . . . .	13
3.1.2	Hardware Interfaces . . . . .	21
3.1.3	Software Interfaces . . . . .	21
3.1.4	Communication Interfaces . . . . .	21
3.2	Functional Requirements . . . . .	22
3.2.1	Definitions of use cases . . . . .	23
3.2.2	Use case diagram . . . . .	28
3.3	Sequence diagrams . . . . .	29
3.4	Performance Requirements . . . . .	32
3.5	Design Constraints . . . . .	32
3.5.1	Standards compliance . . . . .	32
3.5.2	Hardware limitations . . . . .	32
3.5.3	Any other constraint . . . . .	32
3.6	Software System Attributes . . . . .	33
3.6.1	Reliability . . . . .	33
3.6.2	Availability . . . . .	33
3.6.3	Security . . . . .	33
3.6.4	Maintainability . . . . .	33

3.6.5	Portability . . . . .	33
3.7	Scenarios . . . . .	34
3.7.1	Scenario 1 . . . . .	34
3.7.2	Scenario 2 . . . . .	34
3.7.3	Scenario 3 . . . . .	34
<b>4</b>	<b>Formal analysis using Alloy</b>	<b>35</b>
4.1	World Generated . . . . .	40
<b>5</b>	<b>Effort spent</b>	<b>41</b>
<b>6</b>	<b>References</b>	<b>41</b>

# 1 Introduction

## 1.1 Purpose

TrackMe is a company aiming to support interactions between users, who like keeping track of their health status and activities, and third-parties, which can use data and enhance their value.

TrackMe app is composed of a core service:

- Data4Help: which provides basic support for health and location data retrieving and sharing.

and two advanced services which relay on the first one.

- AutomatedSOS: add support for SOS services to elderly people;
- Track4Run: add support for running events.

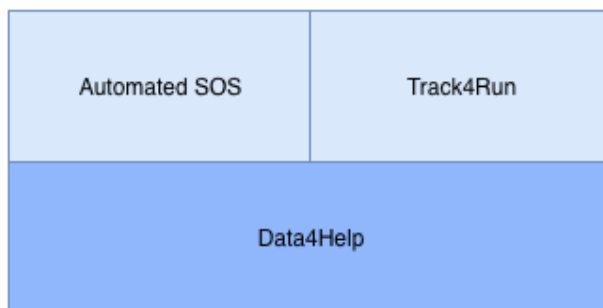


Figure 1: Services abstract structure.

The project consists of a platform divided in a mobile application and a web-based interface to serve third-parties.

The system allows individual users to add and handle data on the app and to third-parties to request and have access to these data.

In particular, a user of the application is able to keep track of his/her position during the activities but also during the normal day-life. The users' data are used to monitor the heartbeat and eventually to send an AutomatedSOS. Furthermore, TrackMe, is able to track all the athletes (that are using the application), during a run previously organised. Indeed TrackMe's app provides a section to setup a group run, specifying the path and other useful information.

## 1.2 Scope

### 1.2.1 Description of the problem

Nowadays a lot of people track their activities with smartphone or wearable devices. For this reason TrackMe provides a new complete user experience allowing all the users to read briefly the information about all their activities history. It also provides a service to organise a group run, during which it's possible to monitor all the athletes information. Furthermore all the users are monitored and, in case of some trouble, an SOS will be launched .

### 1.2.2 World Phenomena

- *General user's health condition*: the machine doesn't know the information about possible user's disease.
- *First aid services status*: the machine doesn't know the actual first aid services status.
- *Overall third parties knowledge status*: the machine doesn't know which informations third parties already have about users.

### 1.2.3 Machine Phenomena

- *Third-parties registration*
- *User's registration*
- *Data anonymisation*

### 1.2.4 Shared Phenomena

- *Vital parameters*: the machine can read vital parameters of the user such as BPM
- *User's location*: the machine knows or can read actual and past user's location

### 1.2.5 Goals

- [G1] Users can be recognised by their credentials.
- [G2] Allow users to keep track of their health data.
- [G3] Allow users to have access to an overview of their data, including health parameters and performed activities.
- [G4] Allow users to manage their data access policy.
- [G5] Allow users to monitor their performance during run workouts.

- [G6] Each time vital signs go below a threshold value, first aid services have to be notified.
- [G7] Allow users to organise running events.
  - [G7.1] Allow users to create running events.
  - [G7.2] Allow users to en-roll to events.
  - [G7.3] Allow spectators to follow participants' live position during events.
- [G8] Allow third parties to access data:
  - [G8.1] Allow third parties to require access to specific user data.
  - [G8.2] Allow third parties to retrieve anonymised aggregated data.
  - [G8.3] Allow third parties to subscribe to data feed

## 1.3 Definitions, Acronyms, Abbreviations

### 1.3.1 Definitions

- **Event:** An event organised by a user.
- **Notification:** A warning that advise the user of a request by third parties.

### 1.3.2 Acronyms

- API: Application Programming Interface;
- ASOS: AutomatedSOS;
- BPM: Beats Per Minutes;
- D4H: Data4Help;
- T4R: Track4Run.

### 1.3.3 Abbreviations

- $[Gn]$ : n-th goal
- $[Dn]$ : n-th domain assumption
- $[Rn]$ : n-th functional requirement

## 1.4 Revision history

- 1.0.0 Initial version (10/11/2018)

## 1.5 Document Structure

This document essentially follows the structure defined by IEEE for RASD document (*ISO/IEC/IEEE 29148 dated Dec 2011*).

## 2 Overall description

### 2.1 Product perspective

#### 2.1.1 Class Diagram

The product perspective of TrackMe is now explained using the UML paradigms. The main classes of the UML are:

- **User** contains all the information about a user like username, password, fiscalCode, hometown, elderly, runners.
- **ThirdParties** contains all the information about the registered organisation.
- **Data** is an abstract class that has two concrete classes: **Health data** and **Location data**
- **Running Event**
- **Location** contains information like the start point and the end point of a running event.

A user creates an Event specifying the path ie the starting location, the ending location and some specific interesting point along the path. After the creation some user can join in. An organisation, as explained above, can subscribe to a feed of data or directly to a single user.

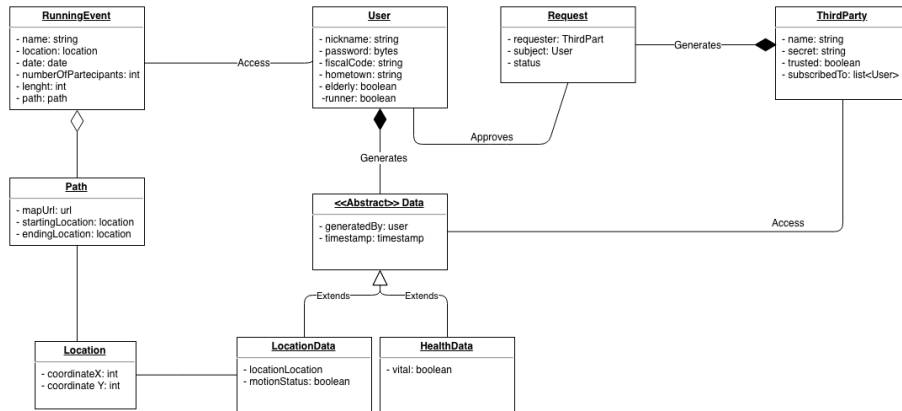


Figure 2: UML diagrams.



