



**POLITECNICO**  
**MILANO 1863**

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

---

# Requirement Analysis and Specification Document (RASD)

---

TRACKME

- v1.0 -

*Authors:*

**Avila, Diego**

**Schiatti, Laura**

**Virdi, Sukhpreet**

903988

904738

904204

November 11<sup>th</sup> , 2018

---

# Contents

---

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Context . . . . .	1
1.2	Purpose . . . . .	1
1.3	Scope . . . . .	2
1.3.1	Description of the given problem . . . . .	2
1.3.2	World and shared phenomena . . . . .	3
1.3.3	Goals . . . . .	4
1.4	Definitions, Acronyms, Abbreviations . . . . .	4
1.4.1	Definitions . . . . .	4
1.4.2	Acronyms . . . . .	5
1.4.3	Abbreviations . . . . .	5
1.5	Revision history . . . . .	6
1.6	Document structure . . . . .	6
<b>2</b>	<b>Overall description</b>	<b>7</b>
2.1	Product perspective . . . . .	7
2.2	Product functions . . . . .	10
2.3	User characteristics . . . . .	10
2.4	Assumptions, dependencies and constraints . . . . .	11
2.4.1	Domain assumptions . . . . .	11
<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 . . . . .	18
3.1.3	Software interfaces . . . . .	19
3.1.4	Communication interfaces . . . . .	19
3.2	Functional requirements . . . . .	19
3.2.1	Use case diagrams . . . . .	23
3.2.2	Use cases description . . . . .	25
3.2.3	Activity diagrams . . . . .	41

3.2.4	Sequence diagrams . . . . .	44
3.2.5	Requirements traceability matrix . . . . .	46
3.3	Performance requirements . . . . .	48
3.4	Design constraints . . . . .	48
3.4.1	Standards compliance . . . . .	48
3.4.2	Hardware limitations . . . . .	49
3.4.3	Any other constraint . . . . .	49
3.5	Software system attributes . . . . .	49
3.5.1	Reliability . . . . .	49
3.5.2	Availability . . . . .	49
3.5.3	Security . . . . .	49
3.5.4	Maintainability . . . . .	50
3.5.5	Portability . . . . .	50
3.5.6	Usability . . . . .	50
<b>4</b>	<b>Formal analysis using Alloy</b>	<b>51</b>
4.1	Data4Help . . . . .	51
4.1.1	Purpose of the model . . . . .	51
4.1.2	Alloy model . . . . .	52
4.1.3	Generated worlds . . . . .	54
4.2	AutomatedSOS . . . . .	56
4.2.1	Purpose of the model . . . . .	56
4.2.2	Alloy model . . . . .	57
4.2.3	Generated worlds . . . . .	59
<b>5</b>	<b>Effort spent</b>	<b>61</b>
<b>6</b>	<b>References</b>	<b>62</b>

---

# List of Figures

---

1.1	High-level description of the problem . . . . .	3
2.1	Domain Model . . . . .	7
2.2	Data4Help Statechart Diagram . . . . .	8
2.3	AutomatedSOS Statechart Diagram . . . . .	9
2.4	Track4Run Statechart Diagram . . . . .	9
3.1	UI: TrackMe's Home Page . . . . .	13
3.2	UI: Login Page . . . . .	14
3.3	UI: Individual User's Registration page . . . . .	14
3.4	UI: Third Party's Registration page . . . . .	15
3.5	UI: Registered individual's Dashboard . . . . .	15
3.6	UI: Registered user's Dashboard who can choose to Accept or Reject the requests . . . . .	16
3.7	UI: Third party user's Dashboard to make requests . . . . .	16
3.8	UI: Organiser's Dashboard - create new run . . . . .	16
3.9	UI: Organizer's Dashboard - Define running circuit . . . . .	17
3.10	UI: Spectators' view List of available run . . . . .	17
3.11	UI: Spectators' view during the race . . . . .	18
3.12	UI: Spectators' view when race has ended . . . . .	18
3.13	Data4Help use case diagram . . . . .	23
3.14	AutomatedSOS Use Case Diagram . . . . .	24
3.15	Track4Run Use Case Diagram . . . . .	25
3.16	Data4Help - Provide Data To Third Parties Activity Diagram . . . . .	42
3.17	AutomatedSOS - Monitoring Individuals Activity Diagram . . . . .	43
3.18	Track4Run - Create a Run and Invite Individuals Activity Diagram . . . . .	43
3.19	Track4Run - Enroll into a Run Activity Diagram . . . . .	44
3.20	Track4Run - Track Runners Activity Diagram . . . . .	44
3.21	Data4Help - Accept Request Sequence Diagram . . . . .	45
3.22	Data4Help - Access to bulk data and subscribe Sequence Diagram . . . . .	45
3.23	Track4Run - Accept Invitation Sequence Diagram . . . . .	46

4.1	D4H - First generated world . . . . .	55
4.2	D4H - Second generated world . . . . .	56
4.3	ASOS - First generated world . . . . .	59
4.4	ASOS - Second generated world . . . . .	60

---

---

# List of Tables

---

1.1	Revision history timeline . . . . .	6
3.1	Traceability matrix . . . . .	48
5.1	Time spent by all team members . . . . .	61
5.2	Time spent by each team member . . . . .	61

---

# Introduction

---

## 1.1 Context

Nowadays, due to the availability of a huge variety of smart electronic devices, more and more applications are developed to help people in their day-to-day activities. In the health-care field, wearable devices such as smartwatches are highly useful since they can be used to collect information about general well-being of users by means of mobile sensor technologies. As expected, measured data has several possible applications including, patient diagnostics and treatment or research motivations.

**TrackMe** is a company that develops health-monitoring devices devoted to measure and record different parameters related to the health status of a person (i.e. body temperature, blood pressure, heart pulse rate and percentage of O<sub>2</sub> in the blood) and also their location. TrackMe health smartwatch is synchronized with an app that gives users access to their data and stats.

## 1.2 Purpose

Taking into account the long list of currently available wearable devices, **TrackMe** is aware that the market is plenty of health-care wearable devices, therefore, to stand out against its competitors, the key ingredient is improving customers experience, and doing so requires understanding their needs and how they interact with the system, in order to provide personalized recommendations and correctly oriented new services.

After analyzing users behaviour, TrackMe decided to focus on third-party companies and profit from users data in a direct way (i.e. extend its business model by implementing **data trading**). This is, provide the collected data (i.e. location and health status) to third parties by means of a new software-based service called **Data4Help**.

Moreover, after some time, TrackMe realizes that a good part of its third-party customers want to use the data acquired through Data4Help to offer a personalized SOS service to elderly people, and decides to build a new service called **AutomatedSOS** to provide a

personal alarm service to subscribed customers by monitoring their health status.

Finally, TrackMe realizes that another great source of revenues could be the development of a service to track athletes participating in a run. In this case, the service, called **Track4Run**, will allow run organizers to define the path, TrackMe wearable-devices users to enrol, and spectators to see on the map the position of all runners during the run.

## 1.3 Scope

### 1.3.1 Description of the given problem

The TrackMe environment will be composed by three systems, whose scope are defined in this section.

**Data4Help** is a system capable to store the physical health data of any individual using any TrackMe wearable device. All the stored data will be available to any third party company, after request and approval of its use is sent to the TrackMe wearable device user, who will be able to accept or reject the request, under no condition. Also, third party companies will be able to request data of a group of users, and Data4Help will provide that information under the solely condition of be able to anonymize it.

In addition, **AutomatedSOS** is a system that will be built on top of Data4Help, and will send requests to every elderly individual in order to access its physical health parameters. It will monitor the physical health status of all of its users in order to establish their health condition, based on previously defined thresholds. If any of the user's health parameters is under or above its threshold, the system will contact the associated health-care service for that user. AutomatedSOS will do its best effort to contact the health-care service, but under no circumstances will follow-up the status of the health-care response, meaning that it will not know if the health-care service answered the notification or not.

Finally, **Track4Run**, also a system built on top of Data4Help, will send requests to every user in order to access their location and let them participate in a defined running circuit. Track4Run organizer users, will be able to setup a race, define its running circuit and send invitations to all the TrackMe wearable device users. Also, the race spectators will be able to follow the participants' location using the Track4Run web site. The location of every participant will only be available during the race.



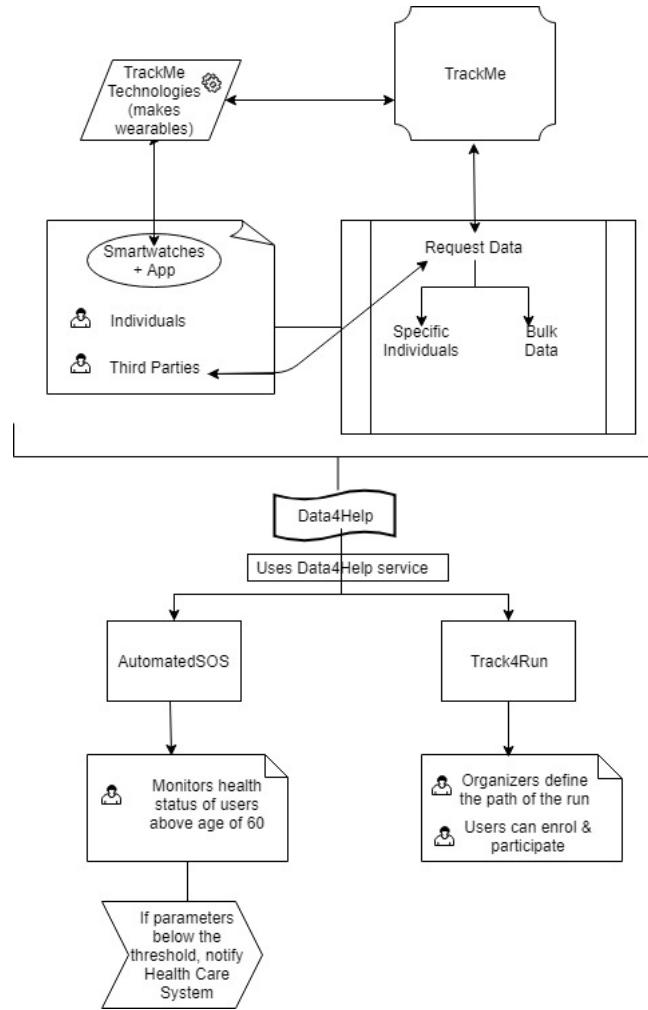


Figure 1.1: High-level description of the problem. It is possible to observe the relationship between the current TrackMe system and Data4Help, and this system with the other two to be built

### 1.3.2 World and shared phenomena

- **World Phenomena**

In order to better understand which entities are relevant for the system and how they interact, it is essential to describe the real world events that are involved, they are

- TrackMe wearable devices.
- Individuals sharing their personal data.
- Third-party customers willing to use the data acquired through the devices.
- Health-care system .

- **Shared Phenomena**

The shared phenomena is composed by all the relevant interactions between the

world and the system. Every interaction is part of a relationship between entities in the real world and Trackme environment. The main ones are listed below.

- The physical health parameters collected by the Trackme devices (i.e. blood pressure, body temperature, etc.), and stored by Data4Help
- Individuals location collected by the Trackme device, and stored by Data4Help
- Health-care system, that let AutomatedSOS send the alarms
- The running circuit defined in Track4Run
- The current location of the athletes participating in a run.

### 1.3.3 Goals

The goals are divided according to each service TrackMe wants to offer to its customers:

- **Data4Help**

- [G1] The individual could allow (or refuse) Data4Help to use their data.
- [G2] The third-party company should be able to access data of a specific individual
- [G3] The third-party company should be able to access anonymized data of groups of individuals under certain constraints.
- [G4] The third-party company could subscribe to get new data related to specific individuals or previously saved search.

- **AutomatedSOS**

- [G5] Provide a service capable to send a notification to the health-care service when any individual's parameter is out of range.

- **Track4Run**

- [G6] Run organizers could define the path for a given run and TrackMe users can enrol to it.
- [G7] Run spectators could track the position of all the runners during a race.

## 1.4 Definitions, Acronyms, Abbreviations

### 1.4.1 Definitions

- **Data trading:** Generate revenue from user data in a much more direct way, by selling user data to a third party.

- **Health status:** Collection of the last measured overall physical health parameters of a user or a group of users.
- **Remote monitoring:** Remote Monitoring (RMON) is a standard specification that facilitates the monitoring of network operational activities through the use of remote devices known as monitors or probes(here, we are using smartwatches).
- **Wearable device:** Devices that can be used to collect data and monitor users' overall physical health, such as body temperature, blood pressure, heart pulse rate, etc.
- **Third party company:** Customer who needs to know the health status of the population for different purposes (e.g. health insurance companies)
- **Normal range:** When the parameters are under the defined threshold values (here, in context: health parameters).
- **Running circuit:** Path defined by the organizer for the run, using the set of nodes.
- **Anonymize:** The action of anonymize means that an individual's identity cannot be inferred using the available data.
- **Parameter out of its normal range:** Meaning that the parameter is under or above a defined threshold.

### 1.4.2 Acronyms

- RASD: Requirement Analysis and Specification Document
- D4H: Data4Help
- ASOS: AutomatedSOS
- T4R: Track4Run
- SSN: Social Security Number

### 1.4.3 Abbreviations

- $[Gn]$ : n-goal.
- $[Dn]$ : n-domain assumption.
- $[Rn]$ : n-functional requirement.
- $[UCn]$ : n-use case.

## 1.5 Revision history

It is important to keep track of the revisions made to this document:

Version	Last modified date
1.0	11 <sup>th</sup> November, 2018

Table 1.1: Revision history timeline

## 1.6 Document structure

This document is divided in six parts, each one devoted to approach each one of the steps required to apply requirements engineering techniques.

- Chapter 1 gives an introduction to the problem and describes the purpose of the application TrackMe. The scope of the application is defined by stating the goals and description of the problem.
- Chapter 2 presents the overall description of the project. The product perspective includes details on the shared phenomena and the domain models.
- Chapter 3 contains the external interface requirements, including: user interfaces, hardware interfaces, software interfaces and communication interfaces. Furthermore, the functional requirements are defined by using use case and sequence diagram. The non-functional requirements are defined through performance requirements, design constraints and software system attributes. All this aspects will be useful for the development team.
- Chapter 4 includes the alloy model and the discussion of its purpose. Also, a world generated by it is shown.
- Chapter 5 shows the effort spent by each group member while working on this project.
- Chapter 6 includes the reference documents.

# Overall description

## 2.1 Product perspective

In the previous section, the scope of the application was delimited and explained in a shallow way, but at this point it is useful to include further details on the shared phenomena and a domain model as a visual representation of the system (Figure 2.1).

