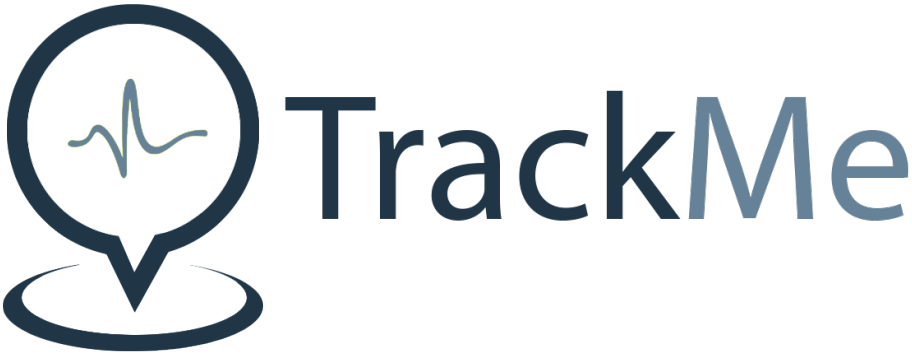
POLITECNICO
MILANO 1863

M.Sc. Computer Science and Engineering
Software Engineering 2 Project



Requirements Analysis and Specification Document

Gargano Jacopo Pio, Giannetti Cristian, Haag Federico

15 December 2018

GitHub Repository: https://github.com/federicohaag/GarganoGiannettiHaag

Version 1.1

Contents

Chapter 1

Introduction

1.1 Purpose

TrackMe wants to offer a service named "Data4Help" on top of which two services, named "AutomatedSOS" and "Track4Run", will be built.

Data4Help: the basic idea behind Data4Help is to acquire the location and health data of *Users* through a *Smart wearable* connected to their smartphone. Moreover, data can be directly sent to *Third parties*. In order to obtain and analyze *User data*, these need to obtain *User* authorization. Furthermore, they can request anonymized data of a group of *Users*.

AutomatedSOS: a service offered only to subscribed customers that constantly monitors their health status. Its purpose is to identify when a *User* is in need of immediate assistance and send an ambulance to their location.

Track4Run: a service used to track runners participating in running competitions. *Organizers* will be able to define a path for the run, *Participants* will share their position and health data and *Spectators* may watch the competition on their smart devices.

In this document, the goals of the system to be are analyzed through the identification of the needed requirements and domain assumptions.

1.2 Scope

TrackMe offers its services in a world where technology and health are taking huge strides forward every day and innovation is commonplace.
Nowadays, people use smart devices such as smartphones and *Smart wearables* more than any other object that they own. This means that any activity they

perform can be integrated with these devices.

TrackMe, with the introduction of Data4Help, offers the possibility to monitor *Users*' location and health data and allows *Third parties* to acquire these data.

When it comes to personal data acquisition, privacy is a fundamental issue that TrackMe needs to consider. Privacy is, in fact, regulated by several laws: there are many restrictions on how *User data* is acquired and stored. Therefore, TrackMe is concerned with *Users*' consent to transferring data to TrackMe itself and to third parties for individual specific analysis. Moreover, TrackMe guarantees that data of groups of individuals (*Group data*) are properly anonymized by checking specific constraints.

Over the course of their daily routine, *Users* perform several actions during which their data can be analyzed to provide them with insights. For instance, they might want to monitor their heart rate while sleeping or to keep track of the distance they have walked during their day and the places they have been to.

People with a potential need for immediate assistance have always been a huge concern for their relatives and technology makers. These may include old people with limited movement and a high chance to need urgent assistance, anyone who has a specific disease, but also a healthy individual who may suffer from a sudden heart failure. Until now, the only practical way to receive help has been calling for help, either by using a cell phone or by pushing an SOS button on a dedicated device. TrackMe proposes to automatize the step of calling for help through AutomatedSOS. In fact, when determined health values will no more be considered as normal, the system will automatically send a request for help.

Furthermore, nowadays, when it comes to sports and working out, having the possibility of collecting and sharing athletes' data is a disruptive innovation. In fact, giving anybody the possibility of having on their smartphone an accurate analysis of their health while performing a work out session is a breakthrough. A sport that is practiced and loved by many is running. Organizing a run requires several steps to be taken such as defining a path, getting athletes to participate and spectators to watch it. TrackMe proposes to simplify the organization of a run, by introducing Track4Run. This service will allow the definition of a path, easy enrollment for participants and a real-time tracking of each runner's position on a map.

1.2.1 Analysis of Shared Phenomena

1. *Users* register to Data4Help.
2. *Users* move.
3. *Users* can have health problems.

4. *Smart wearable* sensors acquire data.
5. *Smart wearables* communicate with Data4Help through smartphones.
6. *Third parties* register to Data4Help.
7. *Third parties* collect data from Data4Help.
8. *Users* grant direct usage and sharing of their personal data.
9. *Users* add a new *Service*.
10. *Organizers* define a path for a *Run*.
11. *Participants* enroll in a *Run*.
12. *Spectators* of a *Run* see on a map the position of the runners.

1.3 Goals

Data4Help

G_1 Collect *User data* through *Smart Wearables*.

G_2 Send specific *User data* to *Third parties* only if *User* consent was given after *Third party* access request.

G_3 Send anonimyzed requested *Group Data* to *Third parties* if the group it refers to is made up of 1000 or more *Users*.

G_4 Send *User data* and *Group Data* to subscribed authorized *Third parties* as soon as they are produced.

G_5 Allow *Users* to manage their subscription to *Services* and to Data4Help.

AutomatedSOS

G_6 Analyze *User data* to check whether or not a *User* is a *User in need*.

G_7 Send an ambulance to the last position of a *User in need*.

Track4Run

G_8 Allow *Organizers* to create a *Run*.

G_9 Allow *Users* to enroll in a *Run* as *Participants*.

G_{10} Allow *Spectators* to watch a *Run*.

1.4 Definitions, Acronyms, Abbreviations

1.4.1 Definitions

User: registered individual of Data4Help who agreed to the acquisition and processing of their data (see *User data*).

User data: *User*'s health data and location acquired by Data4Help.

Third party: a company that is willing to access *User data* stored in Data4Help database. It may offer *Services* to *Users*.

Service: application available for Data4Help *Users*, generally offered by a *Third party*.

Group data: set of *User data* acquired by Data4Help. The set of *Users* is determined by specific characteristics and constraints defined by the *Third party* requesting the data. When sent to the *Third party*, this data is anonymized.

Smart wearable: smart devices that can be worn on the body as accessories. These devices are required to have specific sensors for data acquisition. They must be connected to an external device, such as a smartphone.

Anomalous data: health data that is outside certain intervals identifying a *User* normal health condition.

User in need: registered user of AutomatedSOS in need of assistance since their health data is *anomalous*.