### 2.1.2 State Diagram

The whole system can be also seen as a set of different services relying on the main module Data4Help. Here we describe the main aspects of them with state diagrams.

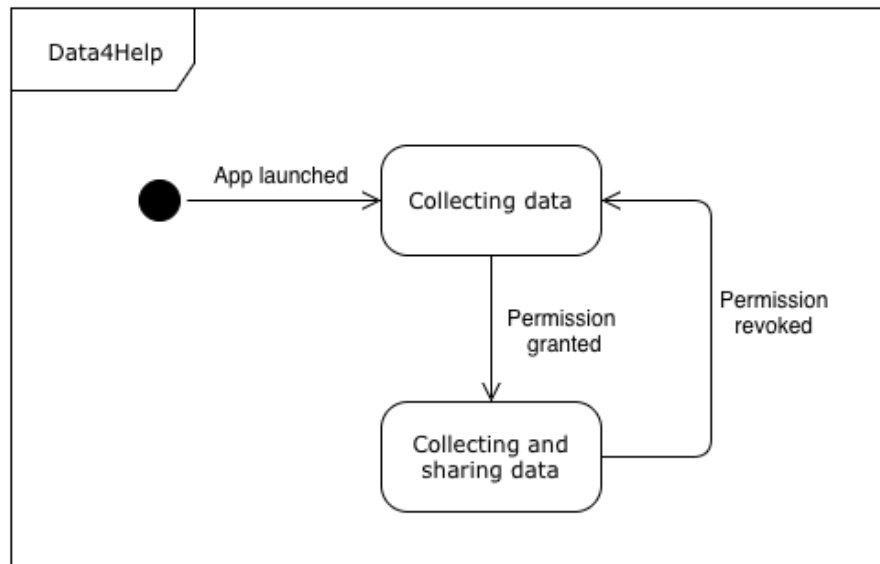


Figure 3: Basic Data4Help service state diagram.



Figure 4: Automated SOS state diagram.

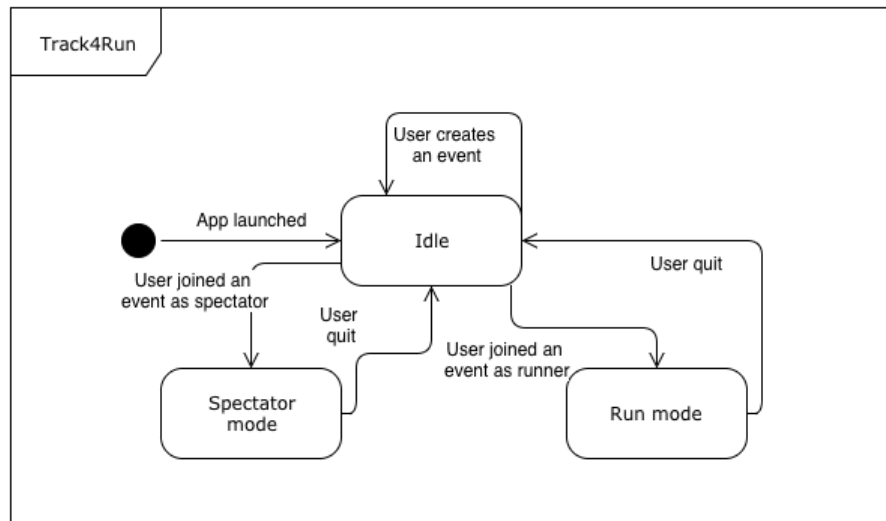


Figure 5: Track4Run state diagram

## 2.2 Product functions

The software developed by TrackMe include three different services: Data4Help supports user's data acquisition through wearable devices, AutomatedSOS, a personalised and non-intrusive SOS service for elderly people and Track4Run, a service to track athletes participating to a run. Data4Help is also a service available for third-parties: in fact, organisations can request data to TrackMe and collect them for pursuing their objectives. Data acquisition can be performed in two different way: directly to a single user or to a groups of individuals. Concerning the single user case, companies, through a social security code, ask permission to access the corresponding information. In the other case, organisations can request, directly to TrackMe, data of group of individuals with particular properties (e.g. users between 20 and 30 years old or living in a certain district). In the latter, TrackMe provides data only if it can anonymised them correctly. AutomatedSOS it's great for those elderly people who want to monitor their health status and have the support of an ambulance in case their vital parameters are under certain thresholds. Track4Run it's a service dedicated to the runners: it offers the possibility to organise a run between other runners, define the path and the duration, allowing the invited ones to en-roll. In addition, Track4Run users who do not participate to the run can see live time on a map the position of all the runners. Finally, Track4Run supports the synchronisation of data with Data4Help.

All the goals presented in section 1.2.5 are going to be implemented. Here we describe deeply the requirements needed to implement application's functions.

- [R1] Users can create an account with credentials.
- [R2] Credentials can be retrievable also if forgotten/lost.
- [R3] Users can log manually or automatically their data.
- [R4] Users have to be able to accept/deny access to single data access request.
- [R5] Users have to be able to see current data policies and to change them.
- [R6] The machine has to be able to read health and position data.
- [R7] The machine has to be able to recognise below threshold parameters.
- [R8] The machine has to be able to communicate with third parties.
- [R9] The machine has to be able to recognise data fragmentation level.
- [R10] The machine has to be able to store users' data.

## 2.3 Users characteristics

**Users:** any person that will use the service to keep track of health data, positioning or automated SOS during any kind of activities (walking or running during a lone run or an organised event).

**Third-parties:** Organisation or other parties interested in the acquisition for their usage of the data provided by the user activities.

## 2.4 Assumptions, dependencies and constraints

### 2.4.1 Domain assumptions

[D1] Data manually inserted by users are to be considered truthful

[D1.1] Data are up to date.

[D1.2] Data are correctly formatted.

[D2] First aid services are ready to handle emergency notifications.

[D3] Running paths proposed by users are well formed (e.g: legit by law).

[D4] External services are reliable (map services).

[D5] Users can be localised through GPS:

[D5.1] GPS service is reliable.