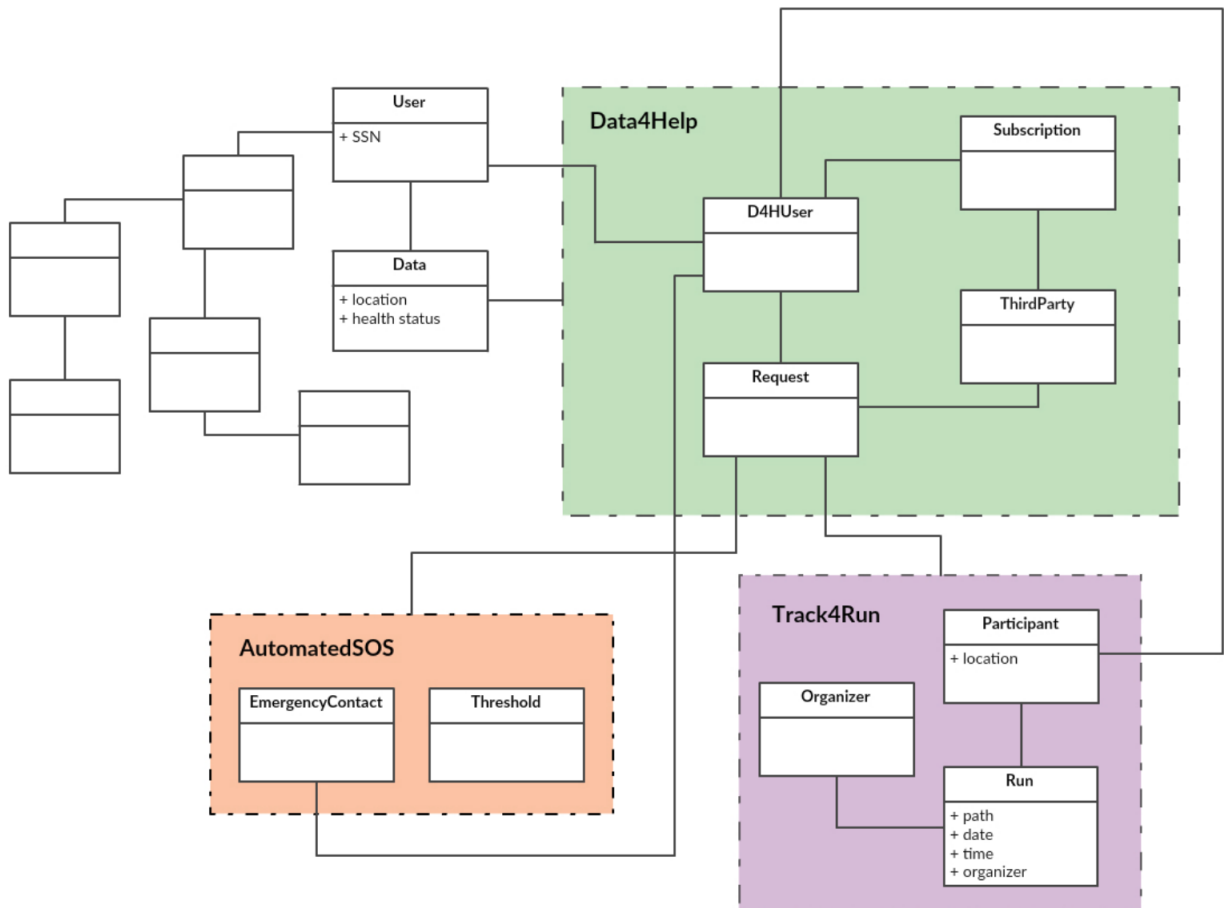


Figure 2.1: Domain Model

Augmenting the services offered by TrackMe requires to enlarge the existing model in such a way that it can include the abstraction of those features. From this perspective, to explain in detail how the data will be organized in the upcoming system, the structure of TrackMe up to now will be treated as a "black box". This means, only those parts of

the whole data model that will allow us to obtain users' basic information and collected data (required by **Data4Help**) will be considered.

On the other hand, **AutomatedSOS** and **Track4Run** are treated as third-parties that make requests for the data that Data4Help offers. Every time a user agrees to activate any of those services, a new request is sent to Data4Help to obtain his data.

To understand the main events happening in the system, it is useful to use statechart diagrams (described in the figure below). In **D4H** (Figure 2.2) the main task is handling requests made by third parties registered in the system. At the beginning each query (i.e. what the third party is asking for) is considered as *unprocessed*. While the constraints are checked, the query is on a *pending* state and can become either *rejected* or *approved*. In the end, they are considered *completed*.

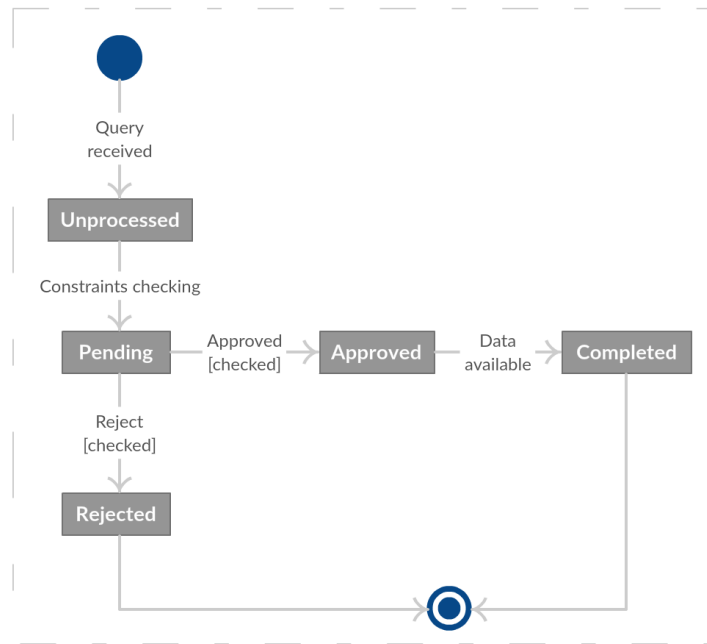


Figure 2.2: Data4Help Statechart Diagram

On the other hand, **ASOS** (Figure 2.3) is an additional service offered to individuals older than 60 years. It is a background process in charge of monitoring the health status of each individual until an anomaly happens (i.e. any parameter out of its normal range), then a notification is sent to his corresponding health-care service (i.e. send the location) and after, the system loops back to be ready to process new upcoming data.

Another fact to highlight is ASOS associates a status to each individual, *stable* or

*critical*, needed to decide if triggering the notification or not (to avoid sending the same notification for a user).

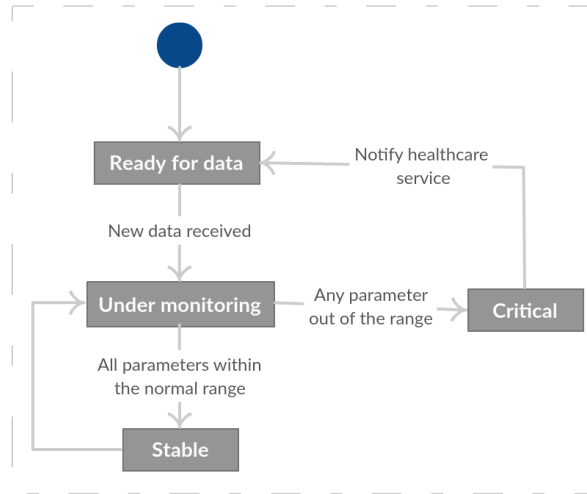


Figure 2.3: AutomatedSOS Statechart Diagram

Finally, **T4R** (Figure 2.4), needs to tackle the "how to setup a run and make it available for the spectators" task. To do so, after an organizer creates a run, it can be in different states depending on whether it was completely configured (i.e. define a running circuit) or not. Also it will be active or unactive according to the date and time when the run will take place.

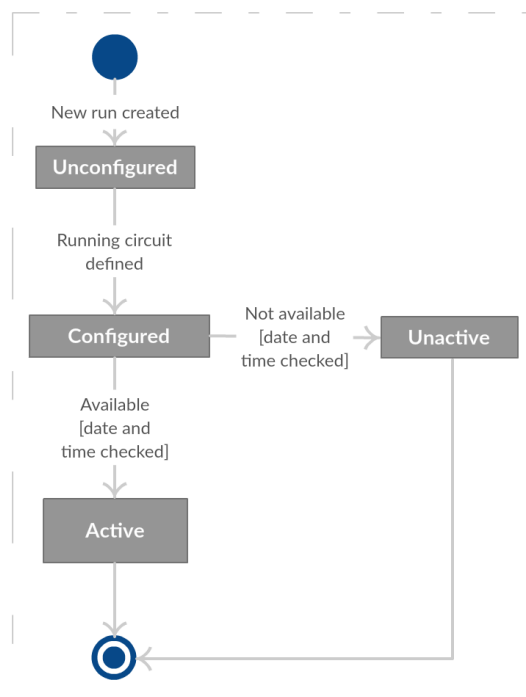


Figure 2.4: Track4Run Statechart Diagram

## 2.2 Product functions

The TrackMe environment is composed, as said before, by a set of three services, with D4H as the leading service. ASOS and T4R are going to be build on top of D4H, and will make use of all of its functionalities. Below, the main features of each service are listed, and a description is offered.

- **Data4Help**

D4H will be the leading service, and the features it will provide are mostly related to registered third party companies. Companies will be able to access different types of data from the individuals wearing the TrackMe devices. They will be able to subscribe to a specific individual data, or to a group of anonymized individuals' data, as long as certain restrictions are fulfilled. The individuals, on the other hand, will be able to accept or reject the request of accessing his/her data, and the third party companies will be notified of the individuals' decision.

- **AutomatedSOS**

ASOS is a complementary service offered to the senior range of users, and it will be built on top of the D4H service. All the elderly individuals of D4H will receive a request to subscribe to this service, whose main feature is to contact the individual's health-care service every time any health parameter collected using TrackMe devices becomes critical (out of normal range).

- **Track4Run**

T4R is the last service offered by TrackMe, and it will, also, be build on top of D4H. Designed as a service for run organizers and runners, who operate the TrackMe devices. The run organizers will be able to define a running circuit, and send invitations to the TrackMe device users; The individuals will be able to register to any particular competition they prefer, only if they have not registered on another competition at the same date and time. Furthermore, while the race is taking place, all spectators will be able to spot, through the T4R site, the location of each registered individual in the circuit.

## 2.3 User characteristics

The target users of the TrackMe system are:

- **Individuals**

- Can register and allows TrackMe to store, analyze and process their data.



- Can manage the requests if some third party company wants access to their data individually.
- If they are above 60 years, they can avail themselves of ASOS service by using the subscribe option.
- Can enroll into the upcoming runs.
- **Organizers**
  - Are users that coordinate runs by defining the path, such that, other individuals are able to enroll.
- **Spectators**
  - Are the audience of the ongoing runs.
  - Are anyone who uses the T4R website.
- **Third party companies**
  - Can register and query for the data.
  - Can subscribe to the acquired data so as to get new data when it is available.
  - ASOS and T4R are considered third party, with the particularity of being managed directly by TrackMe.

Therefore, all the constraints derived from these characteristics must be satisfied from the TrackMe system, as much as possible.

## 2.4 Assumptions, dependencies and constraints

### 2.4.1 Domain assumptions

Domain assumptions help to make clear what it is expected from the external environment and allows constraining the software-driven machine so as to stay in the domain of interest.

- [D1] TrackMe guarantees that the wearables can provide sufficient accuracy and sensitivity when monitoring individuals.
- [D2] TrackMe addresses data protection and integrity against possible attacks.
- [D3] TrackMe devices are up and running during monitoring.

- [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
- [D5] The provided SSN by the individual is valid and trustable.
- [D6] Out of coverage scenarios cannot be handled by ASOS so as to guarantee a 5 seconds reaction.
- [D7] The organizers hold all the legal requirements and permissions necessary to set up a run.

# Specific requirements

This section contains all of the functional and quality requirements of the system. It gives a detailed description of the system and all its features.

## 3.1 External interface requirements

This section provides a detailed description of all inputs and outputs from the system. It also gives a description of the hardware, software and communication interfaces and provides basic prototypes of the user interface.

### 3.1.1 User interfaces

The following mock-ups represent a basic idea of what the web application will look like after the first release:

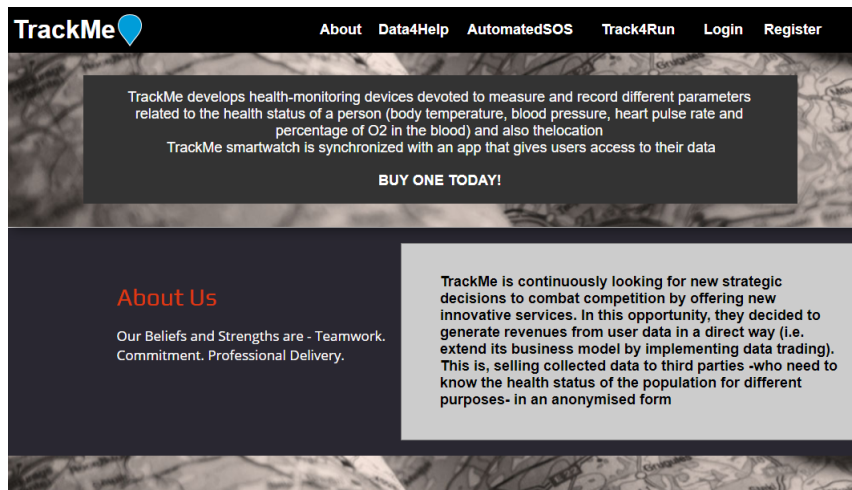


Figure 3.1: TrackMe's Home Page

As when new customers visit the home page of TrackMe, they can read about the work that our company does, what services we offer, what benefits users can achieve by joining the community. In addition, they can buy the wearables and get more information, how

to use them.

Next, comes the web page through which users can Login into the system (if already registered). And if not, they can register themselves through the Register web-page. There are separate register forms for the Individual users and for third party users, which can be seen in the Figure 3.2.