Run: running competition registered on Track4Run.

Organizer: person organizing a *Run*.

Spectator: person participating as spectator of a *Run*.

Participant: *User* who added Track4Run to their services, participating in a *Run*.

Username: *User*'s email address.

Terms and conditions: a set of regulations which *Users* must agree to follow in order to use Data4Help and the *Services* built on top of it.

Privacy statement: describes why and how TrackMe collects and uses personal data and provides information about *Users'* rights.

1.4.2 Acronyms

GPS: Global Positioning Service

GDPR: General Data Protection Regulation

1.4.3 Abbreviations

G_n: n^{th} goal

D_n: n^{th} domain assumption

R_n: n^{th} requirement

S_n: n^{th} scenario

U_n: n^{th} use case

PR_n: n^{th} performance requirement

1.5 Revision History

1. Version 1.0 - 11^{th} November 2018 - First Release
2. Version 1.1 - 15^{th} December 2018 - Second Release

   Major Changes

   - Added brief document purpose (sections 1.1).
   - Definitions revision (section 1.4.1).
   - Removed class Data Category from Data4Help Class Diagram (section 2.1.1).
   - Added class Spectator to Track4Run Class Diagram (section 2.1.3).
   - S_3 revision (section 3.2.1).
   - Add New Service Sequence Diagram revision (section 3.2.4).
   - Requirements revision (section 3.2.5).
   - Reliability further explained (section 3.5.1).
   - Minor fixes

1.6 Reference Documents

- Rumbaugh, Jacobson, Booch. 1999. *The Unified Modeling Language Reference Manual*. Addison-Wesley.
- MIT Software Design Group. *Appendix B: Alloy Language Reference*. http://alloytools.org/documentation.html
- MIT Software Design Group. *Tutorial Materials, Slides*. http://alloytools.org/tutorials/day-course/

1.7 Document Structure

This document is divided into six chapters.

Chapter 1 It introduces the project in terms of purpose, scope and goals. Moreover, it contains the definitions, acronyms and abbreviations needed to properly understand the sections following. All the documents used during the development of this project are listed.

Chapter 2 This chapter goes deep into the system description and definition. In particular it has:

- A detailed description of the product to be delivered and its features.
- A list of all domain assumptions that enable to reach the goals described in section 1.3.
- A description of users for which the product was thought.

Chapter 3 It analyzes and lists all the requirements needed to reach the goals described in section 1.3. Requirements are a huge and fundamental part of the project life cycle. Therefore, the chapter is divided in several sections, one for each aspect of requirements definition:

- Functional Requirements are listed as a description of interfaces, scenarios and uses cases (use case and sequence diagrams are included).
- Non Functional Requirements: performance, reliability, availability, security, portability.
- Any design constraint.

Chapter 4 This chapter is dedicated to the formalization of the system and its scope through a formal representation of entities and constraints using Alloy. The representation is split between Data4Help, AutomatedSOS and Track4Run to keep the generated world as simple as possible and to keep the three systems separated even if they have some interaction. In fact, AutomatedSOS and Track4Run do not have interactions between each other, but they do with Data4Help. Therefore Data4Help appears in the world dedicated to the two *Services*.

Chapter 5 Effort spent by all team members is shown as the list of all activities done during the realization of this document.

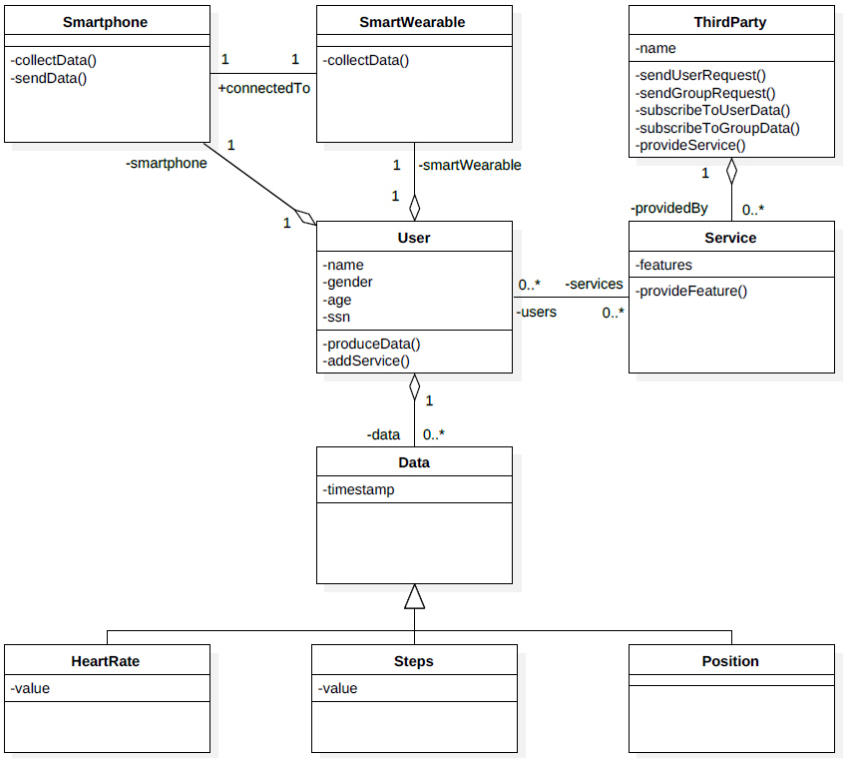
Chapter 6 References of documents that this project was developed upon.