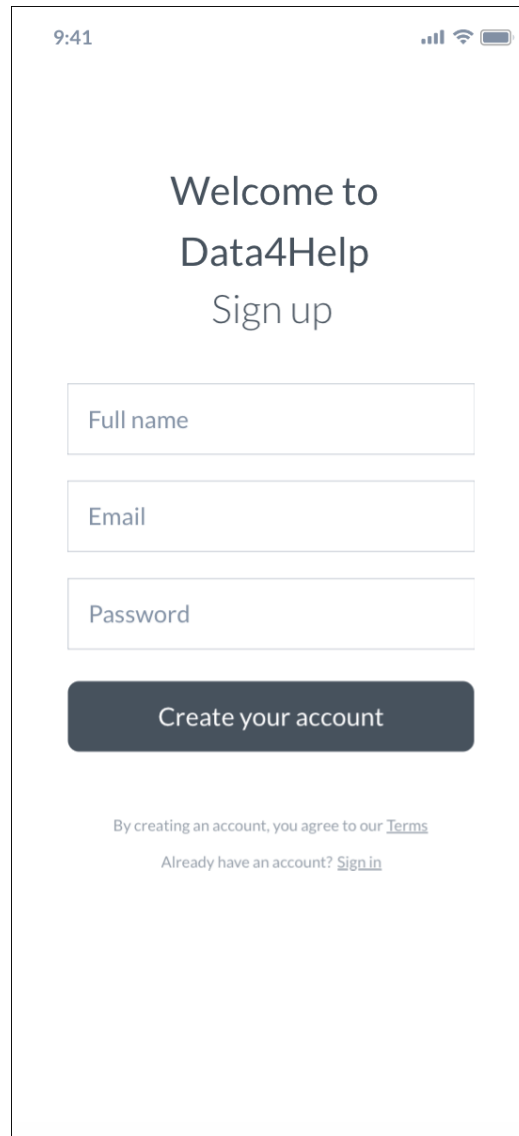
[D5.2] GPS service is accurate enough to track runners during workouts.

[D6] Users are not affected with diseases that can influence heartbeat ratio.

## 3 Specific Requirements

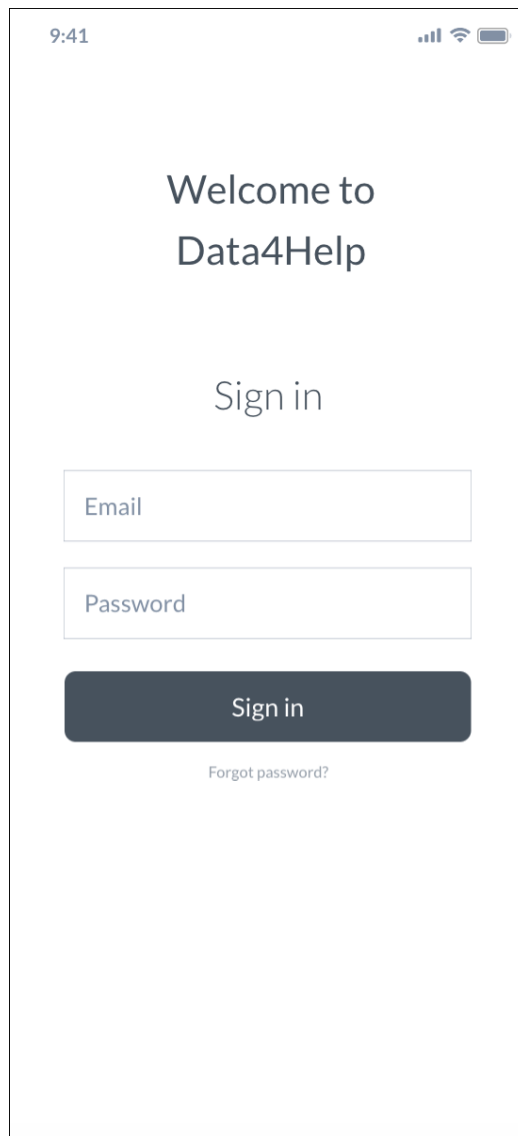
### 3.1 External Interface Requirements

#### 3.1.1 User Interfaces



A mobile application interface for signing up. The screen is white with a dark grey header bar at the top. The header bar contains the time '9:41' on the left and signal, Wi-Fi, and battery icons on the right. The main content area is centered and contains the text 'Welcome to Data4Help' in a large, bold, dark grey font, followed by 'Sign up' in a smaller, regular dark grey font. Below the text are three stacked input fields with light grey borders and placeholder text: 'Full name', 'Email', and 'Password'. Below these fields is a dark grey button with rounded corners and the text 'Create your account' in white. At the bottom, there is a line of small text: 'By creating an account, you agree to our [Terms](#)'. Below that is another line of small text: 'Already have an account? [Sign in](#)'.

Figure 6: Sign Up



9:41

Signal strength, Wi-Fi, and battery status icons

# Welcome to Data4Help

Sign in

Email

Password

Sign in

[Forgot password?](#)

This is a mobile app sign-in screen. At the top, the status bar shows the time as 9:41 and icons for signal strength, Wi-Fi, and battery. The main heading is 'Welcome to Data4Help'. Below this is a 'Sign in' label. There are two input fields: 'Email' and 'Password'. A dark blue 'Sign in' button is positioned below the password field. At the bottom, there is a link for 'Forgot password?'.