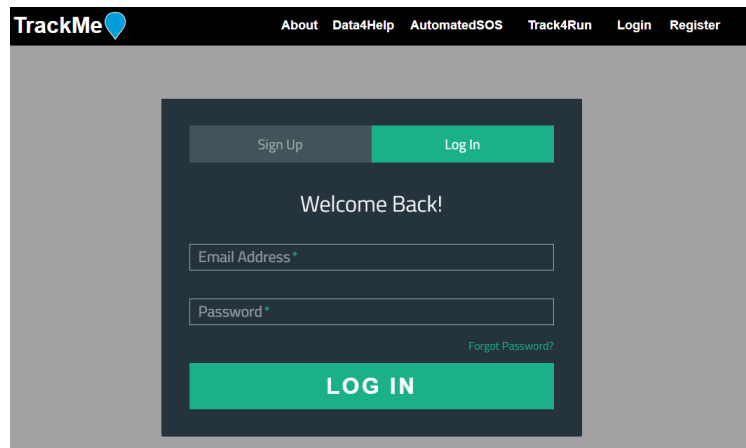
The screenshot shows the TrackMe website's login interface. At the top, a black navigation bar contains the TrackMe logo and links for About, Data4Help, AutomatedSOS, Track4Run, Login, and Register. The main content area has a dark gray background. A central white box contains a 'Sign Up' button (disabled) and a 'Log In' button (active). Below these is the text 'Welcome Back!'. There are input fields for 'Email Address\*' and 'Password\*', with a 'Forgot Password?' link next to the password field. A large green 'LOG IN' button is at the bottom of the white box.

Figure 3.2: Login Page

Registration is free for all the users and it doesn't take much time to complete the forms, as can be seen in the Figure 3.3 and Figure 3.4.

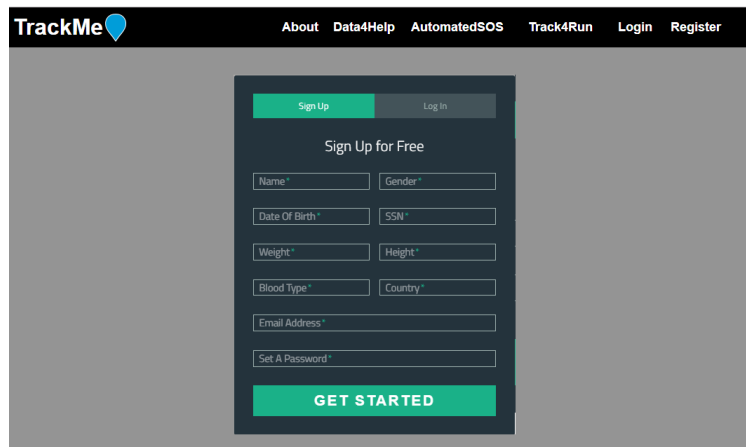
The screenshot shows the TrackMe website's individual user registration form. The navigation bar is the same as in Figure 3.2. The main content area has a dark gray background. A central white box contains a 'Sign Up' button (active) and a 'Log In' button (disabled). Below these is the text 'Sign Up for Free'. There are several input fields: 'Name\*', 'Gender\*', 'Date Of Birth\*', 'SSN\*', 'Weight\*', 'Height\*', 'Blood Type\*', 'Country\*', 'Email Address\*', and 'Set A Password\*'. A large green 'GET STARTED' button is at the bottom of the white box.

Figure 3.3: Individual User's Registration page

Figure 3.4: Third Party's Registration page

In the Figure 3.5, its personal dashboard for the individual users, who can manage and view the requests and are able to view who all third parties have subscribed to their data.

#	Date	Time	Business Name
3326	10/21/2013	3:29 PM	AutomatedSOS
3325	10/21/2013	3:20 PM	Track4Run
3324	10/21/2013	3:03 PM	ABC

Figure 3.5: Registered user's Dashboard

In the Figure 3.6, can be seen that the customer has two options to choose: accept or reject the request for data acquisition. As soon as an action is taken upon the request, it gets deleted from the page.

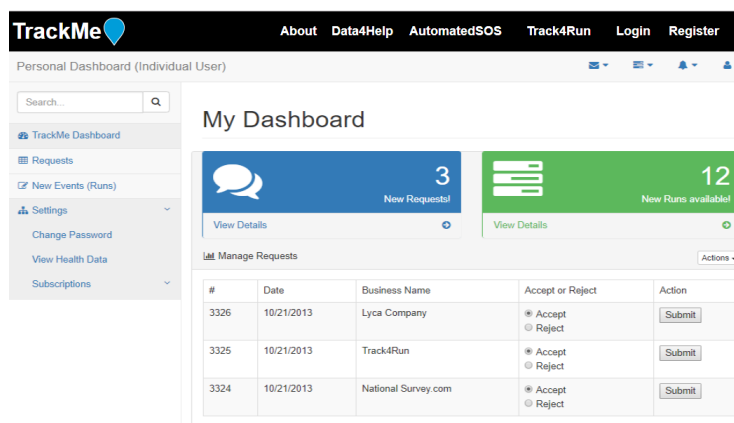


Figure 3.6: Registered user's Dashboard who can choose to Accept or Reject the requests

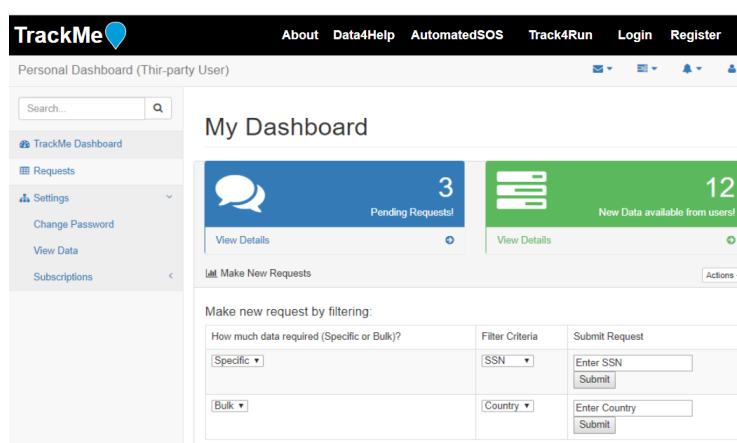


Figure 3.7: Third party user's Dashboard to make requests

In the Figure 3.7, Third party users can make request for the data using the filtering criteria provided by the system.

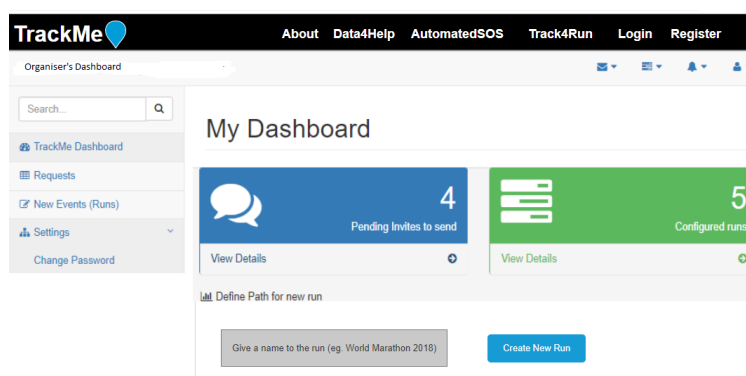


Figure 3.8: Organizer's Dashboard- create new run

In the Figure 3.8, Organizer's are able to view the pending invites to send to targeted participants and here, they can start configuring a new run by clicking on 'Create new run' button.

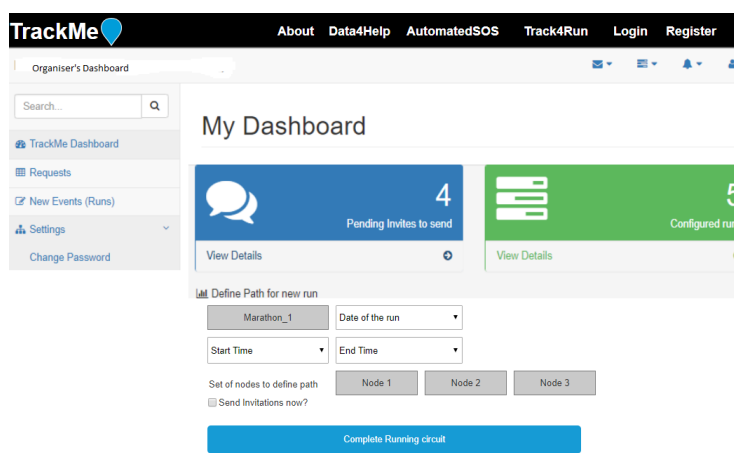


Figure 3.9: Organizer's Dashboard- Define running circuit

In the Figure 3.9, Organizer's are able to define the parameters to set up a complete running circuit and here, they can select whether to send invitations now or not by clicking the checkbox.

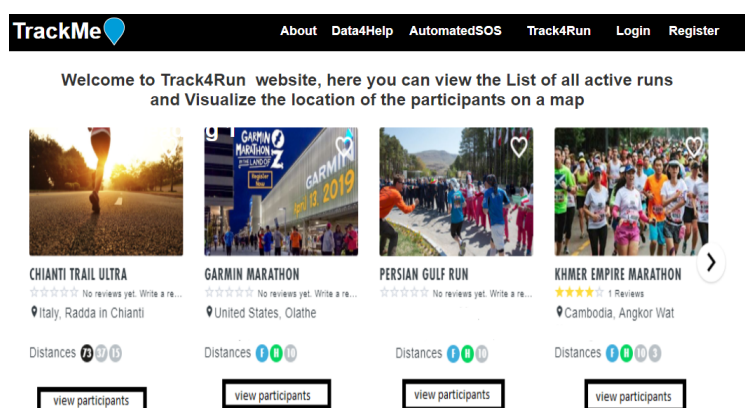


Figure 3.10: Spectators' view List of available run

In the Figure 3.10, Spectators are able to view all the available runs, and by clicking on view participants, they are redirected to the map-view of the selected run.

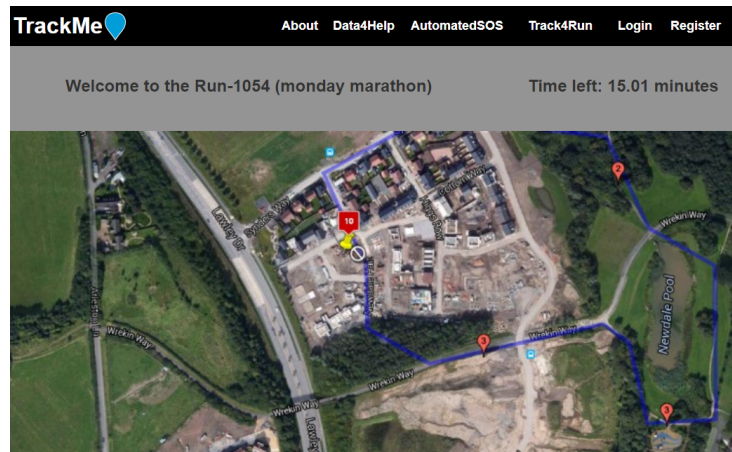


Figure 3.11: Spectators' view during the race

In the Figure 3.11, Spectators are able to view the participants' position during the run and time left for the race to end.

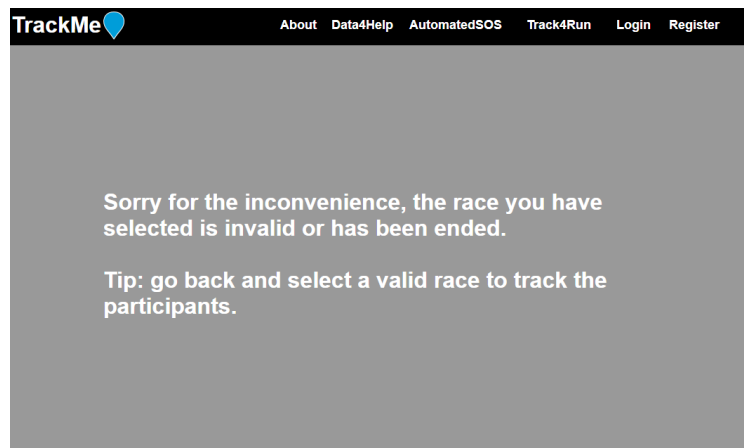


Figure 3.12: Spectators' view when race has ended

In the Figure 3.12, Spectators are able to view an error message when the race has been ended.

### 3.1.2 Hardware interfaces

No such direct hardware interfaces are required by our system, as this is a web portal application. The physical GPS is managed by the TrackMe devices. And the hardware connection to the database server is managed by the underlying operating system on the web server. Thus, it can be easily accessible from any device or any location if **Internet service** is available.



### 3.1.3 Software interfaces

Web applications are by nature distributed applications. Specifically, web applications are accessed with a web browser and are popular because of the ease of using the browser as a user client. The software Interfaces provided by the system are as follows:

- **Data4Help:** A service developed by TrackMe which collects all the data of the individuals and provides the data if necessary; It provides an interface to the third party users to request data.
- **AutomatedSOS:** A service developed on the top of D4H which provides support (the interface that communicates with the health-care service) in case of emergency to the subscribed users.
- **Track4Run:** A service developed on the top of D4H which provides *different interfaces* to organizers (to build the path of a run), individuals (to enroll in available run only under certain constraints) and spectators (to view the position of the runners during a run).

### 3.1.4 Communication interfaces

The communication between different parts of the system is important since they depend on each other. However, the way communication is achieved is not important for the system and is therefore handled by the underlying operating system for web portal.

## 3.2 Functional requirements

The functional requirements are those which are the fundamental actions of the system. As before, every requirement is divided by subsystem (D4H, ASOS and T4R), and the relation with each goal is shown.

[G1] The individual could allow (or refuse) Data4Help to use their data.

[D2] TrackMe addresses data protection and integrity against possible attacks.

[D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.

[D5] The provided SSN by the individual is valid and trustable.

[R1] The system must allow an individual to register a new account.

- [R2] The system must allow an individual to access to their account.
- [R3] The system must allow an individual to accept or reject their requests of accessing personal data.
- [R4] The system must be able to communicate with TrackMe database in order to obtain the health status and location of an individual.
- [G2] The third-party company should be able to access data of a specific individual.
  - [D2] TrackMe addresses data protection and integrity against possible attacks.
  - [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
  - [D5] The provided SSN by the individual is valid and trustable.
  - [R5] The system must allow a third party company to register a new account.
  - [R6] The system must allow a third party company to access to its account.
  - [R7] The system must be able to notify the individual that a third party company wants to access its data.
  - [R7.1] The system must be able to notify the the third party company about the individual's decision.
  - [R8] The system must allow a third party company to search for an individual health status and location using his/her SSN.
- [G3] The third-party company should be able to access anonymized data of groups of individuals under certain constraints.
  - [D2] TrackMe addresses data protection and integrity against possible attacks.
  - [D3] TrackMe devices are up and running during monitoring.
  - [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
  - [R5] The system must allow a third party company to register a new account.
  - [R6] The system must allow a third party company to access to its account.
  - [R9] The system must allow a third party company to filter data of an anonymized group of individuals by country, age, gender and blood type parameters.
  - [R10] The system must be able to anonymize the data of a group of individuals.
- [G4] The third-party company could subscribe to get new data related to specific individuals or previously saved query.