Chapter 2

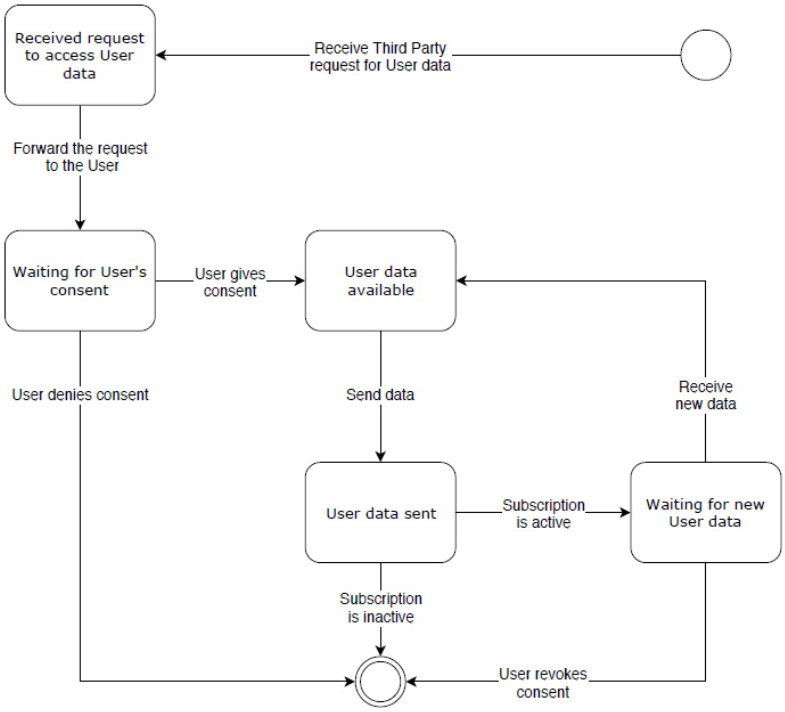
Overall Description

2.1 Product perspective

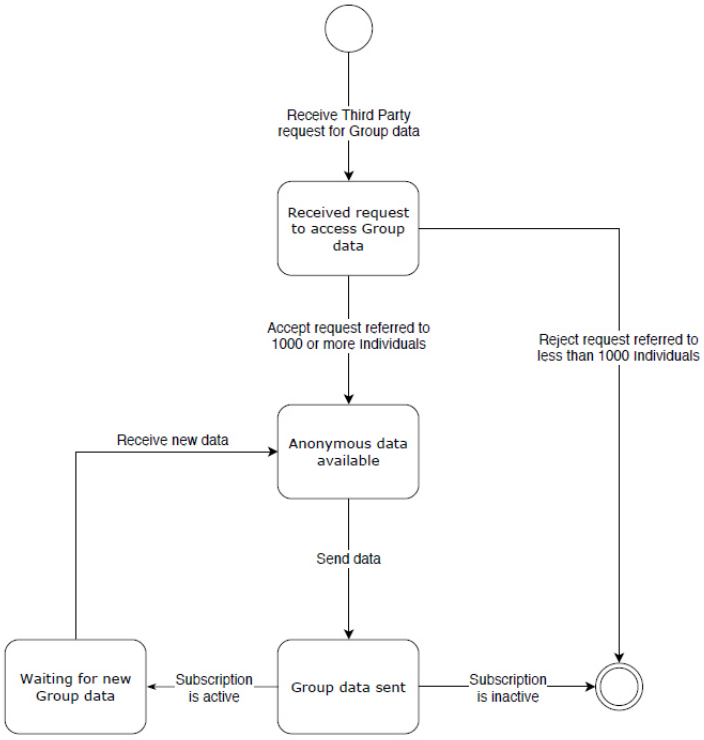
2.1.1 Data4Help



Data4Help class diagram



Data4Help state chart referred to *User data* request

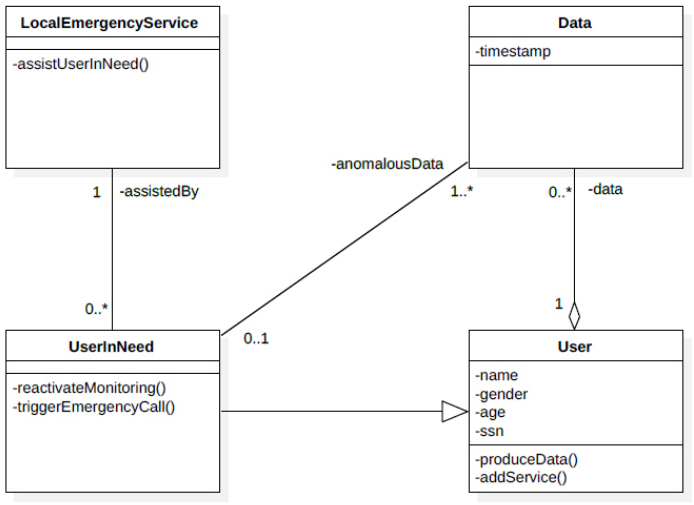


Data4Help state chart referred to *Group data* request

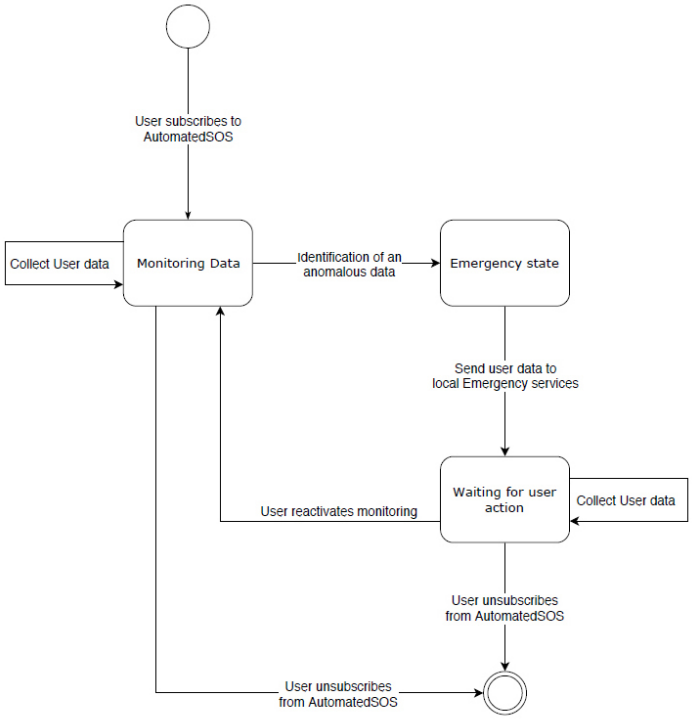
Shared Phenomena

- *User* subscribes to Data4Help.
- *User* logs in to Data4Help.
- *User* adds a new *Service* on personal Data4Help account granting to it the direct access of data.
- *User* unsubscribes from Data4Help.
- *Smart wearables* send data to Data4Help through smartphones.

2.1.2 AutomatedSOS



AutomatedSOS class diagram



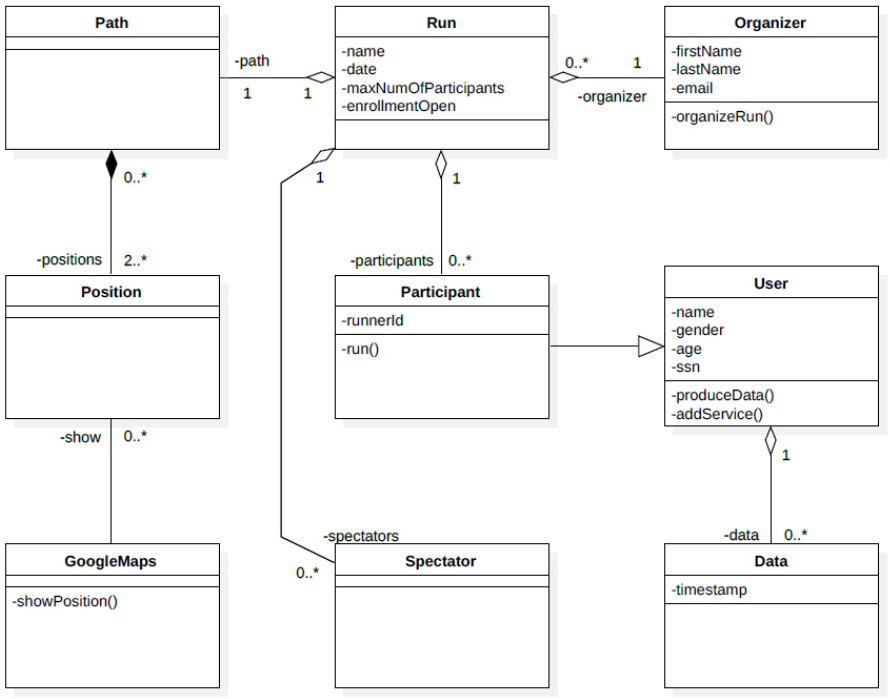
AutomatedSOS state chart

The product, as it is built on top of Data4Help, inherits from it the same shared phenomena.