Figure 7: Sign In

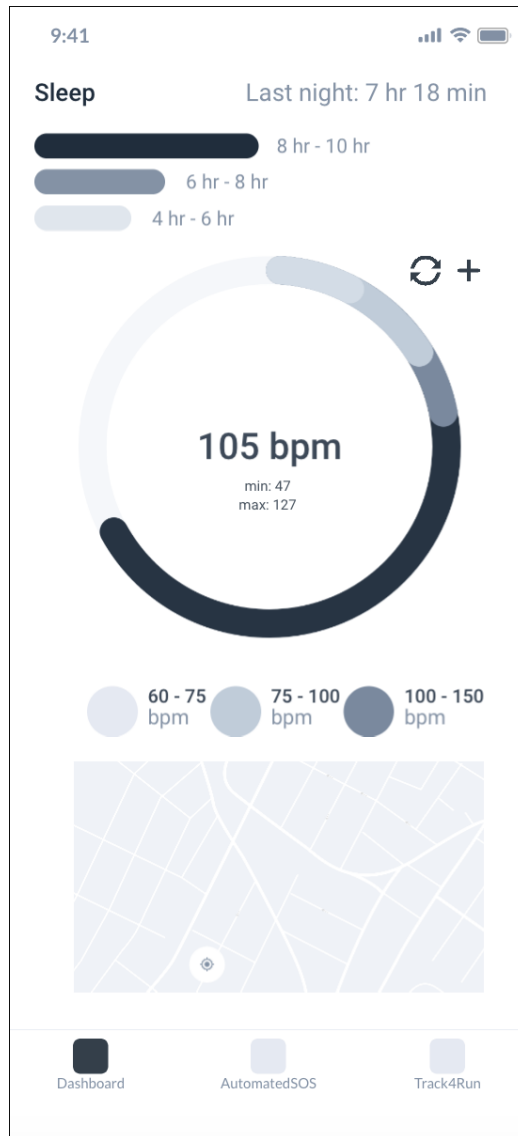


Figure 8: Data4Help dashboard

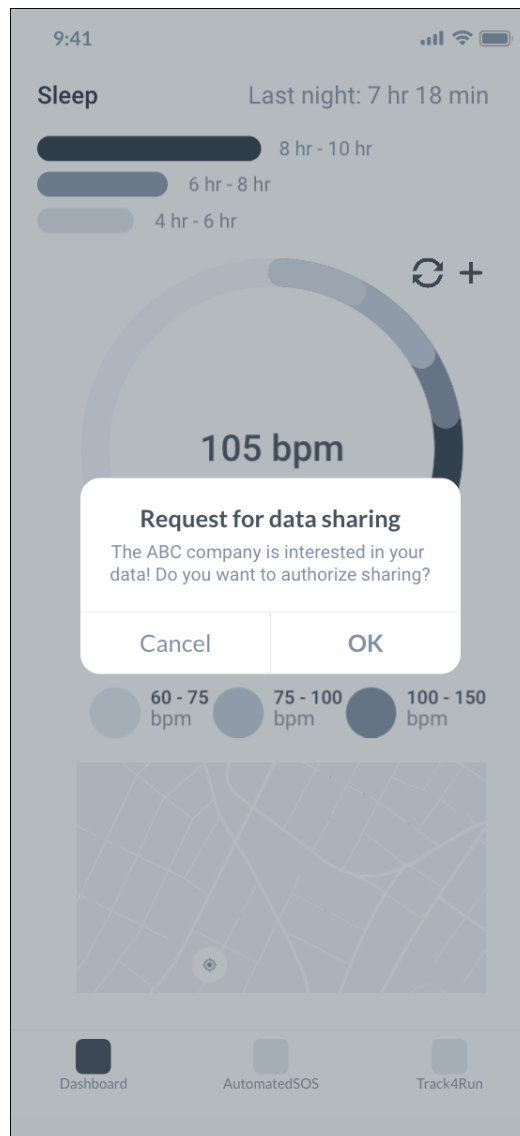


Figure 9: Request for data sharing





Figure 10: AutomatedSOS dashboard

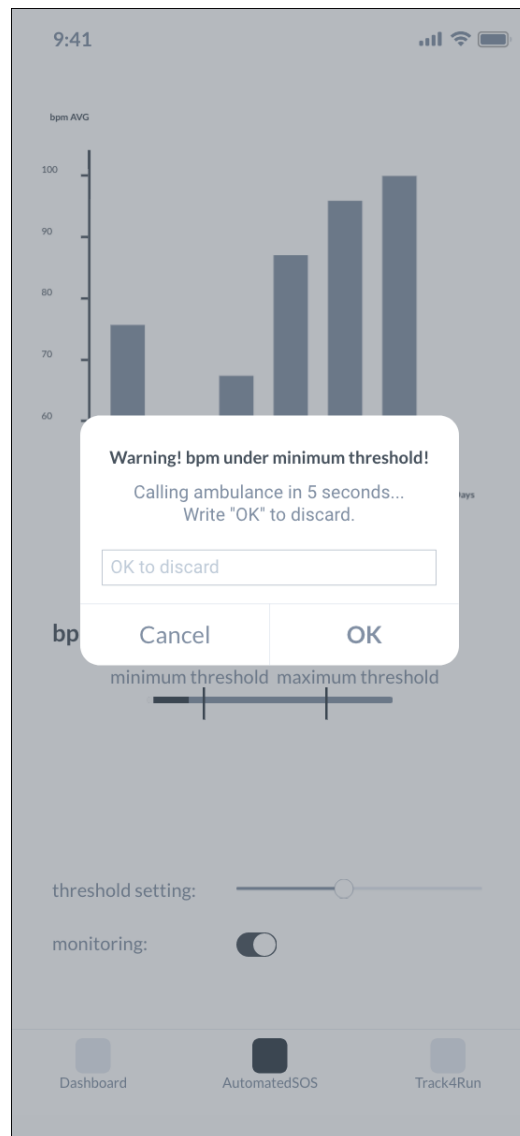


Figure 11: Emergency case

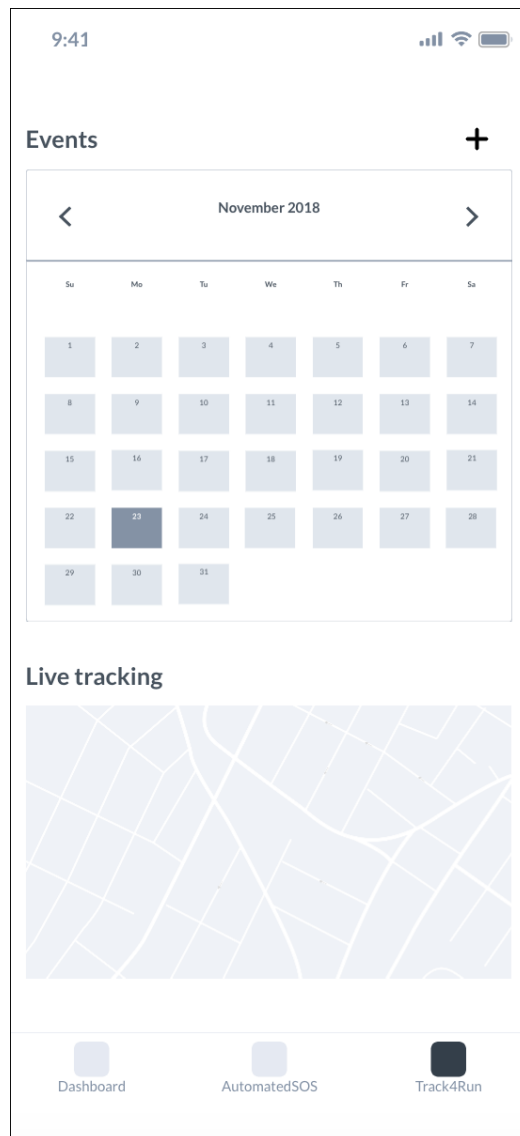


Figure 12: Track4Run dashboard


9:41

New run creation

Date selection

Sun 6 Sep	7	57
Mon 7 Sep	8	58
Tue 8 Sep	9	59
Wed 9 Sep	10	00
Thu 10 Sep	11	01
Fri 11 Sep	12	02
Sat 12 Sep	13	03

Path selection



Dashboard

AutomatedSOS

Track4Run

Figure 13: New run creation

### 3.1.2 Hardware Interfaces

TrackMe is a software application that does not export any hardware interface; instead, at the final user is required a smartphone and at the third-party is required a web browser.

### 3.1.3 Software Interfaces

- **Data4Help**

User's health data can be inserted in Data4Help section of the application manually or automatically synchronised with data acquired by individual's wearable device.

Third-parties can access to the different data through a web-based interface in which, in case the organisation is interested in the information of some specific individual, can select the single user by his/her fiscal code and send to him/her the request of data's visualisation; instead, in case of interest in accessing data of groups of individuals, companies can select the parameters of interest and send to TrackMe the request on hold for a positive response.

- **AutomatedSOS**

Similarly as Data4Help, AutomatedSOS automatically synchronises data acquired from user's wearable device and monitors them to guarantee the safety of elderly people.

In case of emergency, an ambulance will reach the location of the customer; moreover, AutomatedSOS allows to follow live-time the position of the coming ambulance.

- **Track4Run**

Since Track4Run aims to schedule and organise runs for the users of the service, it makes use of an external calendar service to allow individuals to create and en-roll races.

For those who do not participate to the runs, Track4Run features the possibility to see live-time the position of the runners.

### 3.1.4 Communication Interfaces

The system is going to use an API based communication system to serve both the mobile application and the third-parties.