- [D2] TrackMe addresses data protection and integrity against possible attacks.
- [D3] TrackMe devices are up and running during monitoring.
- [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
- [D5] The provided SSN by the individual is valid and trustable.
- [R5] The system must allow a third party company to register a new account.
- [R6] The system must allow a third party company to access to its account.
- [R11] The system must allow a third party company to subscribe to an individual health status and location.
- [R12] The system must allow a third party company to subscribe to data of an anonymized group of individuals
- [G5] Provide a service capable to send a notification to the health-care service when any individual's parameter is out of range.
- [D1] TrackMe guarantees that the wearables can provide sufficient accuracy and sensitivity when monitoring individuals.
- [D3] TrackMe devices are up and running during monitoring.
- [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
- [D5] The provided SSN by the individual is valid and trustable.
- [D6] Out of coverage scenarios cannot be handled by ASOS so as to guarantee a 5 seconds reaction.
- [R13] The system must be able to send a request for monitoring an individual's data when he/she is older than 60 years old.
- [R14] The system must be able to monitor, and compare against defined thresholds, the health status of an individual.
- [R15] The system must be able to contact the health-care service associated to an individual.
- [G6] Run organizers could define the path for a given run and TrackMe users can enroll to it.
- [D1] TrackMe guarantees that the wearables can provide sufficient accuracy and sensitivity when monitoring individuals.
- [D2] TrackMe addresses data protection and integrity against possible attacks.

- [D3] TrackMe devices are up and running during monitoring.
- [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
- [D7] The organizers meet all the legal requirements and permissions necessary to set up a run.
- [R16] The system must allow a participant to register a new account.
- [R17] The system must allow a participant to access to their account.
- [R18] The system must allow an organizer to register a new account.
- [R19] The system must allow an organizer to access to their account.
- [R20] The system must allow an organizer to create a race event.
- [R21] The system must allow an organizer to define the running circuit of a race event.
- [R22] The system must allow an organizer to send invitations to participants to enroll in a race event.
- [R23] The system must allow a participant to accept or reject an invitation to a race.
- [G7] Run spectators could track the position of all the runners during a race.
- [D1] TrackMe guarantees that the wearables can provide sufficient accuracy and sensitivity when monitoring individuals.
- [D2] TrackMe addresses data protection and integrity against possible attacks.
- [D3] TrackMe devices are up and running during monitoring.
- [D4] The data collected is directly related to the individuals' by their SSN and is structured according to the data scheme required by D4H.
- [R16] The system must allow a participant to register a new account.
- [R17] The system must allow a participant to access to their account.
- [R18] The system must allow an organizer to register a new account.
- [R19] The system must allow an organizer to access to their account.
- [R20] The system must allow an organizer to create a race event.
- [R23] The system must allow a participant to accept or reject an invitation to a race.
- [R24] The system must allow any spectator of a run to view in a map the participants' location
- [R25] The system must allow a spectator to click on a participant location in order to view his/her health status.

### 3.2.1 Use case diagrams

In the current section the use cases for all the subsystems are shown.

- **Data4Help**

In the Figure 3.13 the Data4Help use cases are shown. The most important use cases are *Manage requests*, *Access individual data*, *Access bulk data*, and *Send a request*. As shown in the figure, the actors related to this system are the *Individual*, who is the user of the TrackMe wearable device, and the *Third party company*, which is the actor who will access the individual's data.

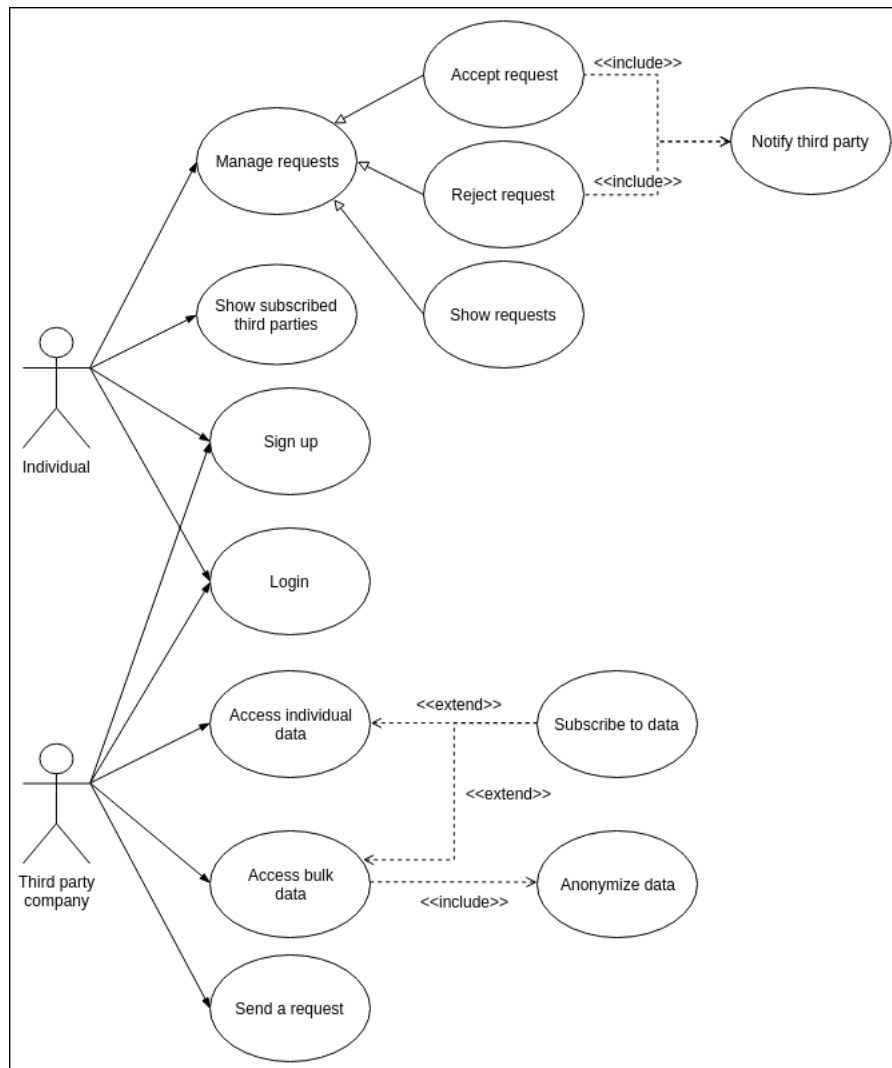


Figure 3.13: Data4Help Use Case Diagram

- **AutomatedSOS**

In the Figure 3.14 AutomatedSOS use cases are shown. In the case of ASOS, the system has two passive actors, one of them is the health-care service, that is affected by the use case *Contact health-care service*, and D4H, which is affected

by the use case *Send request for subscription*. On the other hand, the use case *Receive individual's data* is affected by D4H, since it is that service which sends the individual's information to ASOS.

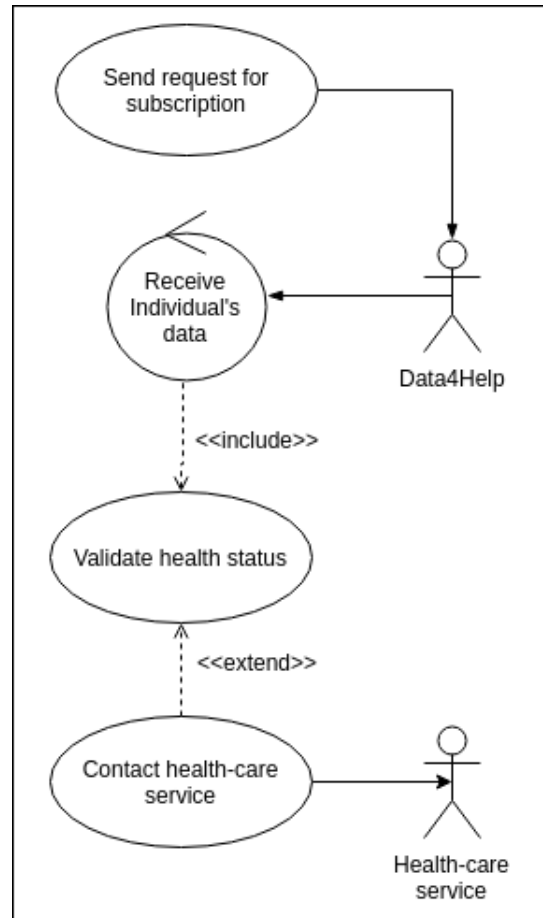


Figure 3.14: AutomatedSOS Use Case Diagram

- **Track4Run**

In the Figure 3.15 the Track4Run use cases are shown. The most important use cases are *Manage invitations*, and *Create a run*. In this system, the actors are the *Participant*, who is the TrackMe wearable device user, the *Organizer*, who will setup the run, and the *Spectators*, who are the audience that may view the location of the participants during a given run.

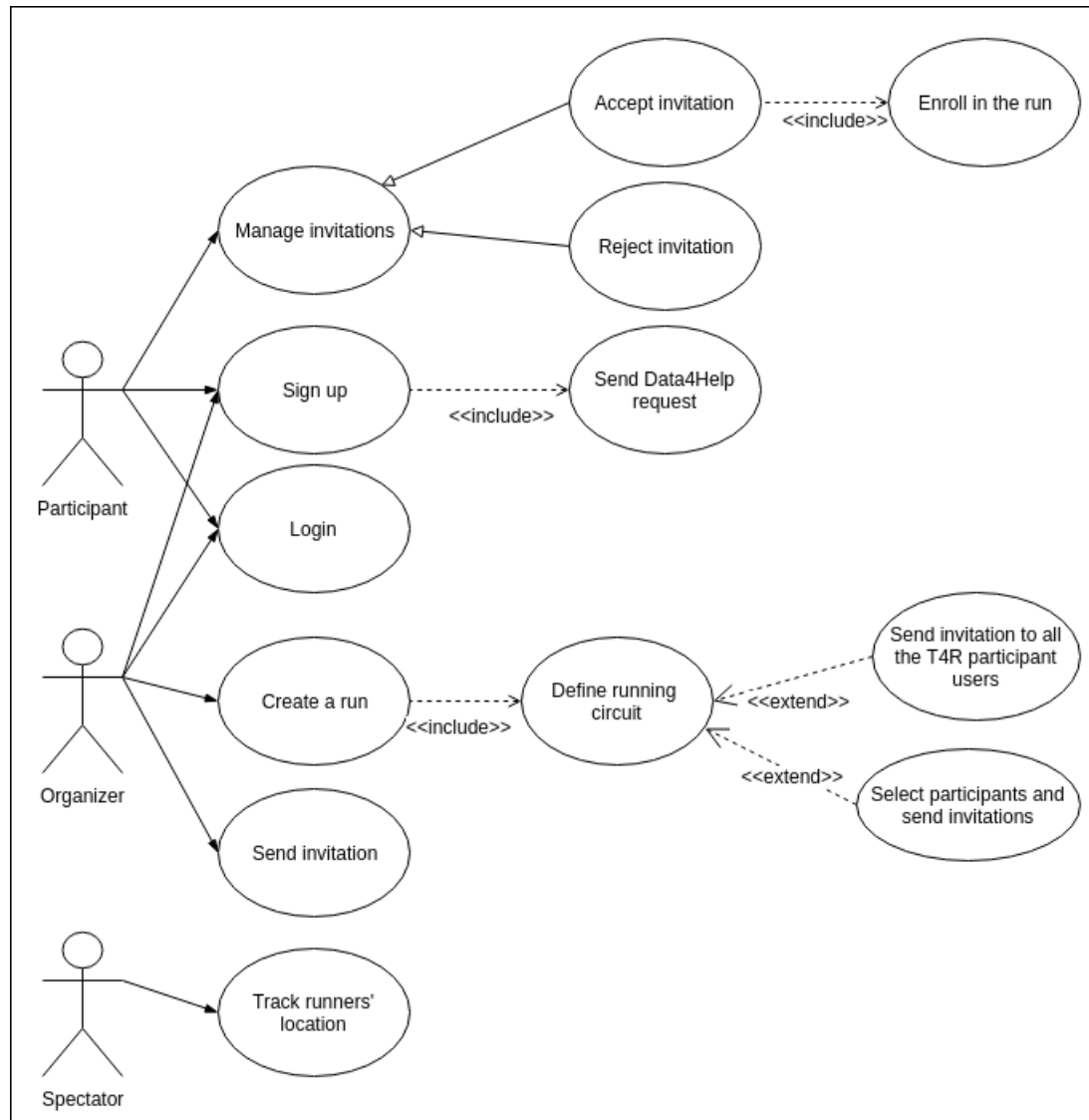


Figure 3.15: Track4Run Use Case Diagram

### 3.2.2 Use cases description

In the following section a description of each use case is provided. For every use case, an ID is defined, as well as the entry conditions, steps to accomplish the exit condition and any exception that may occur. As in previous sections, all the use cases are described based on the subsystem they belong to.

- **Data4Help**

**ID:** [UC1]

**Name:** Manage requests

**Actor:** Individual

**Entry conditions:**

1. A logged in individual

**Event flow:**

1. The individual clicks on Requests button
2. The system shows the page with a list of all pending request for the individual (**Show requests** [UC2] )
3. The individual selects the desired action (accept or reject) and clicks the Submit button
4. The system **Accepts the request** ( [UC3] ) or **Rejects the request** ( [UC4] )

**Exit conditions:**

- The request is set as accepted or rejected

**Exceptions:** -

---

**ID:** [UC2]**Name:** Show requests**Actor:** Individual**Entry conditions:**

1. A logged in individual
2. A request to access the individual's health status and location

**Event flow:**

1. The system gets all the pending requests
2. The system shows all the pending requests to the individual

**Exit conditions:** -**Exceptions:**

1. If there are no pending requests, the system should show a message to the user

---

**ID:** [UC3]**Name:** Accept the request**Actor:** Individual**Entry conditions:**

1. A logged in individual
-



2. A request to access the individual's health status and location
3. The individual clicked on the submit button to accept a request

**Event flow:**

1. The system sets the pending request as Accepted
2. The system **Notifies the third party** ( [UC5] )

**Exit conditions:**

1. The request is set to Accepted

**Exceptions:** -

---

**ID:** [UC4]**Name:** Reject the request**Actor:** Individual**Entry conditions:**

1. A logged in individual
2. A request to access the individual's health status and location
3. The individual clicked on the submit button to reject a request

**Event flow:**

1. The system sets the pending request as Rejected
2. The system **Notifies the third party** ( [UC5] )

**Exit conditions:**

1. The request is set to Rejected

**Exceptions:** -

**Scenario:** A TrackMe device user, who is already registered in D4H, has receive a notification in his cellphone saying that a company he never heard of wants to access his health data and location. Since it is an unknown company, he logs in into D4H website, goes to the Manage Requests sections, and rejects the suspicious request.