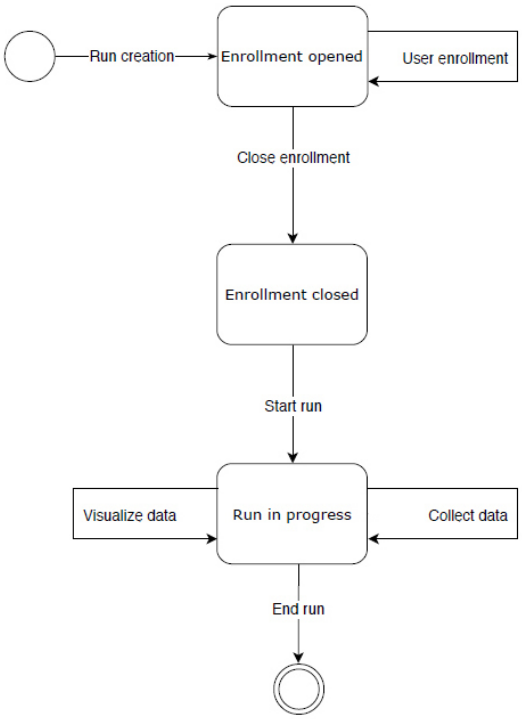
Shared Phenomena

- *User* becomes a *User in need*.
- AutomatedSOS calls local emergency services.

2.1.3 Track4Run



Track4Run class diagram



Track4Run state chart

The product, because it is built on top of Data4Help, inherits from it the same shared phenomena.

Shared Phenomena

- *Organizer* creates a running competition.
- *User* enrolls in a running competition.
- *Spectators* watches the *Participants*' tracking map.

2.2 Product functions

2.2.1 Data4Help

User Registration

Data4Help will allow individuals to register. These will register by entering all the required information (see R_2). When registering to Data4Help, an individual will first declare to have read the *Privacy statement* and secondly they

will have to accept the *Terms and conditions*, which specifically include their consent to the acquisition and processing of their data, including sensitive ones, by TrackMe.
The *User* registration process will be carried out on the Data4Help application (see section 3.1.1).

Third Party Registration

A *Third party* may register to Data4Help including all required information (see R_3). Once *Terms and conditions* have been accepted by the *Third party*, it will be successfully registered to Data4Help.

User Data Acquisition

Data4Help will acquire *User data* through *Smart wearables*. *Users* must give consent to the acquisition of their data when registering to Data4Help.

Third Party Data Request

Once a *Third party* is registered to Data4Help, it can request access to *User data* acquired through Data4Help and stored by TrackMe. *Third parties* may request data that refers either to a specific individual - *User data* - or to a group of *Users* identified by common characteristics - *Group data*.
Consent to individual data access is left to the specific *User*, who can either give or deny it to a *Third party* request.
Group data will be shared with *Third parties* as long as TrackMe will be able to anonymize it properly (see R_{23}).

Data Management and Privacy

All data acquired through Data4Help will be stored on a database accessible only by TrackMe. At any time, a *User* will be able to revoke the previously given consent to any *Third party* or to TrackMe. Moreover, a *User* may exercise their right to data portability, which means that TrackMe will have to provide them with all the collected data regarding them (see R_{32} and R_{34}). Finally, *Users* may ask the deletion of all their data stored by TrackMe (see R_{33} and R_{35}).
By guaranteeing these functions, Data4Help will respect existing general regulations on privacy (e.g. EU GDPR).

2.2.2 AutomatedSOS

User Subscription

All Data4Help *Users* can add AutomatedSOS to their *Services* through Data4Help application (see section 3.1.1).

Health Status Monitoring

The *Service* will constantly monitor *User*'s health data to verify if it is *Anomalous*.

Calling an Ambulance

In case the health status of a subscribed *User* is considered not to be good, AutomatedSOS will make a call to local emergency services within 5 seconds asking to send an ambulance to the last registered location of the *User*.

2.2.3 Track4Run

User Registration

Track4Run will be a *Service* used by three different kinds of users: *Organizers*, *Participants* and *Spectators*. *Organizers* will register to Track4Run by filling in all required information in the organizers registration form (see R_{41}). *Participants* will enroll in the *Run* through the Data4Help application. *Spectators* may watch a *Run* looking for it through its name or identifier.

Run Creation and Path Definition

Organizers can create a *Run*. They will be able to give the *Run* a name, set a date and time the *Run* is going to be held on and define a path for it. Moreover, they may limit the number of *Participants* or decide when to close enrollment (see R_{43} and section 3.1.1).

Display Runners on Map

Track4Run will display a map with the real time position of all the *Participants* during a *Run* (see section 3.1.1). *Spectators* may watch a *Run* by inserting its name or identifier. Real time statistics of *Participants* will be shown (e.g. heart rate, standings).

2.3 User characteristics

2.3.1 Data4Help

Users: People having at least a device with a sensor connected to the Internet, willing to share their data (see *User data* in section 1.4.1) with TrackMe so as to use the *Services* built on top of Data4Help.

Third parties: Companies or private persons willing to collect bulk data. This data is mainly used for building *Services* on top of Data4Help; for many of these *Services* it is very important that data is transferred in real time.

Otherwise data may be used for statistical analysis. In both cases, *Third parties* need that collected data is correct and accurate.

2.3.2 AutomatedSOS

Users: People with a high probability of needing immediate assistance. AutomatedSOS users are willing to monitor their health parameters and GPS location to prevent finding themselves alone when in need. These are mainly elderly people, especially those living by themselves. However, all categories of people may want to use AutomatedSOS, specifically those suffering from a disease that may strike any moment.

2.3.3 Track4Run

Organizers: Companies or private persons organizing *Runs* willing to better engage *Spectators* giving them the possibility to track in real-time the position of all *Participants*. They need to provide this *Service* easily in order to ensure *Spectators* and *Participants* are not prevented from using it.