Data will be exchanged in multi-platform data formats such as JSON or XML.

## 3.2 Functional Requirements

Here we map each goal to the corresponding machine requirement(s) and world domain assumption(s) that have to be made in order to reach it.

- G1 Users can be recognised by their credentials.
  - R1 Users can create an account with credentials.
  - R2 Credentials can be retrievable also if forgotten/lost.
- G2 Allow users to keep track of their health data.
  - R3 Users can log manually or automatically their data.
  - R6 The machine has to be able to read health and position data.
- G3 Allow users to have access to an overview of their data, including health parameters and performed activities.
  - R10 The machine has to be able to store users' data.
- G4 Allow users to manage their data access policy.
  - R4 Users have to be able to accept/deny access to single data access request.
  - R5 Users have to be able to see current data policies and to change them.
- G5 Allow users to monitor their performances during workouts.
  - D5 External services are reliable (map services).
  - D6 Users can be localised through GPS.
- G6 Each time vital signs go below a threshold value, first aid services have to be notified.
  - R6 The machine has to be able to read health and position data.
  - R7 The machine has to be able to recognise below threshold parameters.
  - D1 Data manually logged by users well describe reality.
  - D3 First aid services are ready to handle emergency notifications.
  - D7 Users are not affected with diseases that can influence heartbeat ratio.
- G7 Allow users to organise running events.
  - D4 Running paths proposed by users are well formed (e.g. legit by law).
- G8 Allow third parties to access data.
  - R8 The machine has to be able to communicate with third parties.
  - R9 The machine has to be able to recognise data fragmentation level.
  - R10 The machine has to be able to store users' data.

### 3.2.1 Definitions of use cases

Name	Sign up
Actor	User
Entry conditions	The application has been correctly installed on user's device
Events flow	1. Select the "Sign up" option 2. Fill in all the fields necessary for registering to the services 3. Tap on the sign up button to confirm the inputted data
Exit conditions	Data has been successfully saved and the customer can now use all the services offered by the application
Exceptions	1. User is already registered 2. User inputted email is already registered 3. User has chosen a username already taken 4. Not all fields were correctly filled 5. User has to recompile the sign up module correcting the invalid fields

Table 1: Sign up case

Name	Log in
Actor	User
Entry conditions	The installed application and the registration to the platform are mandatory to correctly log in
Events flow	1. Select the "Log in" option 2. Fill in the "Username" and "Password" fields 3. Tap on the log in button to confirm the credentials
Exit conditions	User has been correctly identified by the system and redirected to the main page of the application
Exceptions	1. User inputted an invalid username 2. User inputted an invalid password 3. User is not register to the platform 4. User has to correct the invalid fields of the log in page

Table 2: Log in use case

Name	Manual data acquisition
Actor	User
Entry conditions	The application has to be correctly installed and the user has to be logged in
Events flow	<ol style="list-style-type: none"> <li>1. Open the application and select the Data4Help section</li> <li>2. Tap the "Insert" button to manually insert data into the application</li> <li>3. Confirm the inserted data by tapping the "Confirm" button</li> <li>4. Wait until the data inserted are correctly processed by the server</li> </ol>
Exit conditions	Data are correctly updated and visible inside the application
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the insertion phase</li> <li>2. Inserted data are out-of-date</li> <li>3. User is invited to update the inserted information or to try again later</li> </ol>

Table 3: Manual data acquisition

Name	Automatic data acquisition
Actor	User
Entry conditions	The application has to be correctly installed and the user has to be logged in
Events flow	<ol style="list-style-type: none"> <li>1. Open the application and select the Data4Help section</li> <li>2. If not already updated, tap on the "synchronise" button</li> <li>3. Wait until the synchronisation between the smartphone and the wearable device is completed</li> <li>4. Wait until the data inserted are correctly processed by the server</li> </ol>
Exit conditions	Data are correctly updated and visible inside the application
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the insertion phase</li> <li>2. Synchronisation problem between the smartphone and the wearable device</li> <li>3. User is invited to check the connection between devices or to try again later</li> </ol>

Table 4: Automatic data acquisition



Name	Specific-individual data request
Actor	Third-party
Entry conditions	The third-party has to be logged in on the TrackMe's web service page
Events flow	<ol style="list-style-type: none"> <li>1. Select from the web page the interested individual by his/her fiscal code</li> <li>2. Send a request for data sharing to the individual</li> <li>3. Wait for the individual's response</li> </ol>
Exit conditions	Customer has accepted the third-party request and the latter is now in possession of the individual's information
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the request phase</li> <li>2. Negative response by the customer</li> <li>3. The third-party is invited to try with another individual or to try again later</li> </ol>

Table 5: Specific-individual data request

Name	Groups-of-individuals data request
Actor	Third-party
Entry conditions	The third-party has to be logged in on the TrackMe's web service page
Events flow	<ol style="list-style-type: none"> <li>1. Select from the web page the interested parameters for which the third-party wants to receive the data</li> <li>2. Wait for the data-anonymisation control by TrackMe for the requested parameters</li> </ol>
Exit conditions	TrackMe has ensured the anonymisation of the data and has accepted the third-party request; the latter is now in possession of the individual's information
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the request phase</li> <li>2. Negative response by TrackMe (data-anonymisation control failed)</li> <li>3. The third-party is invited to try with less restrictive parameters or to try again later</li> </ol>

Table 6: Groups-of-individuals data request

Name	Emergency situation
Actor	AutomatedSOS
Entry conditions	The application has to be correctly installed and the user has to be logged in
Events flow	<ol style="list-style-type: none"> <li>1. The service automatically notice that one or more vital parameters are under the safe threshold</li> <li>2. The service asks to the customer if there are any problems with a pop-up on-screen</li> <li>3. If the service doesn't receive a response within 5 seconds, an ambulance is alerted to reach the customer's location</li> </ol>
Exit conditions	The ambulance reach the customer's location to take of him/her
Exceptions	<ol style="list-style-type: none"> <li>1. Network error while contacting the ambulance service</li> <li>2. The system keeps trying to reach the ambulance service until it succeeds or the customer stops the process</li> </ol>

Table 7: Emergency situation

Name	Definition of a path
Actor	User
Entry conditions	The application has to be correctly installed and the user has to be logged in
Events flow	<ol style="list-style-type: none"> <li>1. Open the application and select the Track4Run section</li> <li>2. Tap on the "Create new run" button</li> <li>3. Select the starting and ending points on the map</li> <li>4. Select the main spots of the path in order to define a complete route on the map</li> <li>5. Wait until the data inserted are correctly processed by the server</li> </ol>
Exit conditions	The path has been correctly defined by the customer and the run can be shared with the other users
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the definition of the path</li> <li>2. The starting/ending point or the main spots are not selectable</li> <li>3. The path defined is not safe for the runners</li> <li>4. The user is invited to redefine the path or to try again later</li> </ol>