---

**ID:** [UC5]**Name:** Notify third party

---

**Actor:** Individual

**Entry conditions:**

1. A logged in individual
2. A request is set to Accepted or Rejected

**Event flow:**

1. The system adds a notification of the individual's decision to the third party company

**Exit conditions:**

1. The system shows a new notification in the Third party company dashboard

**Exceptions:** -

---

**ID:** [UC6]

**Name:** Sign up

**Actor:** Individual, Third party company

**Entry conditions:** -

**Event flow:**

1. The Individual or the Third party company enters to the sign up page of the website
2. The Individual or the Third party company completes all the mandatory fields
3. The Individual or the Third party company clicks on the "Get started" button

**Exit conditions:**

1. The system creates an account for an Individual or a Third party company

**Exceptions:**

1. If either the Individual or the Third party company are already registered, an error message will be shown
2. If there some of the fields in the registration form that are not filled in, an error message will be shown

---

**ID:** [UC7]

**Name:** Login

---

**Actor:** Individual, Third party company

**Entry conditions:** -

**Event flow:**

1. The Individual or the Third party company enters to the login page of the website
2. The Individual or the Third party company completes the email address and password fields
3. The Individual or the Third party company clicks on the "Log in" button

**Exit conditions:**

1. The system redirects the Individual or a Third party company to the profile page

**Exceptions:**

1. If the credentials are wrong, an error message will be shown
2. If the password or the email are missing, an error message will be shown

---

**ID:** [UC8]

**Name:** Send a request

**Actor:** Third party company

**Entry conditions:**

1. The Third party company is logged in

**Event flow:**

1. The Third party company clicks on the Requests button
2. The system shows a form with a filter criteria, which contains the SSN field
3. The Third party company fills in the SSN field and clicks submit

**Exit conditions:**

1. The system sends a notification to the Individual

**Exceptions:**

1. If the SSN is not valid, an error message will be shown

2. If the SSN does not corresponds to an active Individual, an error message will be shown

**Scenario:** The company HelpMe needs updates on the health status of one of its clients in order to offer a particular service. To be able to access the data of the client, makes a request for accessing the health data and location of the user, using his SSN. To do so, an employee logs in into D4H, and sends a request to the client using the D4H website.

---

**ID:** [UC9]

**Name:** Access individual data

**Actor:** Third party company

**Entry conditions:**

1. The Third party company is logged in
2. The Third party has sent a request to access an specific Individual health status and location

**Event flow:**

1. The Third party company clicks on the View data button
2. The system shows a form with a filter criteria, which contains the SSN field
3. The Third party company fills in the SSN field and clicks submit
4. The system gets the health status and location of the given SSN
5. The system shows the health status and location of the Individual
6. The third party company may click on Subscribe, which extends to the use case [UC10]

**Exit conditions:**

1. The system shows the health status and location of the given Individual

**Exceptions:**

1. If the SSN is not valid, an error message will be shown
  2. If the SSN does not corresponds to an active Individual, an error message will be shown
  3. If the SSN does not corresponds to an Individual who accepted the previously sent request, an error message will be shown
-

---

**ID:** [UC10]

**Name:** Subscribe to data

**Actor:** Third party company

**Entry conditions:**

1. The Third party company is logged in
2. The Third party has made a search
3. The Third party company clicks on the Subscribe button

**Event flow:**

1. The system creates a subscription between the Third party company and saved query, that can be an specific SSN or a more general query

**Exit conditions:**

1. The system starts to notify the Third party company of new incoming data of the Individual or the anonymized group of individuals

**Exceptions:** -

**Scenario:** A company needs to subscribe to the health data of one of its users. To do so, an employee logs in into D4H website, looks for the users health data and location using its SSN, and subscribes to the query. Once the employee does that, the company will receive any update of its user's health status and location.

---

**ID:** [UC11]

**Name:** Access bulk data

**Actor:** Third party company

**Entry conditions:**

1. The Third party company is logged in

**Event flow:**

1. The Third party company clicks on the View data button
  2. The system shows a form with different filter criteria, which contains country, age, gender and blood type fields
  3. The Third party company selects the different filters
  4. The system **Anonymize the data**( [UC12] )
-

5. The system shows the health status and location of the anonymized group of individuals
6. The third party company may click on Subscribe, which extends to the use case [UC10]

**Exit conditions:**

1. The system shows the health status of the anonymized group of individuals

**Exceptions:**

1. If the data cannot be anonymized, an error message will be shown
2. If there is not enough data, an error message will be shown

---

**ID:** [UC12]**Name:** Anonymize data**Actor:** Third party company**Entry conditions:**

1. The Third party company is logged in
2. The Third party company has made a search for a bulk of data

**Event flow:**

1. The system gets the data based on the filters the Third party company has provided
2. The system anonymize the data, by removing the location of each user

**Exit conditions:**

1. The system returns the anonymized data

**Exceptions:**

1. If there are less than 1000 individuals in the request of data, an error is returned
2. If there is not enough data, an error is returned

---

**ID:** [UC28]**Name:** Show subscribed third parties**Actor:** Individual**Entry conditions:**

---

1. The Individual is logged in

**Event flow:**

1. The Individual clicks on "Subscribed companies" button
2. The system shows a web page with all the Third party companies that have subscribed to the Individual's data

**Exit conditions:**

1. The system redirects the current Individual to the "Subscribed companies" section

**Exceptions:**

1. If there are no subscribed third party companies, a message is shown

**• AutomatedSOS**

**ID:** [UC13]

**Name:** Send request for subscription

**Actor:** D4H

**Entry conditions:**

- The system knows the SSN of the elderly individual to be subscribed

**Event flow:**

1. The system sends a request to Data4Help in order to subscribe to the health status and location of the given individual
2. Data4Help sends an OK response to advice the system that the request was sent

**Exit conditions:**

1. Data4Help placed a request for subscription to the individual

**Exceptions:**

---

**ID:** [UC14]

**Name:** Receive individual's data

**Actor:** D4H

**Entry conditions:**

- The individual is subscribed to ASOS
-

**Event flow:**

1. Every time the health status and location of the subscribed individual, D4H sends the data to ASOS
2. The system **Validate the health status** ( [UC() 15]) of the individual

**Exit conditions:****Exceptions:**

---

**ID:** [UC15]**Name:** Validate health status**Actor:** -**Entry conditions:**

- The individual is subscribed to ASOS
- ASOS has received new health data of the individual
- The system has several thresholds configured

**Event flow:**

1. The system validates the health condition of the individual using the configured thresholds
2. The system **Contacts the health-care service** ( [UC() 16]) if needed

**Exit conditions:****Exceptions:**

---

**ID:** [UC16]**Name:** Contact health-care service**Actor:** Health-care service**Entry conditions:**

- Some parameters of the health data of the individual are above or below the threshold

**Event flow:**

1. The system gets the health-care service contact of the given individual
  2. The system contacts the health-care service, and sends the health data and location of the given individual
-



**Exit conditions:**

- The system has contacted the health-care service

**Exceptions:** -

**Scenario:** An individual subscribed to ASOS service is having a hart attack. ASOS has validated the individual's health status and compared the parameters with the defined thresholds, and has decided to call the health-care service. To do so, it contacts the health-care service associated to the individual, using a previously defined protocol and API, and sends to them all the data of the individual.

---

- **Track4Run**

**ID:** [UC17]

**Name:** Sign up

**Actor:** Participant, Organizer

**Entry conditions:** -

**Event flow:**

1. The Participant or the Organizer enters to the Track4Run sign up page
2. The Participant or the Organizer fill in all the mandatory fields in the registration form
3. The Participant or the Organizer click on Register button
4. The system **Sends a Data4Help request** ( [UC19] )

**Exit conditions:**

1. The system creates an account for a Participant or an Organizer

**Exceptions:**

1. If either the Participant or the Organizer are already registered, an error message will be shown
2. If there some of the fields in the registration form that are not filled in, an error message will be shown

---

**ID:** [UC18]

**Name:** Login

**Actor:** Participant, Organizer

---

**Entry conditions:** -

**Event flow:**

1. The Participant or the Organizer enters to the login page of the website
2. The Participant or the Organizer completes the email address and password fields
3. The Participant or the Organizer clicks on the "Log in" button

**Exit conditions:**

1. The system redirects the Participant or the Organizer to the profile page

**Exceptions:**

1. If the credentials are wrong, an error message will be shown
2. If the password or the email are missing, an error message will be shown

---

**ID:** [UC19]

**Name:** Send Data4Help request

**Actor:** Participant

**Entry conditions:**

1. The Participant has already registered

**Event flow:**

1. The system sends a notification to Data4Help in order to request health status and location of the user
2. Data4Help sends an OK response to advice the system that the request was sent

**Exit conditions:**

1. The Participant received a notification from Track4Run

**Exceptions:** -

---

**ID:** [UC20]

**Name:** Manage invitations

**Actor:** Participant

**Entry conditions:**

---

1. The Participant is logged in

**Event flow:**

1. The Participant clicks on Invitations button
2. The system shows all the run invitations to the user
3. The Participant clicks either on **Accept invitation** ( [UC21] ) or **Reject invitation** ( [UC22] )

**Exit conditions:**

1. The system shows the user the pending invitations

**Exceptions:** -

---

**ID:** [UC21]**Name:** Accept invitation**Actor:** Participant**Entry conditions:**

1. The Participant is logged in
2. The Participant has clicked on Accept invitation

**Event flow:**

1. The system sets the invitation as Accepted
2. The system **Enrols the participant into the run** ( [UC23] )
3. The system shows all the run invitations to the user

**Exit conditions:**

1. The Participant is enrolled into the run

**Exceptions:** -

---

**ID:** [UC22]**Name:** Reject invitation**Actor:** Participant**Entry conditions:**

1. The Participant is logged in
  2. The Participant has clicked on Reject invitation
-

**Event flow:**

1. The system sets the invitation as Rejected
2. The system shows all the run invitations to the user

**Exit conditions:** -**Exceptions:** -

---

**ID:** [UC23]**Name:** Enrol in the run**Actor:** Participant**Entry conditions:**

1. The Participant is logged in
2. The Participant has clicked on Accept invitation

**Event flow:**

1. The system adds the Participant into the participants list of the run
2. The system shows the Participant a confirmation message

**Exit conditions:**

1. The Participant is enrolled into the run

**Exceptions:** -

---

**ID:** [UC24]**Name:** Create a run**Actor:** Organizer**Entry conditions:**

1. The Organizer is logged in

**Event flow:**

1. The Organizer clicks on the "Create a run" button
  2. The system shows a form with the following fields: Name, Start day and End time
  3. The Organizer fills in all the fields and clicks the Create button
  4. The system shows the **Define running circuit** ( [UC25] ) web page
-

**Exit conditions:**

1. The system creates the run

**Exceptions:**

1. If the Start date field is less than "Today", an error will be shown

---

**ID:** [UC25]**Name:** Define running circuit**Actor:** Organizer**Entry conditions:**

1. The Organizer is logged in
2. The Organizer has created a run

**Event flow:**

1. The system shows the Running circuit web page
2. The Organizer defines the running circuit by clicking on the map
3. Once the Organizer has finished defining the running circuit, clicks on Submit button
4. The system shows a message asking if the Organizer wants to send invitations or not
5. The system extends either [UC29] or [UC30]

**Exit conditions:**

1. The system creates the running circuit and associate it with the run event

**Exceptions:** -

---

**ID:** [UC26]**Name:** Send invitation**Actor:** Organizer**Entry conditions:**

1. The Organizer is logged in

**Event flow:**

1. The Organizer clicks on "Send invitations" button
2. The system shows a list with all pending running events
3. The Organizer clicks on one running event
4. The system shows a list of all not-enrolled, Track4Run Participants
5. The Organizer selects the Participants he/she wants to invite, and clicks the Invite button

**Exit conditions:**

1. The system sends invitations to all the selected Participants

**Exceptions: -**

---

**ID:** [UC27]**Name:** Track runners' location**Actor:** Spectator**Entry conditions:**

1. The Spectator is in the Track4Run site

**Event flow:**

1. The Spectator clicks on "Current events" button
2. The system shows a list of all running events which has started but has not finished
3. The Spectator selects one of the running events
4. The system shows a map with all the Participants' location

**Exit conditions: -****Exceptions: -**

**Scenario:** A marathon is taking place in the city, and a runner's family want to cheer he during the race. Since they are late and the event has already begun, they enter to the T4R website and go to the Current events web page, select the event and look for their family member.

---

**ID:** [UC29]**Name:** Send invitation to all the T4R participant users**Actor:** Organizer**Entry conditions:**

1. The Organizer has defined a running circuit of a race event

**Event flow:**

1. The system sends an invitation to all the T4R participant users

**Exit conditions:**

1. The system redirects the Organizer to the dashboard page of the web site

**Exceptions:**

1. If an error occurs, the system will show an error message

---

**ID:** [UC30]

**Name:** Select participants and send invitations

**Actor:** Organizer

**Entry conditions:**

1. The Organizer has defined a running circuit of a race event

**Event flow:**

1. The system shows a list with all the T4R Participant users
2. The Organizer filters the Participant users using some of the following filters:  
Min Age, Max Age, Country and City
3. The system shows the Participants that match the selected filter
4. The Organizer selects some Participants and clicks the Invite button

**Exit conditions:**

1. The system sends invitations to all the selected Participants

**Exceptions:**

1. If an error occurs, the system will show an error message

### 3.2.3 Activity diagrams

In order to describe in detail the flow followed by each one of the offered services, activity diagrams are highly useful.

For **D4H**, it is important to specify the steps and conditions involved in giving access to data requested by third parties (Figure 3.16).