Participants: People participating in a *Run*. They need to have a small device with no required interaction during the *Run* so as to avoid distractions.

Spectators: People participating as spectators of *Runs*. They are willing to enjoy the event by tracking *Participants* during all the *Run*. Watching a *Run* must be easy: no need of particular devices or installed applications.

2.4 Assumptions, Dependencies and Constraints

2.4.1 Domain Assumptions

D_1 Personal data inserted by the *User* at sign up corresponds to their real data.

D_2 *User data* collected at a certain instant corresponds to the actual status (GPS position and health data) of the *User* at that precise moment.

D_3 The maps in use accurately represent the world.

D_4 A *Third party* can receive consent to *User data* access only through a *Service* it offers and can use the data only for that specific *Service*.

D_5 AutomatedSOS and Track4Run are *Services* developed by TrackMe.

D_6 AutomatedSOS and Track4Run are subscribed to new *User data* of all their *Users*.

D_7 When AutomatedSOS needs to send an ambulance to a *User in need* it forwards the request to local emergency services, which eventually dispatch an ambulance.

D_8 *Smart Wearables* are correctly worn by *Users*.

D_9 Data4Help and all *Services*, including AutomatedSOS and Track4Run, are always online.

D_{10} *Users* own a working smartphone which is always connected to the Internet.

D_{11} *Users* own a working *Smart wearable* which is always connected to the *User*'s smartphone.

D_{12} Either smartphones or *Smart wearable* have a working and active GPS.

2.4.2 Dependencies

- Availability, performance and reliability of services, including those of AutomatedSOS and Track4Run, depend respectively on the availability, performance and reliability of Data4Help.
- Effectiveness of AutomatedSOS depends on local emergency services response time.

2.4.3 Constraints

Data4Help

- Smartphones must always be online.
- Smartphones or *Smart wearables* must have GPS activated.
- Smartphones must be able to communicate with *Smart wearables* (e.g. via Bluetooth).
- Smartphones must have enough free space for downloading and installing the Data4Help application.

AutomatedSOS

- *User* must be wearing *Smart wearable* at all times to allow for *Anomalous data* detection.
- AutomatedSOS must always be able to call local emergency services.

Track4Run

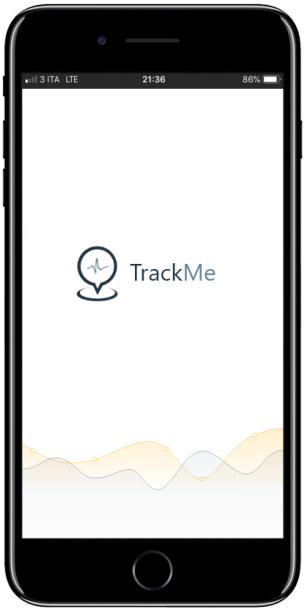
- *Organizers* and *Spectators* must have a modern browser installed on a device connected to the Internet in order to access respectively the *Organizers* administrator panel and the *Spectators* dedicated website.
- Only *Users* may enroll in a *Run*: individuals not registered to Data4Help cannot enroll in a *Run*.