Table 8: Definition of a path

Name	Enrolment to a run
Actor	User
Entry conditions	The application has to be correctly installed and the user has to be logged in
Events flow	<ol style="list-style-type: none"> <li>1. Open the application and select the Track4Run section</li> <li>2. Tap on the "Available runs" button to show all the forthcoming runs</li> <li>3. Select a run from those available and confirm the enrolment</li> </ol>
Exit conditions	The user has correctly en-roll to the run
Exceptions	<ol style="list-style-type: none"> <li>1. Network error during the enrolment</li> <li>2. The run selected is no longer available/sold out</li> <li>3. The user is invited to choose another available run or to try again later</li> </ol>

Table 9: Enrolment to a run

### 3.2.2 Use case diagram

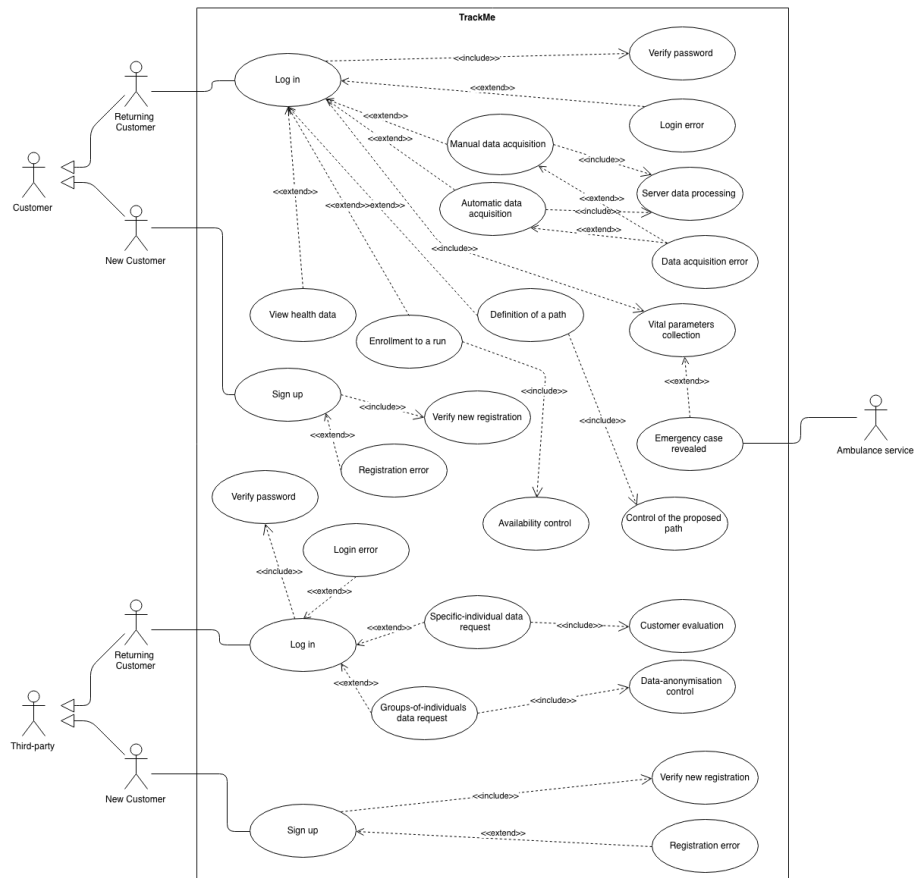


Figure 14: Use case diagram

### 3.3 Sequence diagrams

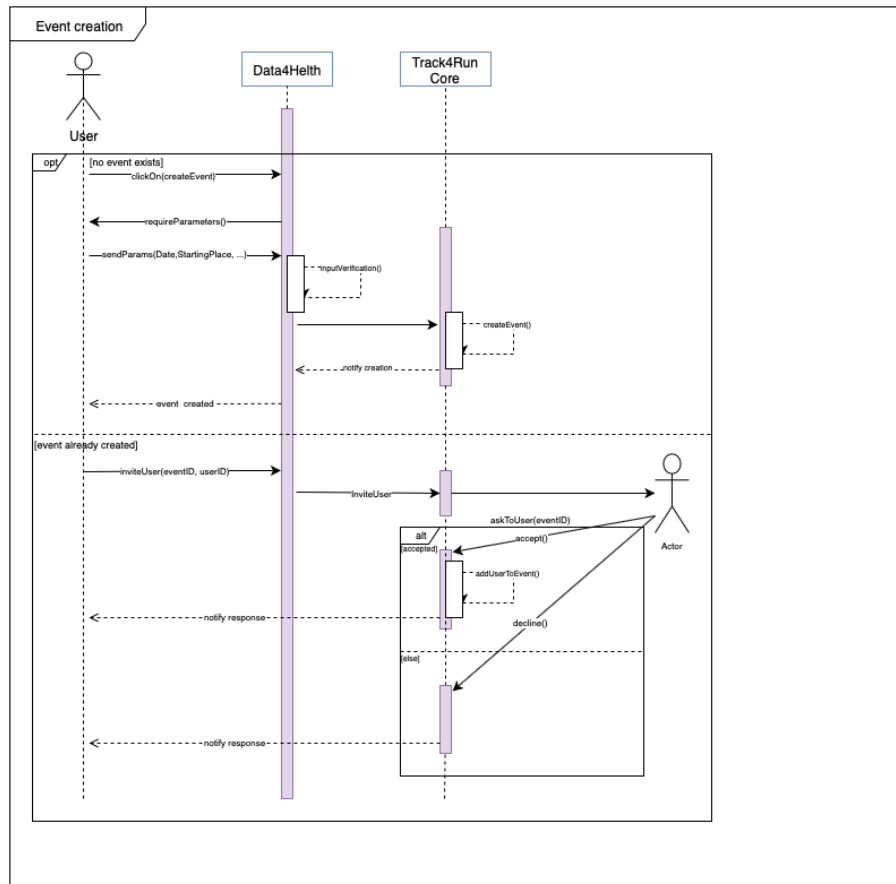


Figure 15: "Event creation and handling" sequence diagram

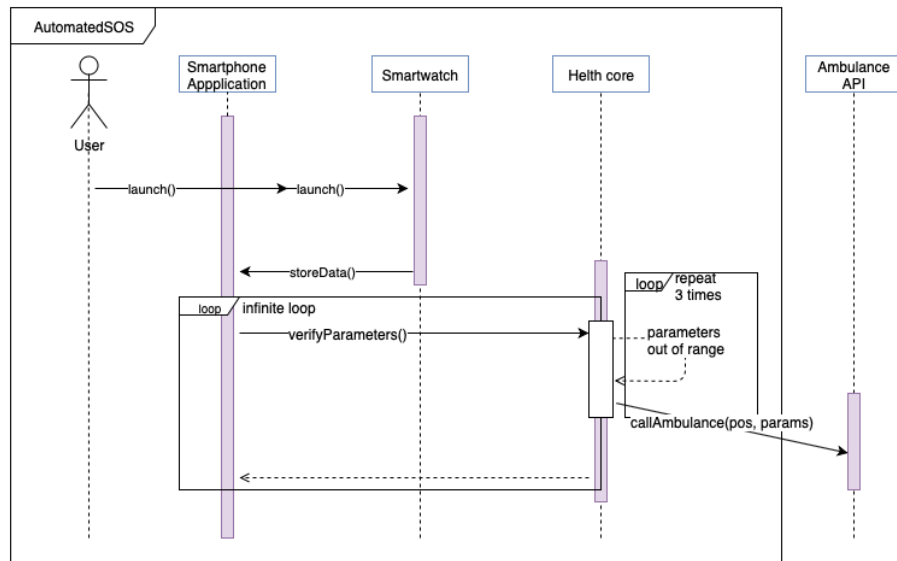


Figure 16: "AutomatedSOS" sequence diagram

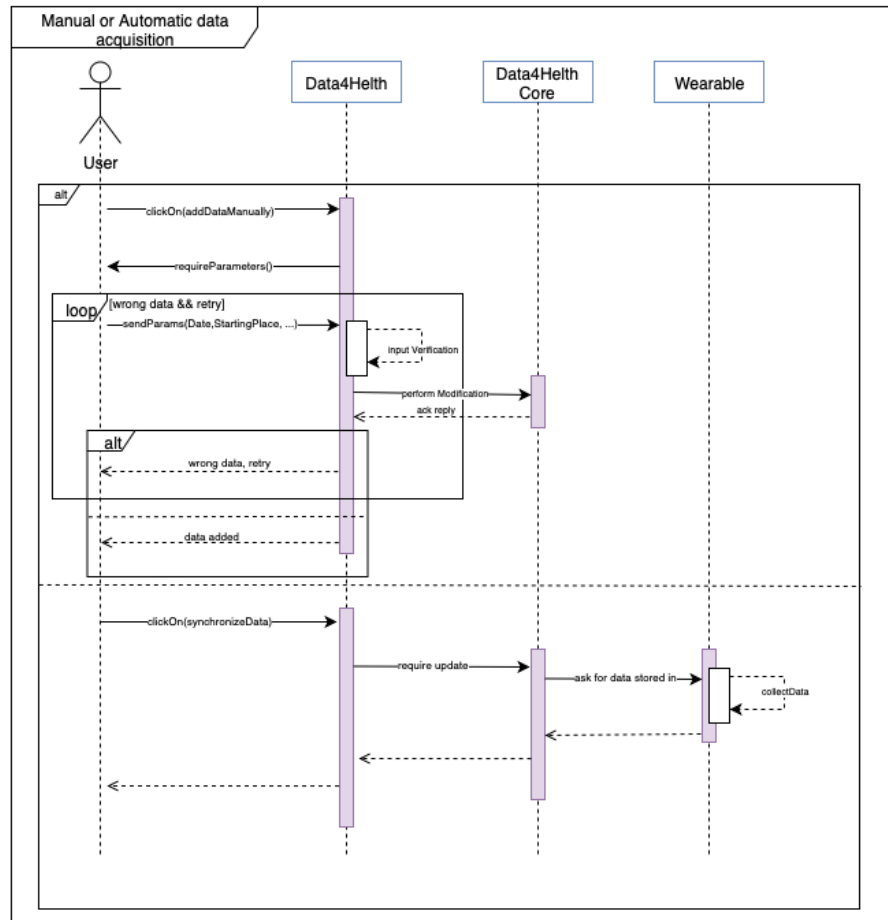


Figure 17: "Automatic/Manual data acquisition" sequence diagram

### **3.4 Performance Requirements**

The application is designed to be consumed by Italian citizens. We estimate around 30k users at app launch, with around 10% simultaneously active. We estimate that once the application will be brought over onto the Android platform, the total users count will increase by 25%, topping at about 38k users. The application will be able to adapt to the users active concurrently.

### **3.5 Design Constraints**

#### **3.5.1 Standards compliance**

The software application should handle a big amount of data, stored online on third-parties' servers. It also should manage a great number of simultaneous requests when, in example, third-parties ask to access data or during the period of enrolment to a run.

#### **3.5.2 Hardware limitations**

TrackMe is a software application available only on iOS devices. It requests an internet connection (WiFi, 3G, 4G/LTE) and GPS connectivity. Some functionalities are available only for the owners of wearable devices.

#### **3.5.3 Any other constraint**

The application needs to access to the current location of the user. It also may need to access individual's contact list to easily find possible user's friends.



## **3.6 Software System Attributes**

### **3.6.1 Reliability**

The application and the external services offered by the application must be available 24/7.

Servers' maintenance, communicated to the user in advance and only during determined hours of the day, is the only exceptions accepted.