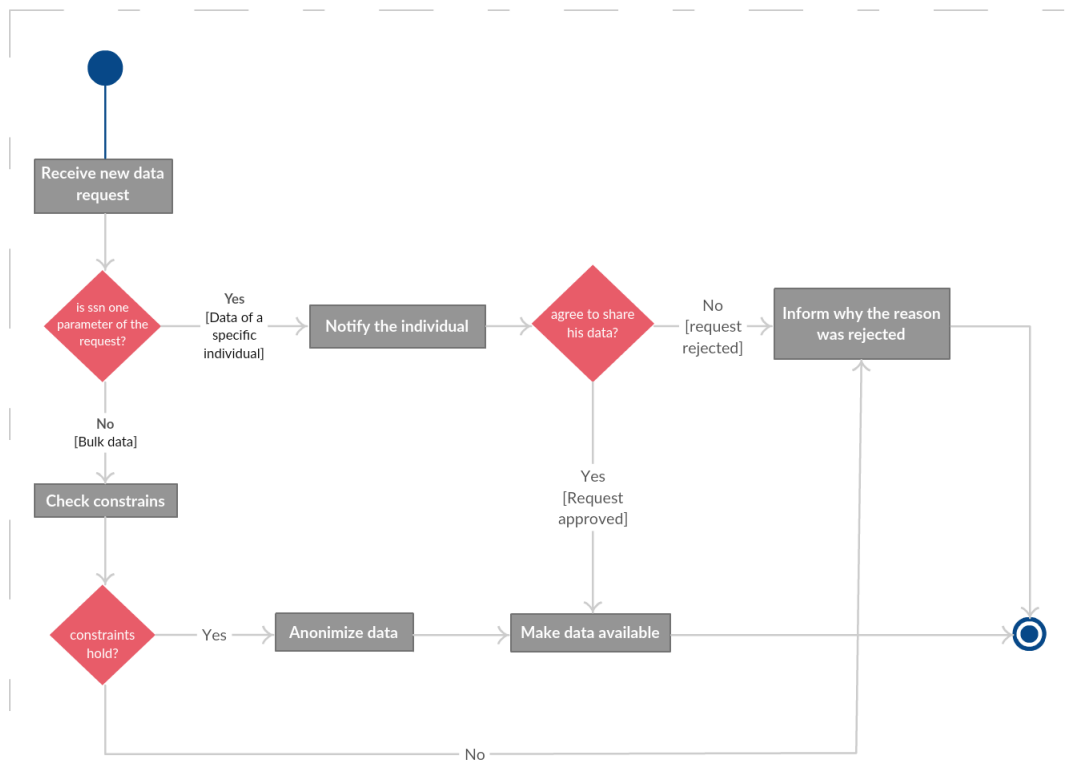


Figure 3.16: Data4Help - Provide Data To Third Parties Activity Diagram

If an individual (older than 60 years) activates ASOS service, D4H sends to a predefined URL, provided by ASOS, the data of that user immediately after it is received from the devices. ASOS compares the parameters with their corresponding threshold, and if the current status of the individual is *stable*, a notification is send to his associated health-care service. On the other hand, if the current status is *critical*, means that a notification was send previously, so the system goes back to a monitoring activity.



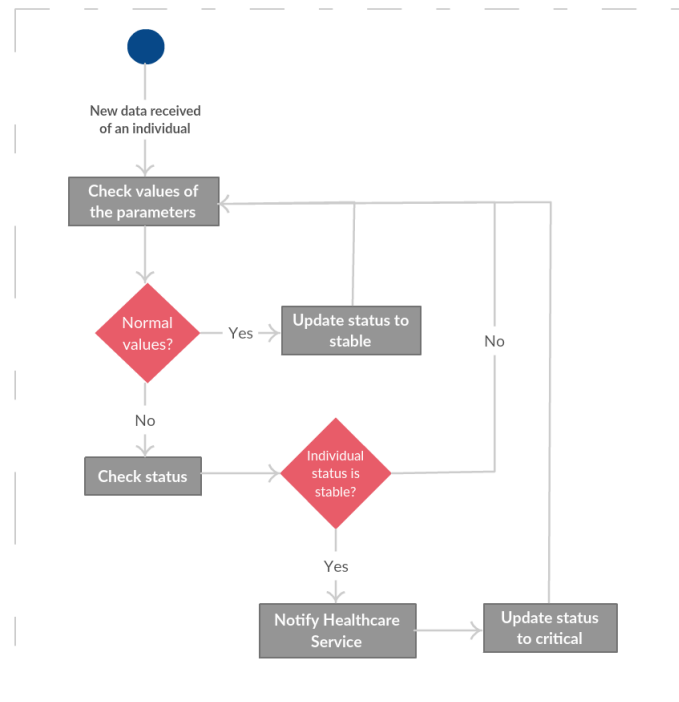


Figure 3.17: AutomatedSOS - Monitoring Individuals Activity Diagram

In **T4R**, there are three main activities involving each type of actor in the system. Organizers can *configure a run* and *send invitations to individuals* (Figure 3.18), invited individuals can *enroll into a run* (Figure 3.19) and spectators can *track participants of an active run* (Figure 3.20).

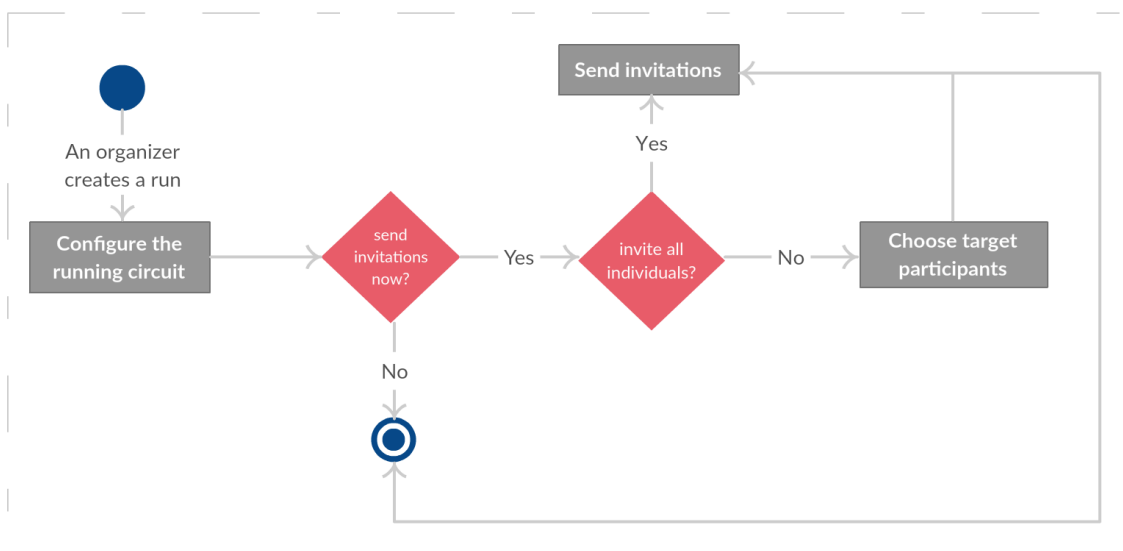


Figure 3.18: Track4Run - Create a Run and Invite Individuals Activity Diagram

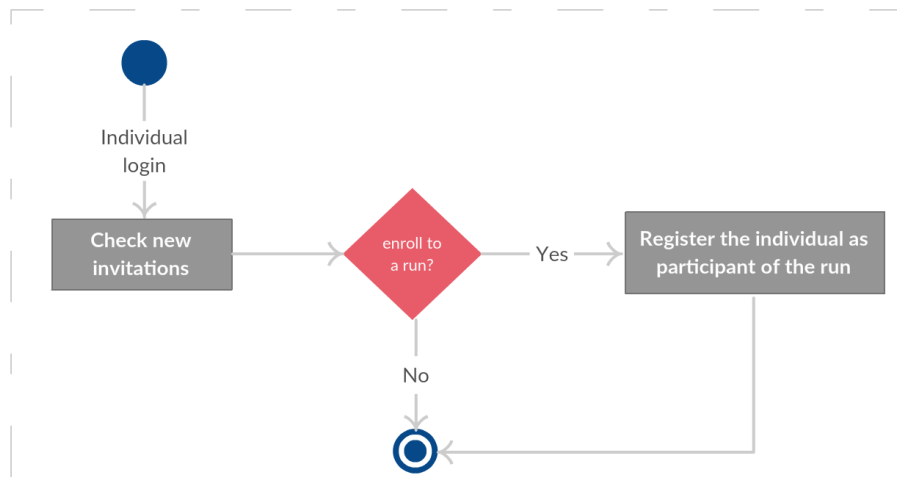


Figure 3.19: Track4Run - Enroll into a Run Activity Diagram

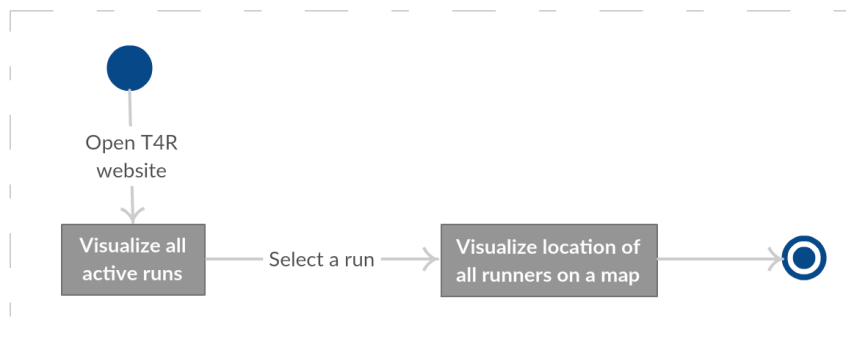


Figure 3.20: Track4Run - Track Runners Activity Diagram

### 3.2.4 Sequence diagrams

In the following section three sequence diagrams are described. All of them, are an important part of the system and represent the following processes: *accept a request* for accessing the individual's health status and location, *search and subscribe to a bulk of data*, and *accept an invitation* to enroll in a race.

To begin with, the *accept a request* for accessing the individual's health status and location is shown in the Figure 3.21. It is possible to observe the Manage Requests use case ( [UC1] ) flow, where the Individual asks D4H to get his/her pending requests, and the interaction between D4H and the Request collection, which returns the pending requests to the Individual. Also, once the Individual has accepted a request, it changes its status to *Accepted* and the Third party company is notified.

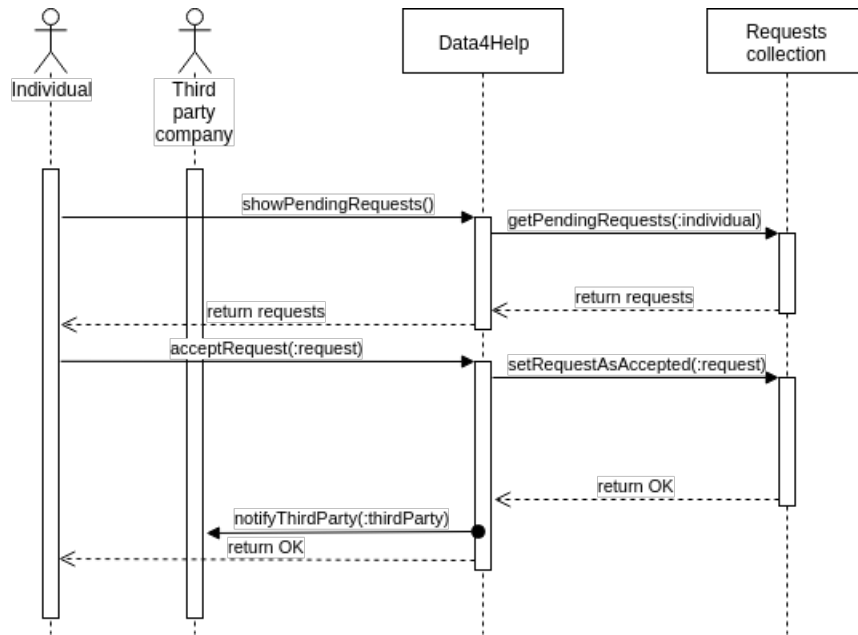


Figure 3.21: Data4Help - Accept Request Sequence Diagram

In the Figure 3.22 can be observed the sequence diagram of Access bulk data use case ( [UC11] ) and the optional Subscribe to data use case ( [UC10] ). Also in this case, it is possible to notice the action taken by the actor (the Third party company) to get data from D4H. The *search* action makes D4H to look for the data in the Data collection, anonymize it, and return it to the actor. Optionally, the Third party company can save and subscribe to the previously used query in order to get updates.

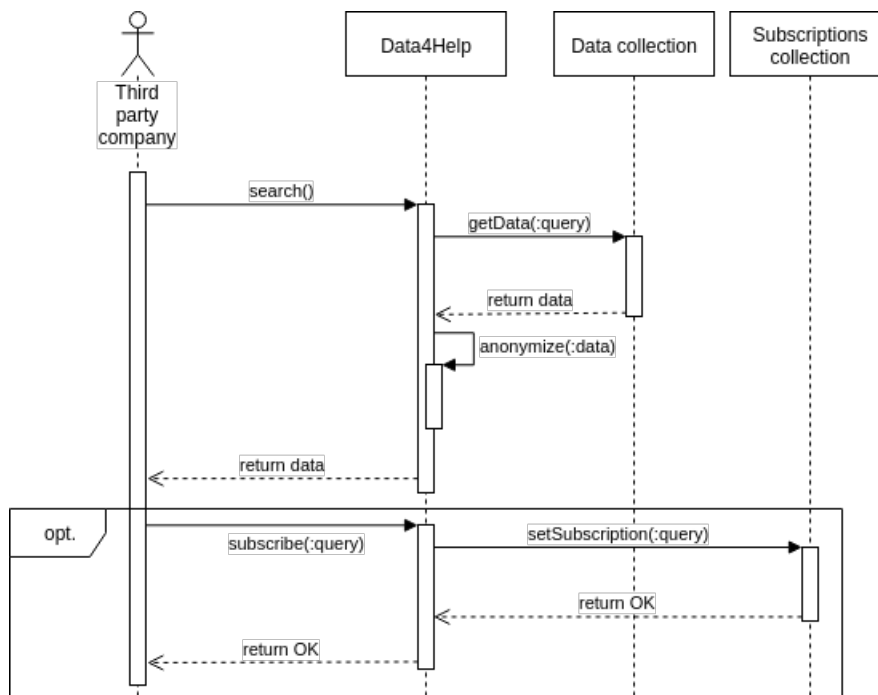


Figure 3.22: Data4Help - Access to bulk data and subscribe Sequence Diagram

Finally, the Accept invitation to a run use case ( [UC21] ) is modelled in the Figure 3.23. In this case, the Participant asks T4R to show his/her pending invitations, which are obtained from the Invitations collection. Once the Participant has all the pending invitations, he/she can select one and accept the invitation, action that will enrol the Participant to the given run, and set the invitation as *Accepted*.