Chapter 3

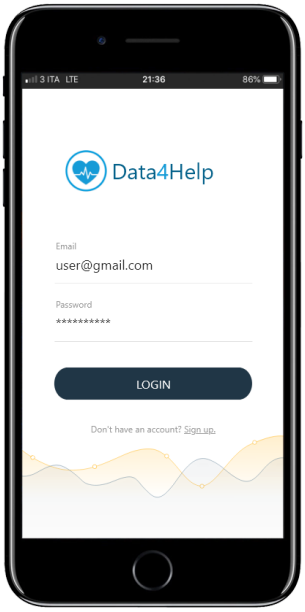
Specific Requirements

3.1 External Interface Requirements

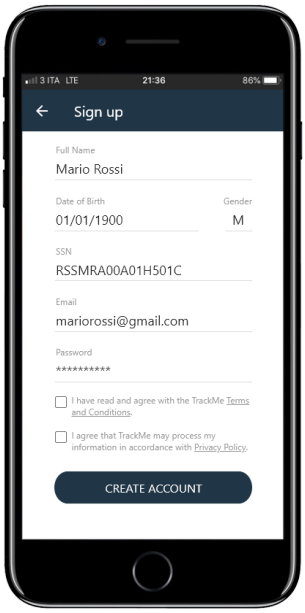
3.1.1 User Interfaces



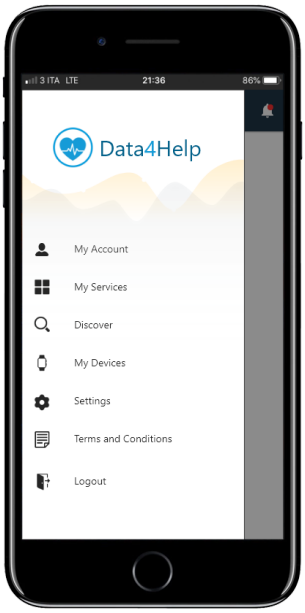
Data4Help Welcome Page



Data4Help Login Page

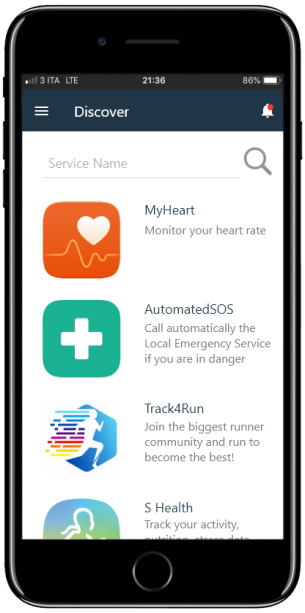


Data4Help Sign Up Page

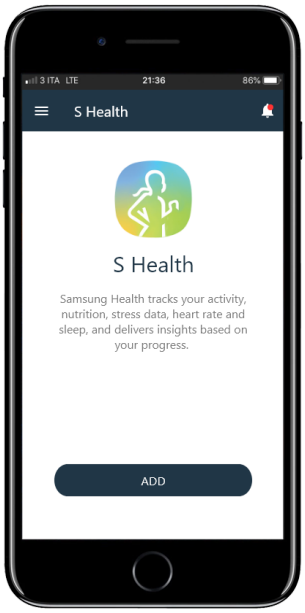


Data4Help Menu

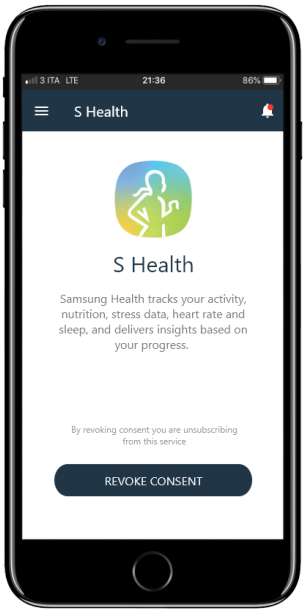
User Services Page



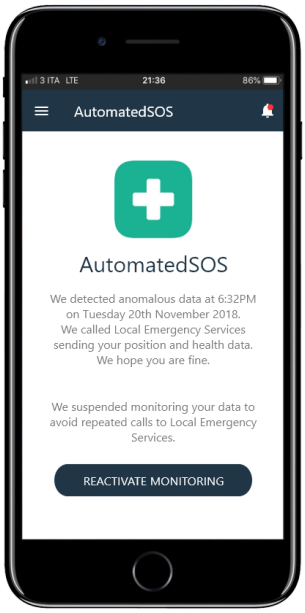
Services Discovery Page



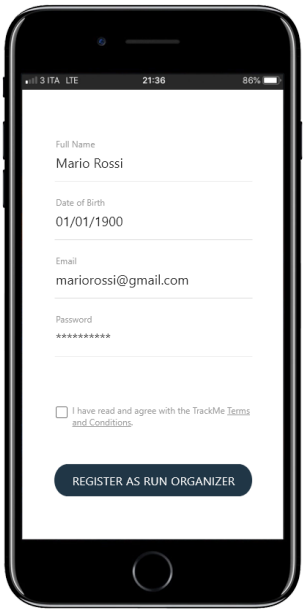
Add Service Page



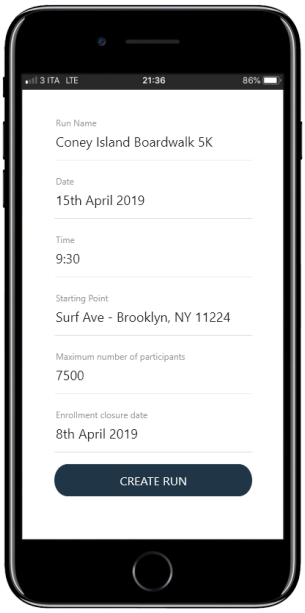
Subscribed User Service Page



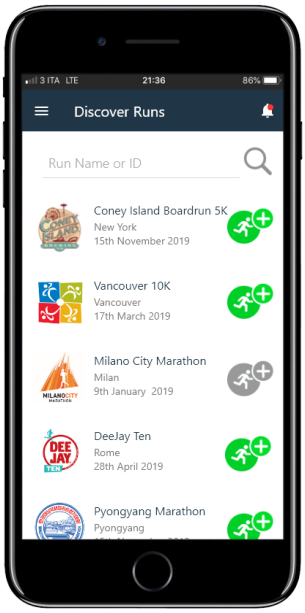
AutomatedSOS Page



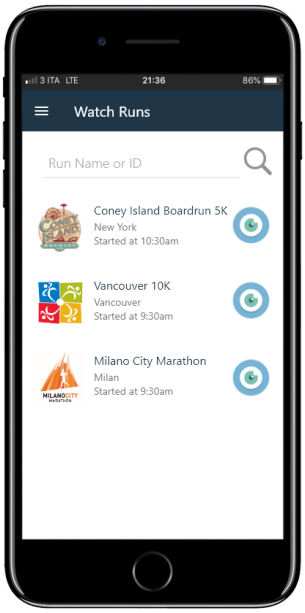
Organizer Registration Page



New Run Creation Page



Runs Discovery Page



Spectators Run Discovery Page

Watch Real Time Run Page

3.1.2 Hardware Interfaces

Data4Help is a service based on native applications for smartphones and on a centralized backend handling the retrieval of data from users and their storage. Data4Help acts only as an intermediary between the sources of data and the parties that want to get access to those. Due to this, Data4Help has no hardware interfaces.
Obviously also the services built on top of it, being pure software adds-on working on top of basic API's of Data4Help, don't have hardware interfaces.

3.1.3 Software Interfaces

Data4Help will expose an API to *Services* offered by *Third parties* to send *User data* and *Group data*. In particular, its main functions are to:

- Retrieve all *User data* available for a given *Username*.
- Retrieve last n records of *User data* for a given *Username*, where n is a specified quantity.
- Retrieve all *User data* available to form *Group data* according to the following parameters: age, sex, residence.
- Send a notification to a specific *User*.

3.1.4 Communication Interfaces

The system to be will only make use of the usual communication protocols (TCP/IP) to guarantee the connection between the *User's Smartphones* and the *Services*' backend systems.
Other protocols, as Bluetooth for example, may be used by *Smart wearables* to comunicate with *User's Smartphone* but this is out of the system to be scope.

3.2 Functional Requirements

3.2.1 Scenarios

Data4Help

S_1 Dante is an individual who would like to keep track of his GPS position and health data. For this purpose he decides to use Data4Help. He downloads the Data4Help application on his smartphone and proceeds to sign up. He inserts all required information, which include his name, his social security number and date of birth. He is asked to insert an email that will later be his *Username* and a password. Dante inserts his name as his password and the system tells him that the inserted password is shorter than 8 characters, so he tries again with a new one. Eventually he inserts a valid password, accepts the *Terms and conditions* and taps on "Create an Account". He is successfully signed up, after receiving a confirmation email by TrackMe. He tries to login to the application by inserting the newly created *Username* and password. The system accepts the credentials and Dante is in.

S_2 YourHealth is a company that analyzes individuals' health data to provide users with insights on their well-being. It decides to offer its service also on Data4Help so as to have a greater pool of users. The person in charge navigates to the *Third party* dedicated website and clicks on "Sign Up". They fill in all required information about their company, insert an email that will be used as *Username* and a valid password. They then accept the *Terms and conditions* by clicking the specific checkbox, and finally click on "Create an Account". YourHealth receives an email confirming the account creation: now YourHealth *Services* are available to all Data4Help *Users*.

S_3 Dante, a Data4Help *User*, needs to monitor his heart rate through the day. He navigates to the "Discover" page inside of his Data4Help application and scrolls through the available *Services*. He finds MyHeart, a *Service* developed by YourHealth, a *Third party* registered to Data4Help. The description of the service seems to suit his need, so he decides to add MyHeart to his *Services*. In order to finalize the subscription, Dante will have to accept that his data will be sent to YourHealth for analysis. He does so by checking the consent checkbox and tapping on the "Add" button. After a while, in the specific MyHeart *Service* page, the "Analyze" button appears. Dante taps on it and promptly he sees a personalized graph showing his heart rate levels throughout the day, starting from the first day he registered to Data4Help.