### **3.6.2 Availability**

TrackMe services are available only in Italy for free.

### **3.6.3 Security**

The security and privacy of the final user's will be guarantee by TrackMe storing data and sensitive information only on encrypted servers.

For online transactions, secure internet protocols are going to be adopted.

### **3.6.4 Maintainability**

### **3.6.5 Portability**

Portability to the Android operating system will be take into consideration after a first initial phase of development dedicated to the iOS market.

The application will also be constantly updated to guarantee compatibility to an increasing number of wearable devices.

## **3.7 Scenarios**

### **3.7.1 Scenario 1**

Tim has to sustain an important medical exam aimed to discover if he suffers or not of tachycardia. He has available the latest model of smartwatch that supports the ECG monitor so his doctor suggests to use Data4Help to monitor for 24/48 hours his heart rate. Data4Help allows Tim to just use his smartwatch, instead of the classic Dynamic ECG machine, for registering his heart's data in the app so that his doctor, after committing a request to Tim through his fiscal code, can examine the results after few hours by the end of the test and give to Tim a response in a very short time.

### **3.7.2 Scenario 2**

Fitness & More is a brand new sport center situated near a very big residential area with many educational institutions from elementary to high schools. The centre offers swimming and tennis lessons with expert instructors and also a well supplied gym with personal coaches.

Fitness & More asks to TrackMe the access to weight and height data of kids between 6 and 19 year old who live in the above mentioned area to make an analysis of the presence of overweighted individuals and so sponsor its sport center.

### **3.7.3 Scenario 3**

The next PolimiRun will be held on the 11th of November in Lecco. Polisport, the sport organisation of the Politecnico di Milano, decided to arrange a few workouts for the runners who are attending the competition and so they suggest to use Track4Run to manage the path of the workouts around the city of Milan and also to let runners en-roll the training sessions.

## 4 Formal analysis using Alloy

In this section we give a description of the system through and Alloy model. The following relations are described:

- Data access request system is described. Third parties can access users' data if a previous request has been made and approved.
- SOS events are raised in case of low vital parameters.
- Users can join to running events.