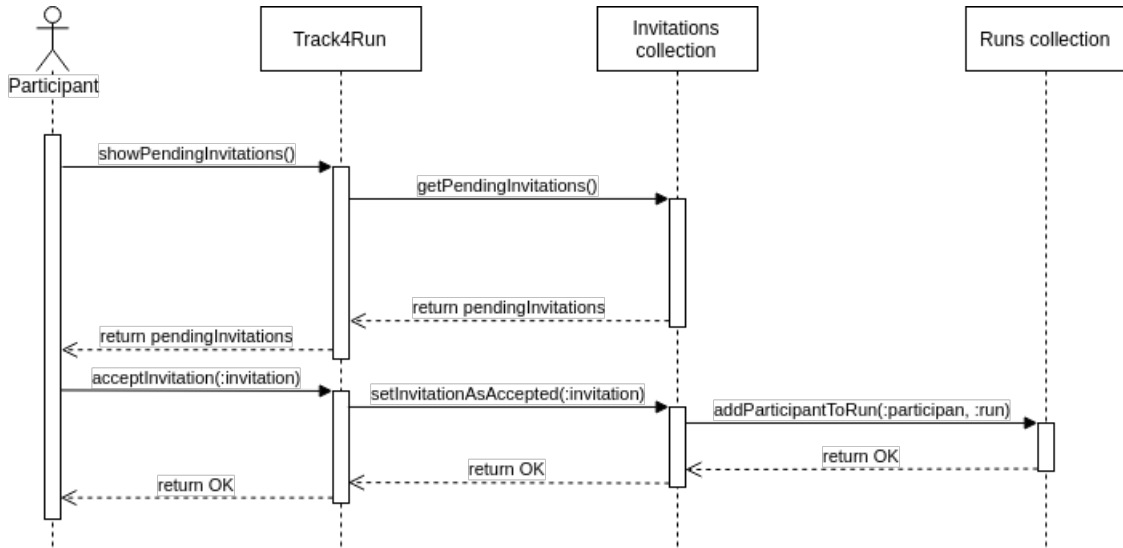


Figure 3.23: Track4Run - Accept Invitation Sequence Diagram

### 3.2.5 Requirements traceability matrix

Goal Id	Requirement Id	Use case Id	Comments
[G1]	[R1]	[UC6]	
	[R2]	[UC7]	
	[R3]	[UC1]	
		[UC2]	
		[UC3]	
		[UC4]	
		[UC5]	
	[R4]	[UC9]	
[G2]	[R5]	[UC6]	
	[R6]	[UC7]	
	[R8]	[UC9]	
	[R7]	[UC8]	
[G3]	[R5]	[UC6]	
	[R6]	[UC7]	
	[R9]	[UC11]	

[G4]		[UC12]	
	[R10]	[UC12]	
	[R5]	[UC6]	
	[R6]	[UC7]	
	[R11]	[UC9]	
		[UC10]	
	[R12]	[UC10]	
		[UC11]	
		[UC12]	
[G5]	[R13]	[UC13]	
	[R14]	[UC14]	
		[UC15]	
	[R15]	[UC16]	
[G6]	[R16]	[UC17]	
		[UC19]	
	[R17]	[UC18]	
	[R18]	[UC17]	
	[R19]	[UC18]	
	[R20]	[UC24]	
	[R21]	[UC25]	
	[R22]	[UC26]	
		[UC29]	
		[UC30]	
	[R23]	[UC20]	
		[UC21]	
		[UC22]	
		[UC23]	
[G7]	[R16]	[UC17]	
		[UC19]	
	[R17]	[UC18]	
	[R18]	[UC17]	
	[R19]	[UC18]	
	[R20]	[UC24]	
	[R23]	[UC20]	
		[UC21]	
		[UC22]	

		[UC23]	
	[R24]	[UC27]	
	[R25]	[UC27]	

Table 3.1: Traceability matrix

### 3.3 Performance requirements

The system is provided to serve all the trackme wearable device users at the same time simultaneously. It guarantees high level performance and maintenance operation costs as low as possible. Technical support for installation is not required, as it is only a web application and users can access it from anywhere where the internet service is provided. Additionally, under ASOS service, the system makes sure the system responds and notifies the health-care system within 5 seconds, when the thresholds are below the certain defined level. Users will rely on the application in order to organize their send or accept/reject requests for data and notifications, so we have to guarantee quick, reactive and correct response.

### 3.4 Design constraints

#### 3.4.1 Standards compliance

- The application is responsive that makes web pages render well on a variety of devices and window or screen sizes. Content, design and performance are necessary across all devices to ensure usability and satisfaction.
- The application complies with the web standards of the World Wide Web Consortium (W3C).
- As the Web provides many mechanisms to interlink data across systems, the system keep user's personal information private and fragmented.
- The system correctly preserves and restores user or application state within the session. The application preserves user or state when leaving the web browser and prevents accidental data loss due to back-navigation and other state changes. When returning to the web page, the application must restore the preserved state and any significant stateful transaction that was pending.

### 3.4.2 Hardware limitations

This is a web based application with no strict hardware limitations. The only requirement is a wearable device and its app that posts the data when online. Users can access the application from anywhere, from any system using the internet service.

### 3.4.3 Any other constraint

- **Regulatory Policies**

The application will have to ask for user's position and personal data in order to have track of the health status. Social security numbers(SSN) and email addresses won't be used for commercial uses.

Moreover, the system makes sure the third-party companies who registers are legally established. Thus, there are no fake companies who tries to exploit user's data.

## 3.5 Software system attributes

### 3.5.1 Reliability

D4H, T4R and ASOS infrastructure must guarantee at least two database mirrors and three server instances up. This way, any failure in the database or in the instances will keep the whole system up and running.

### 3.5.2 Availability

Since ASOS depends on D4H, both systems must be available 99.9% of the time. On the other hand, T4R must have an up time of 95%.

### 3.5.3 Security

D4H and T4R databases must have different users and password and the access from outside the servers' network must not be allowed.

Furthermore, D4H must keep track of every request made by the third parties, and must log every access to the third party users' account or individuals' account. Moreover, ASOS must have log every time the system contacts a health-care service, specifying the individual's ID, the contacted health-care service and the individual's health status.

Finally, all the connections to the TrackMe services and from the TrackMe services should be encrypted using PKI (HTTP over SSL).

### **3.5.4 Maintainability**

D4H, T4R, ASOS systems must maintain a service log in order to have a very detailed sequence of steps when a bug is found.

A software developer, shall be able to add a new product feature, including source code modifications and testing. The test cases of the three systems should have a 90% of code coverage.

### **3.5.5 Portability**

D4H, ASOS and T4R, should be written in a way that, moving from one hosting service to another, should not affect the service and the code of the system.

### **3.5.6 Usability**

D4H and T4R must function properly in the following Internet browsers: Google Chrome version 31 or higher, Mozilla Firefox version 26 or higher, Internet Explorer Edge or higher. In addition, both websites must be responsive so users should be able to use them from a smartphone or tablet without inconvenience.

There is no specific usability requirement for ASOS.



---

# Formal analysis using Alloy

---

As any real-life system, all the services provided by TrackMe are described by a set of essential properties and constraints. All of them have been defined in the basis of the problem domain and the needs of its users. To check whether or not the properties will be satisfied by the system and the constraints will never be violated, it is useful to make a formal analysis building an alloy model.

In this case, D4H, ASOS and T4R will have a corresponding model with different purposes according to what is important given each service.

## 4.1 Data4Help

### 4.1.1 Purpose of the model

The purpose of the following model is to formalize the Goal 2. In this case, the alloy model defines the following objects: *Status*, which can be *Approved*, *Pending* or *Rejected*, *NotificationType*, which can be *ApprovedNotification* or *RejectedNotification*, *ThirdParty*, *Individual*, *Request*, *Query*, and *QueryResponse*. Individual objects contain a set of Request objects, which relate them with the requests that were sent to them by ThirdParty objects. A Request object, relates a ThirdParty with an Individual, and has a Status and a NotificationType, which is the notification sent to the ThirdParty when the Individual approved or rejected the Request. Finally, the Query objects relate a Third Party with an Individual, and represent the action of inquiry the health status and location of an Individual.

There are several facts that must be hold when creating an Alloy world. The most important ones are those related to the Query and QueryResponse, in which there cannot exist a QueryResponse related to a Query that was created by a ThirdParty, who is trying to acquire information of an Individual whose Request, for that ThirdParty, has the status Pending or Rejected. On the other hand, there should be one and only one Request that relates a ThirdParty with an Individual, so a ThirdParty cannot send more than one Request to an Individual.

## 4.1.2 Alloy model

```

open util/integer

// declares a set Status with 3 elements
abstract sig Status {}
one sig Approved, Pending, Rejected extends Status{}

// declares a set NotificationType with 2 elements
abstract sig NotificationType{}
one sig ApprovedNotification, RejectedNotification extends NotificationType{}

sig ThirdParty{}

sig Individual{
    requests: some Request
}

sig Request{
    company: one ThirdParty,
    individual: one Individual,
    status: one Status,
    notification: lone NotificationType
}

sig Query{
    company: one ThirdParty,
    individual: one Individual
}

sig QueryResponse{
    query: one Query
}

// An individual is an individual of its request
fact allRequestsShouldBeRelatedToOneIndividualAndMustBeInItsSet {
    (all i: Individual, r: i.requests | r.individual = i) and
    (all r: Request, i: Individual | r.individual = i and r in i.requests)
}

/*
 * There should be 1 request per individual and third party.
 * There shouldn't exist 2 requests for the same individual and same third party.
 */
fact noCommonRequestsWithSameIndividualAndSameThirdParty {
    no disj r1,r2: Request |
        r1.company = r2.company and r1.individual = r2.individual
}

/*
 * All approved request must have an approved notification.
 * All rejected request must have a rejected notification.
 * All pending request must have NO notification.
 */
fact allRequestsShouldHaveAppropriateNotification{
    all r: Request |

```

```

        (r.notification = ApprovedNotification implies r.status = Approved) and
        (r.notification = RejectedNotification implies (r.status = Rejected)) and
        (r.notification = none implies (r.status = Pending))
    }

    /*
     * All queries should have one third party and one individual.
     * We don't care here if the request was approved or rejected.
     */
    fact allQueriesShouldBeRelatedToOneRequestRelation {
        all q: Query | some r: q.individual.requests |
            (r.individual = q.individual and
             r.company = q.company)
    }

    /*
     * All query responses should be related to a query with an accepted request
     * this means that if the request was rejected, and a third party makes a
     * query asking for an individual's data, there should not be a response.
     * Otherwise, if the individual approved the request, there should be a
     * response.
     */
    fact allQueryResponsesShouldHaveQueryWithAnAcceptedRequest {
        all qr: QueryResponse, q: qr.query | some r: q.individual.requests |
            (r.status = Approved and
             r.individual = q.individual and
             r.company = q.company)
    }

    // There should be 1 query response per query
    fact noCommonQueryResponseBetweenTwoQueries {
        no disj qr1, qr2: QueryResponse | qr1.query = qr2.query
    }

    // Every QueryResponse should have a query associated to an approved request
    fact allQueryResponsesAssociatedToAQueryShouldBeRelatedToAnApprovedRequest {
        some q: Query, qr: QueryResponse | some r: q.individual.requests |
            qr.query = q and
            r.individual = q.individual and
            r.company = q.company and
            r.status = Approved
    }

    /*
     * All queries that relates with an individual and a company with an accepted
     * request, should have a response.
     */
    fact allQueriesWithAnAcceptedRequestRelationShouldHaveAResponse {
        all q: Query, i: q.individual, c: q.company | hasApprovedRequest[i, c] implies
            (some qr: QueryResponse |
             qr.query = q and qr.query.company = c and
             qr.query.individual = i)
    }

    /*
     * Check whether the individual has a request that relates him with a company,

```

```

* and it is approved.
*/
pred hasApprovedRequest[i: Individual, c: ThirdParty]{
    one r: i.requests |
    r.company = c and
    r.individual = i and
    r.status = Approved
}

/*
* Show worlds with more than 2 request, more than 2 queries, and more than 1
* query response.
*/
pred show(){
    #Request > 2
    #QueryResponse > 1
    #Query > 2
}

/*
* Every QueryResponse should be related to an individual and a company with
* a common accepted request.
*/
assert everyAcceptedRequestShouldHaveAResponse {
    all qr: QueryResponse, q: qr.query, i: q.individual, c: q.company |
        one r: i.requests |
            r.company = c and
            r.individual = i and
            r.status = Approved
}

assert everyRejectedOrPendingRequestShouldNotHaveAResponse {
    all qr: QueryResponse, q: qr.query, i: q.individual, c: q.company |
        no r: i.requests |
            r.company = c and
            r.individual = i and
            (r.status = Pending or r.status = Rejected)
}

run show

check everyAcceptedRequestShouldHaveAResponse
check everyRejectedOrPendingRequestShouldNotHaveAResponse

```

### 4.1.3 Generated worlds

After running the model described above, different worlds were generated, two of them are:

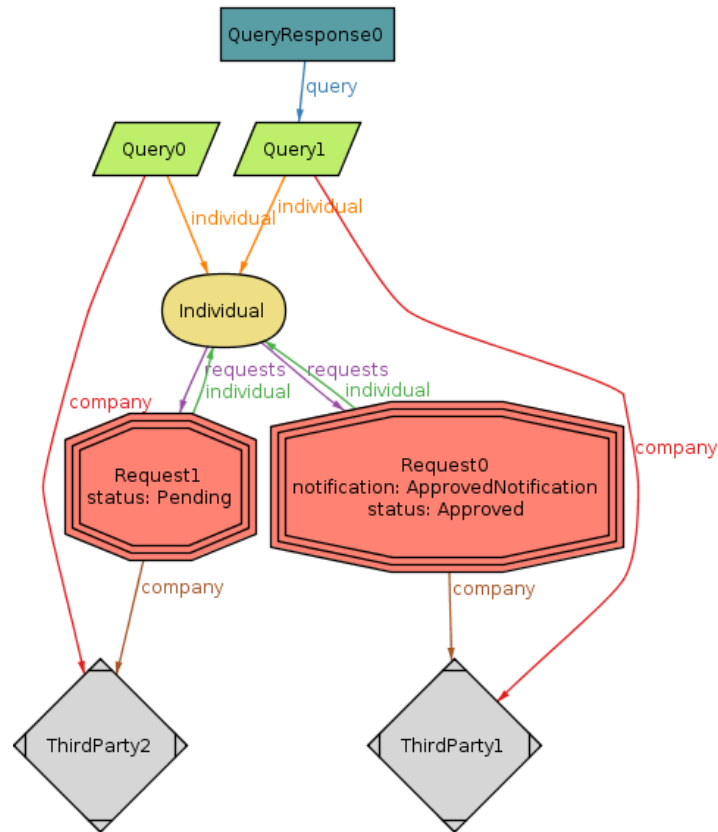


Figure 4.1: D4H - 1<sup>st</sup> generated world. It can be seen an Individual object with two Request objects of different ThirdParty objects. One Request has the status Pending, and the other one has the status Approved. The Query0, relates the ThirdParty whose Request is Pending, and so it has no QueryResponse associated. On the other hand, the Query1 object is associated to the ThirdParty whose Request was accepted, hence it has an associated QueryResponse.