S_4 LocalStats is a company that performs intensive statistics on individuals' positions in some cities of Switzerland. It decides to acquire individuals' GPS positions data from Data4Help to enlarge its database. LocalStats registers as a Data4Help *Third party*. Once registration is complete, the

first request it makes to Data4Help refers to all female *Users* between 30 and 35 years old living in Lausanne. Unfortunately, the number of *Users* with the requested characteristics is less than 1000, which does not guarantee proper data anonymization. Therefore, Data4Help rejects the *Group Data* request. LocalStats tries again changing the interval of interest to 25-35 years old. This time the request refers to more than 1000 *Users* and finally Data4Help can send the requested *Group Data* to LocalStats.

S_5 Dante, from scenario S_3, would like to keep MyHeart active day by day. To do so, he taps on "Analyze Daily", which is a function offered by MyHeart. YourHealth, which developed and manages MyHeart, requests subscription to Dante's new data. Data4Help registers that anytime Dante's *User data* is collected, it needs to send it to YourHealth for analysis. Starting from the following day, Dante does not need anymore to tap on "Analyze" every day: new analysis is provided to him as soon as it is available from MyHeart.

S_6 Dante, who subscribed to Data4Help and used its *Third party Services* for a while, decides that he does not want to use one of them, TrackKer, anymore. Therefore, he navigates to the "My Services" page and taps on TrackKer. The *Service* page shows up and he taps on the "Revoke Consent" button at the bottom of the page. From now on, Data4Help will stop sending Dante's data to the *Third party* managing TrackKer.

AutomatedSOS

S_7 GianVito is a 57 years old man who added AutomatedSOS to his *Services*. After getting very angry at work, he drives home, but as soon as he gets there he feels dizzy and falls on the ground. He is alone and cannot call for help. Fortunately, AutomatedSOS notices that his heart rate is below a certain threshold and identifies him as *User in need*. AutomatedSOS calls local emergency services and sends them GianVito's position and health data. When local emergency services dispatch an ambulance and GianVito is being taken care of, AutomatedSOS waits for GianVito's action, still collecting his data, without possibly identifying it again as *Anomalous*. Finally, GianVito opens up Data4Help application and taps on "Reactivate Monitoring". AutomatedSOS has fulfilled his need for immediate assistance and starts monitoring his health data again.

Track4Run

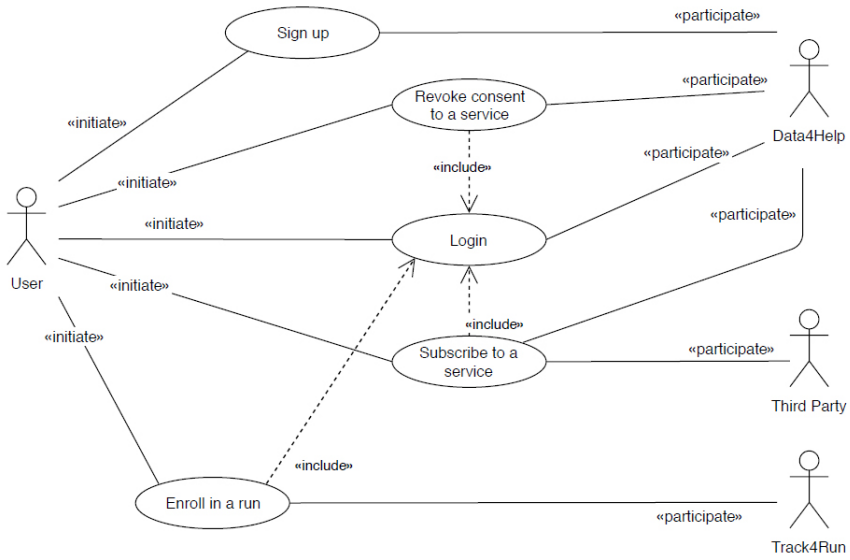
S_8 Charity4All is a Swedish charity association that organizes a running competition every year to raise money for their causes. The person in charge decides to use Track4Run to manage the run. They navigate to the *Run* dedicated website and sign up as an *Organizer*, inserting an email and a password for registration. Once sign up is complete, they click on "Create

Run" and the *Run* creation page shows up. They give the *Run* a name - Run4Char - they define a path around Gothenburg and set the date and time the competition will take place on. They do not want to limit the number of participants, so they click on "Create Run" and obtain a *Run* identifier back from Track4Run. They will distribute this identifier to all viewers who wish to enjoy the *Run* on their devices.

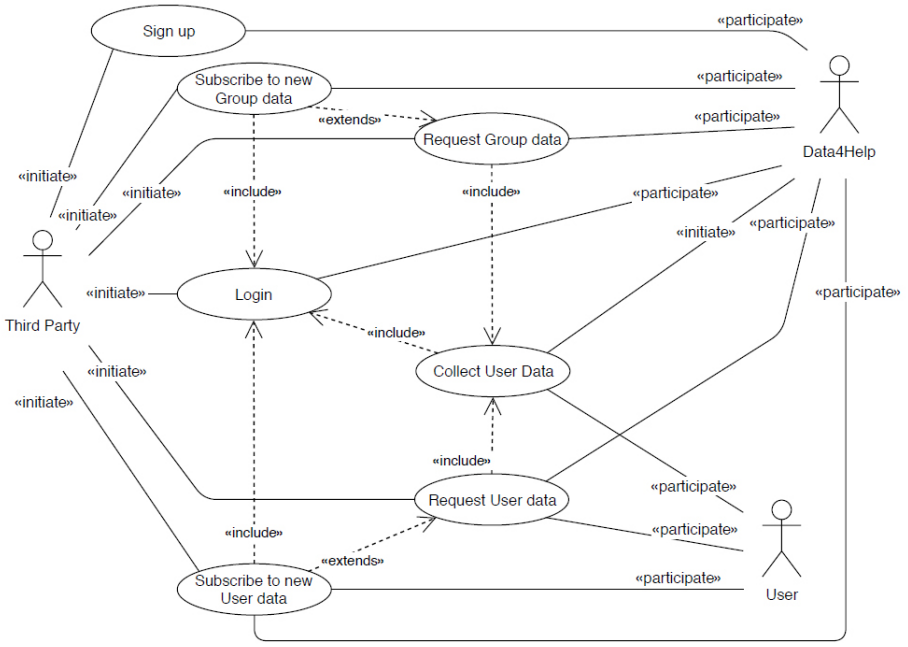
S_9 Hannah lives in Gothenburg and she loves running. In fact, she is subscribed to Track4Run. While browsing the available *Runs* in her city, she finds Run4Char - from S_8. She enrolls in the run right away and Track4Me records her registration. Hannah is now a *Participant* of the *Run*.

S_{10} George enjoys sports a lot, however he is very old now and cannot participate in competitions anymore. He still likes watching sports event, especially when it comes to running. Since he is also into helping others, he is subscribed to Charity4All - from S_8 - newsletter. He reads that they are organizing a *Run* and writes down the *Run* identifier. On the day of the *Run*, he navigates to the *Spectators* dedicated website and inserts the *Run* identifier. As soon as the *Run* starts, he enjoys it by watching the position of the *Participants* on the map right on his device, comfortably in his house.