The following assumptions are taken in order to make the model cleaner:

- All not necessary (to Alloy model) attributes of entities are omitted.
- All entities that have to be unique are identified by a single integer field: *ID*.
- Dates are represented through a single integer field: *day*.
- Threshold alert value for bpm is set to 4.

```

sig RunningEvent {
    runners: set User,
    path: one Path,
    date: Date
}

abstract sig RequestStatus{}
one sig APPROVED extends RequestStatus{}
one sig DECLINED extends RequestStatus{}
one sig PENDING extends RequestStatus{}

sig Request {
    subject: User,
    status: RequestStatus
}

sig Data {
    location: Location,
    bpm: Int
} { bpm > 0}

sig SOS {
    triggeredBy: Data,
    vital: Int
} { vital > 0}

-- All user's not fundamental fields are omitted.
-- Identification pass through a single integer field "Id"
sig User {
    id: Int,
    data: set Data
}{ id > 0 }

sig ThirdParty {
    id: Int,
    subscribedUsers: set User,
    requests: set Request
}{ id > 0}

-- In this Alloy model, date are simplified and represented with a single
  ↪ number.
sig Date {
    day: Int
}{ day > 0 }

sig Location {
    coordinateX: Int,
    coordinateY: Int
} { coordinateX > 0 and coordinateY > 0 }

sig Path {
    startingLocation: one Location,
    endingLocation: one Location
}

/** FAKTS */
fact uniqueEntities {
    no disjoint u1, u2 : User | u1.id = u2.id
    no disjoint t1,t2: ThirdParty | t1.id = t2.id
    no disjoint d1,d2 : Date | d1.day = d2.day
    no disjoint r1,r2 : Request | some t: ThirdParty |
        r1 in t.requests and r2 in t.requests and
        r1.subject = r2.subject
    no disjoint s1,s2 : SOS | s1.triggeredBy = s2.triggeredBy
}

```

```

-- Paths exist only if associated with one running event.
-- Date exist only if associated with one running event.
-- Since data are collected by the user, they cannot exist without him/her
-- Requests only exist if the associated Third Party exist
fact existence {
    all p : Path | one r: RunningEvent | r.path = p
    all d: Date | one r: RunningEvent | r.date = d
    all d: Data | one u: User | d in u.data
    all r: Request | one t: ThirdParty | r in t.requests
    all r: Request | one u: User | r.subject = u
}

fact subscriptions {
    -- if there is a request from a third party to a user that is
    --   ↳ APPROVED, then that third party is
    -- subscribed to that user
    all r: Request, t: ThirdParty |
        (r in t.requests and r.status = APPROVED)
        implies
        r.subject in t.subscribedUsers

    -- if there is a request from a third party to a user that is
    --   ↳ DECLINED or PENDING, that third party
    -- cannot be subscribed to that user
    all r: Request, t: ThirdParty |
        (r in t.requests and (r.status = DECLINED or r.status =
        --   ↳ PENDING))
        implies
        r.subject not in t.subscribedUsers

    -- if there is a third party which is subscribed to a user, then
    --   ↳ there must be a request approved
    -- for that third party and for that user
    all t: ThirdParty, u: User |
        u in t.subscribedUsers
        implies
        some r: Request | r.status = APPROVED and r.subject = u and r
        --   ↳ in t.requests
}

-- Threshold value for BPM is set to 4.
fact SOS {
    all s: SOS | s.vital < 4
    all s: SOS | one d: Data | s.vital = d.bpm and s.triggeredBy = d
}

/**** PREDICATES ****/
pred makeARequest [t, t': ThirdParty, u: User, r: Request]{
    r.subject = u
    r.status = PENDING
    t'.requests = t.requests + r
}

pred approveARequest [t, t':ThirdParty, u:User, r, r' :Request] {
    r'.subject = r.subject
    r'.status = APPROVED
    t'.requests = t.requests + r'
    t'.subscribedUsers = t.subscribedUsers + u
}

pred show {
    #User > 2
    #ThirdParty > 1
    #SOS > 2
    #Location > 1
    #RunningEvent >1

    all u: User | #u.data > 0
}

```

```
    all t: ThirdParty | #t.requests > 0
    some r: Request | r.status = APPROVED
    some r: Request | r.status = DECLINED
  }

run makeARequest for 4 but 7 Int
run approveARequest for 5 but 8 Int
run show
```

**Executing "Run makeARequest for 4 but 7 int"**

Solver=sat4j Bitwidth=7 MaxSeq=4 SkolemDepth=1 Symmetry=20  
84802 vars. 3824 primary vars. 287002 clauses. 744ms.

**Instance** found. Predicate is consistent. 283ms.

**Executing "Run approveARequest for 5 but 8 int"**

Solver=sat4j Bitwidth=8 MaxSeq=5 SkolemDepth=1 Symmetry=20  
230204 vars. 9320 primary vars. 795617 clauses. 1649ms.

**Instance** found. Predicate is consistent. 898ms.

**Executing "Run show"**

Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20  
7091 vars. 477 primary vars. 19337 clauses. 91ms.

**Instance** found. Predicate is consistent. 81ms.

Figure 18: Alloy analyser tool output.

## 4.1 World Generated

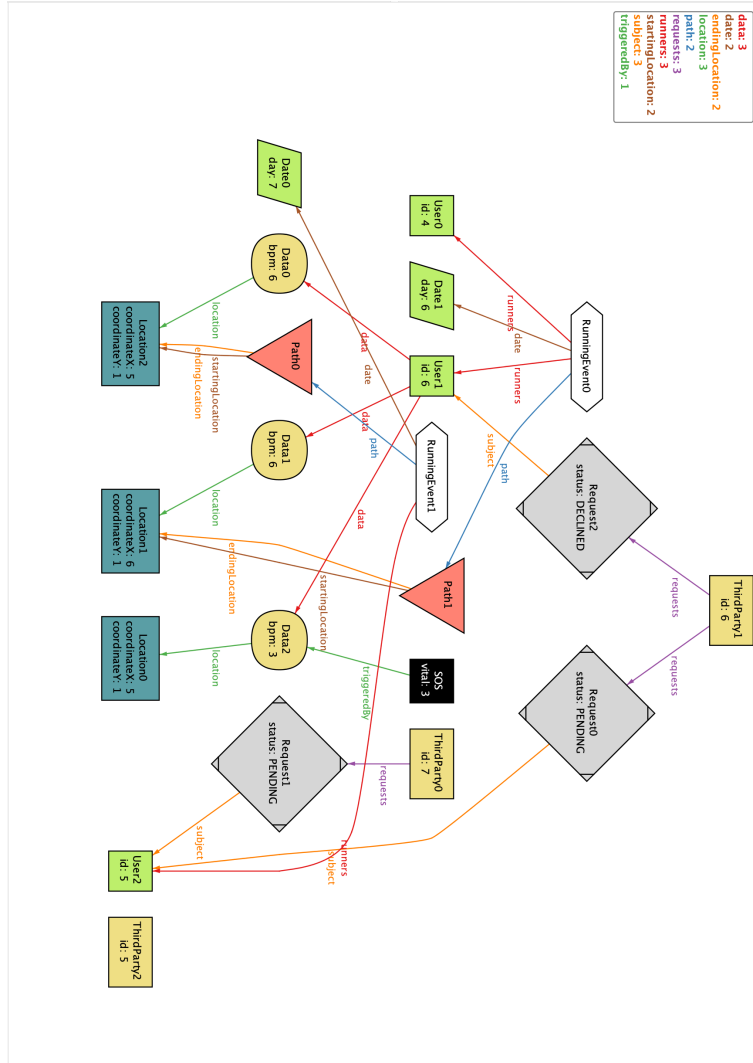


Figure 19: Generated Alloy World



**5    Effort spent**

**6    References**