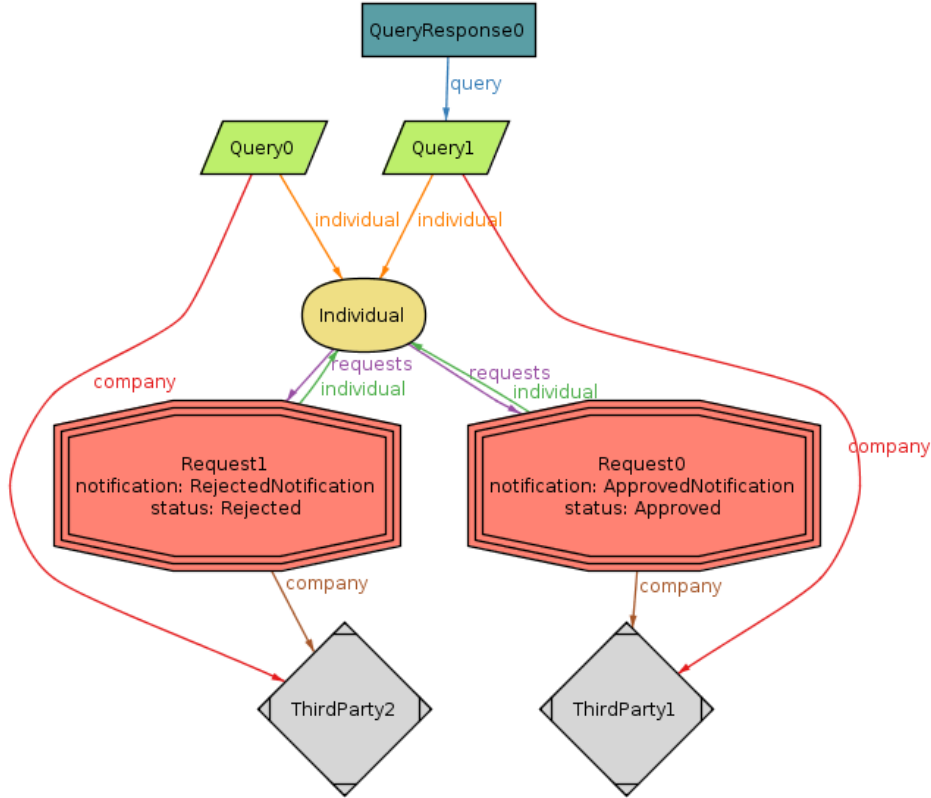


Figure 4.2: D4H - 2<sup>nd</sup> generated world. It can be seen an Individual object with two Request objects of different ThirdParty objects. One Request has the status Rejected, and the other one has the status Approved. The Query0, relates the ThirdParty whose Request is Rejected, and so it has no QueryResponse associated. On the other hand, the Query1 object is associated to the ThirdParty whose Request was accepted, hence it has an associated QueryResponse.

## 4.2 AutomatedSOS

### 4.2.1 Purpose of the model

The purpose of the following model is to formalize the Goal 5. In this case, the alloy model defines the following objects: *Individual*, *Data*, which is the list of health parameters of an Individual, *HealthcareService* and *HSNotification*. Individual objects contain a Data object, and they are related to one HealthcareService object. Data objects contain a list of parameters (health status), which for the sake of simplicity contains only one element. The HealthcareService objects contain a list of HSNotification objects, which are the notifications sent to the health-care service notifying that an Individual is in a critical situation. Finally, the HSNotification objects relates an Individual with a HealthcareService.

The facts in this model are mostly related to having HSNotification objects only when

the health status of an Individual is out of range (again, for the sake of simplicity, values below zero are considered out of normal range). Also, there cannot exist HSNotification objects that relates a HealthcareService with an Individual that does not have it as its health-care service.

## 4.2.2 Alloy model

```

open util/integer as integer

sig Individual{
  data: one Data,
  healthcareService: one HealthcareService
}

sig Data {
  parametersList: set Int, // < 0 the individual is critical
}{
  // in order to simplify the things
  #parametersList = 1
}

sig HealthcareService {
  notification: set HSNotification // any number of notifications
}{
  /* All notifications in the set should have an individual related to
   * the current health care service
   */
  all n: notification | n.individual.healthcareService = this
}

sig HSNotification {
  individual: one Individual,
  healthService: one HealthcareService
}

/* All individuals must have data associated and all data must be related
 * to an individual.
 * It makes no sense to have data with no relation to an individual
 */
fact allIndividualsMustHaveDataAssociatedAndCannotBeDataWithoutIndividual {
  (all i: Individual | some d: Data | i.data = d) and
  (all d: Data | some i: Individual | i.data = d)
}

/* All individuals must have a relation with a health care service, and all
 * the health care services must be related to an individual.
 * It makes no sense to have a health care service with no relation to an individual
 */
fact allIndividualMustHaveAHealthcareServiceAndCannotBeHealthcareServiceWithoutIndividual {
  (all i: Individual | i.healthcareService ≠ none) and
  (all h: HealthcareService | some i: Individual | i.healthcareService = h)
}

```

```

/* All notifications must have a relation with a health care service and an individual.
 * Furthermore, the individual related to the notification must be related to the same
 * health care service the notification has.
 */
fact allNotificationsMustHaveAnAssociatedHealthcareServiceAndIndividual {
  (all n: HSNotification | some hs: HealthcareService | n.healthService = hs) and
  (all n: HSNotification | some i: Individual | n.individual = i and
    i.healthcareService = n.healthService and
    i.healthcareService = n.individual.healthcareService)
}

// There must not be 2 notifications with the same healthcare service
fact noCommonHSNotificationWithSameHealthService {
  no disj n1,n2: HSNotification |
    n1.healthService = n2.healthService and
    n1.individual = n2.individual
}

/* If any parameter in the data of the individual is less than 0, it should have a
 * notification related to it and its health care service.
 */
fact allIndividualsWhoseDataIsLessThanZeroMustHaveAHSNotification {
  (all i: Individual | some p: i.data.parametersList |
    p < 0 implies i.healthcareService.notification ≠ none
    and i.healthcareService.notification.individual = i
    and i.healthcareService.notification.healthService = i.healthcareService) and
  (all n: HSNotification, i: n.individual | some p: i.data.parametersList | p < 0)
}

// compare against defined thresholds
pred checkThresholds[d: Data]{
  // any parameter on an individual's data is below its threshold
  some p: d.parametersList | p < 0
}

pred show(){
  #Individual > 2
  #HSNotification > 1
}

/* Having an individual with a parameter less than zero, implies that there is a
 * notification related to it and its health care service
 */
assert NotificationToHealthcareServiceSent{
  all i: Individual | checkThresholds[i.data] implies
    (all h: i.healthcareService | some n: i.healthcareService.notification |
      n.healthService = h and n.individual = i)
}

run show

check NotificationToHealthcareServiceSent

```



### 4.2.3 Generated worlds

After running the model described above, different worlds were generated, two of them are:

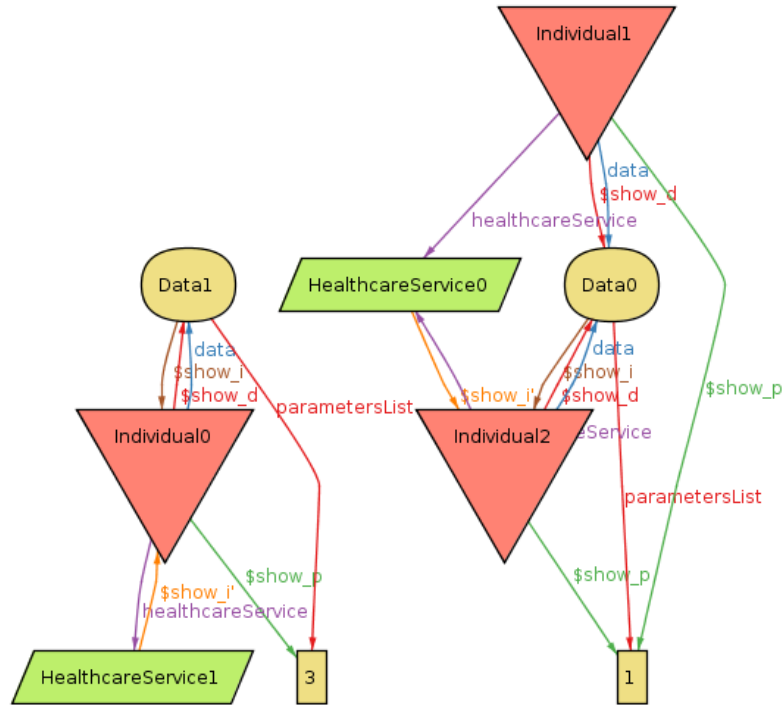


Figure 4.3: ASOS - 1<sup>st</sup> generated world. It can be seen three Individual objects, two of them related to HealthcareService0 object, and the other one related to HealthcareService1 object. None of the Individual objects have a Data parameter below zero, therefore there are no HSNotification objects.

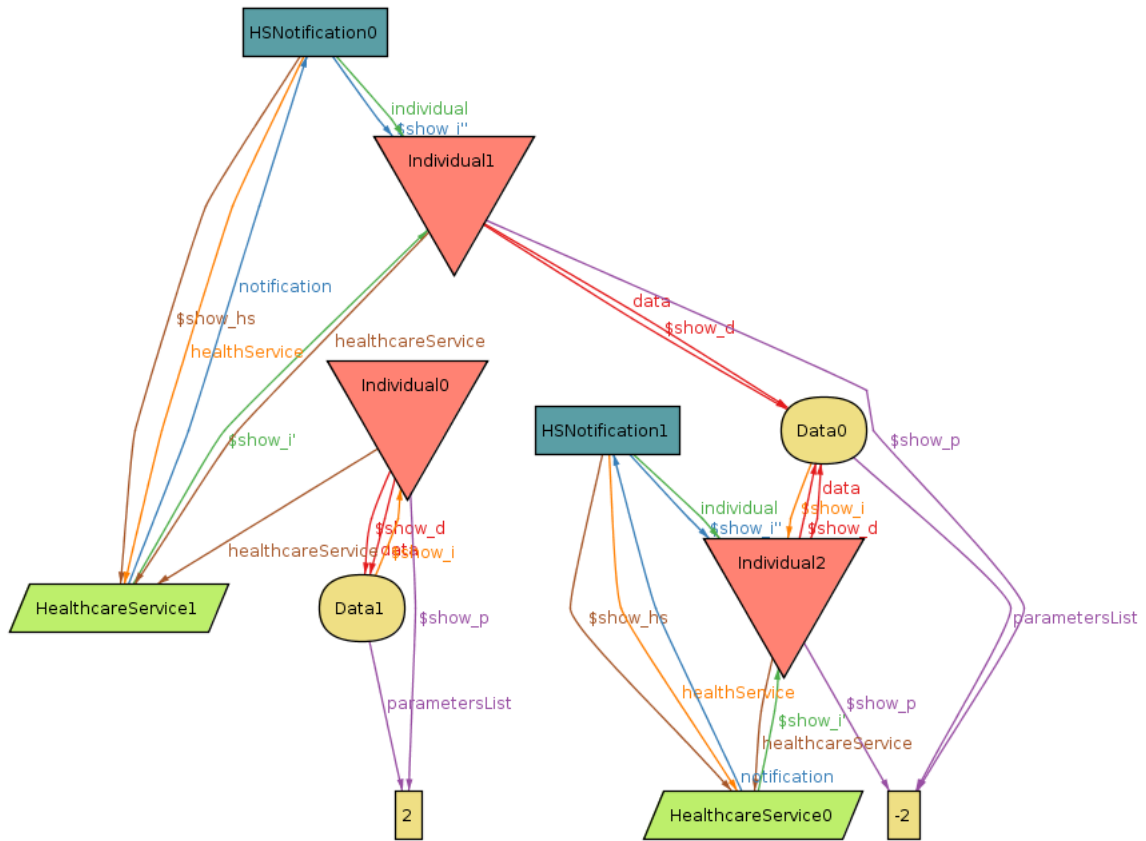


Figure 4.4: ASOS - 2<sup>nd</sup> generated world. It can be seen three Individual objects, two of them related to HealthcareService1 and the other one related to HealthcareService0 object. In this case the Individual1 and Individual2 objects have Data parameters below zero, hence there are HSNotification objects that relate them with the corresponding HealthcareService objects. The Individual0 object has Data parameters above zero, therefore there is no HSNotification object for it.

# Effort spent

Team Work	
Task	Hours
Understanding the problem	3
Brainstorming	2
World and shared phenomena	2
Definitions, acronyms, abbreviations	1
Software system attributes	2
Alloy coding	7
Checking document	4
<b>Total</b>	<b>20</b>

Table 5.1: Time spent by all team members

Individual Work					
Diego Avila		Laura Schiatti		Sukhpreet Kaur	
Task	Hours	Task	Hours	Task	Hours
Goals	4	Context	1	Layout	3
Product functions	2	Purpose	2	Problem description	2
F requirements	3	Domain model	4	User characteristics	1
UC diagrams	1	Statechart diagrams	4	User interfaces	4
UC description	7	Assumptions	2	Design constraints	1
Scenarios	1	Activity diagrams	3	External Interface	1
Sequence diagrams	2	Effort spent	2	NF requirements	1
Traceability matrix	3	Alloy section layout	1		
SS attributes	2				
<b>Total</b>	<b>25</b>	<b>Total</b>	<b>19</b>	<b>Total</b>	<b>13</b>

Table 5.2: Time spent by each team member

# References

---

- Requirement Analysis and Specification Document: AA 2017-2018.pdf”. Version 1.0 - 26.10.2017
- Henriksen, A., Haugen Mikalsen, M., Woldaregay, A. Z., Muzny, M., Hartvigsen, G., Hopstock, L. A., Grimsgaard, S. (2018)  
Using Fitness Trackers and Smartwatches to Measure Physical Activity in Research: Analysis of Consumer Wrist-Worn Wearables. Journal of medical Internet research, 20(3), e110. doi:10.2196/jmir.9157.  
Retrieved from: <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5887043/>
- IEEE. (1993). IEEE Recommended Practice for Software Requirements Specifications (IEEE 830-1993).  
Retrieved from <https://standards.ieee.org/standard/830-1993.html>
- Sloane, A. M. (2009). Software Abstractions: Logic, Language, and Analysis by Jackson Daniel, The MIT Press, 2006, 366pp, ISBN 978-0262101141.