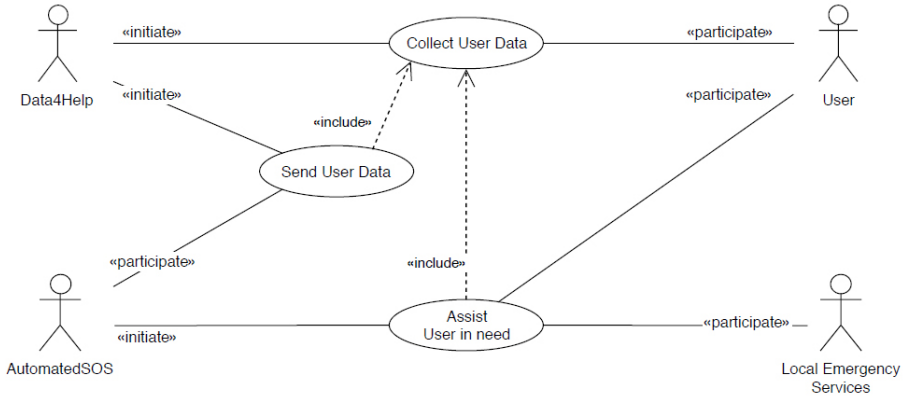
3.2.2 Use Case Diagrams



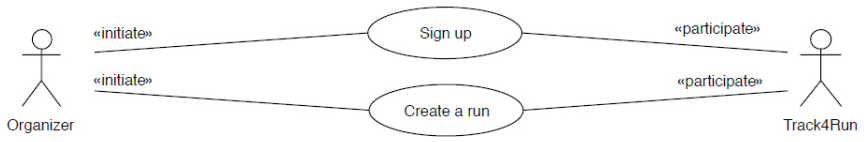
User Use Case Diagram



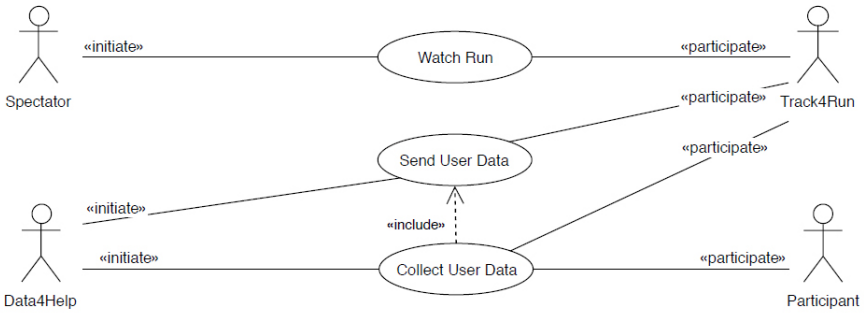
Third Party Use Case Diagram



AutomatedSOS Use Case Diagram



Track4Run Organizer Use Case Diagram



Track4Run Spectator Use Case Diagram

3.2.3 Use Case Analysis

Data4Help

Name	**User Sign Up**
Actors	*User*, Data4Help
Entry Conditions	*User* successfully installed Data4Help application on their smartphone.
Events Flow	1. *User* taps on "Sign Up" button. 2. *User* fills in all required fields for *User* registration, including *Username* and password. 3. *User* checks the "Accept terms and conditions" checkbox. 4. *User* taps on "Create an Account" button. 5. Data4Help saves *User* information.
Exit Condition	*User* successfully registered by Data4Help.
Exceptions	1. Inserted email already registered for another *User*. 2. Inserted password is not valid. 3. Not all required fields are filled in. 4. "Accept terms and conditions" checkbox not checked. 5. *User* already signed up. *User* is invited to try again signing up, reporting which error(s) they have committed.

U_1

Name	**Third Party Sign Up**
Actors	*Third party*, Data4Help
Entry Conditions	*Third party* is connected to the *Third party* dedicated website.
Events Flow	1. *Third party* clicks on "Sign Up" button. 2. *Third party* fills in all required fields for *Third party* registration, including *Username* and password. 3. *Third party* checks the "Accept terms and conditions" checkbox. 4. *Third party* clicks on "Create an Account" button. 5. Data4Help saves *Third party* information.
Exit Condition	*Third party* successfully registered by Data4Help.
Exceptions	1. Inserted email already registered for another *Third party*. 2. Inserted password is not valid. 3. Not all required fields are filled in. 4. "Accept terms and conditions" checkbox not checked. 5. *Third party* already signed up. *Third party* is invited to try again signing up, reporting which error(s) it has committed.

U_2

Name	**User Login**
Actors	*User*, Data4Help
Entry Conditions	*User* successfully registered to Data4Help and installed Data4Help application on their smartphone.
Events Flow	1. *User* enters *Username*. 2. *User* enters password. 3. *User* taps on "Login" button. 4. Data4Help checks *User* credentials.
Exit Condition	*User* is successfully logged in.
Exceptions	1. Inserted *Username* is not valid. 2. Inserted password is not correct. *User* is invited to try again logging in.

U_3

Name	**Third Party Login**
Actors	*Third party*, Data4Help
Entry Conditions	*Third party* successfully registered to Data4Help and is connected to the *Third party* dedicated website.
Events Flow	1. *Third party* enters *Username*. 2. *Third party* enters password. 3. *Third party* clicks on "Login" button. 4. Data4Help checks *Third party* credentials.
Exit Condition	*Third party* is successfully logged in.
Exceptions	1. Inserted *Username* is not valid. 2. Inserted password is not correct. *Third party* is invited to try again logging in .

U_4

Name	**Data4Help collects User data**
Actors	*User*, Data4Help
Entry Conditions	*User* successfully registered to Data4Help and installed Data4Help application on their smartphone.
Events Flow	1. *Smart wearable* sensors acquire *User data*. 2. *User data* is sent to Data4Help through the *User* smartphone via Internet. 3. Data4Help receives, validates and authenticates *User data*. 4. Data4Help identifies the *User* to whom data refers. 5. Data4Help stores the newly collected *User data* in a database.
Exit Condition	Data4Help correctly collected *User data*.
Exceptions	1. Data4Help does not validate the just received *User data*. 2. Data4Help does not authenticate the just received *User data*. 3. Data4Help cannot identify the *User* to whom data refers. Data4Help discards the message.

U_5

Name	**Third Party requests User data**
Actors	*Third party*, Data4Help, *User*
Entry Conditions	*Third party* and *User* successfully registered to Data4Help.
Events Flow	1. *Third party* requests access to specific *User data*. 2. Data4Help forwards the request to the specific *User* unless the consent was already given. 3. *User* gives consent to the requesting *Third party* to access their data.
Exit Condition	Data4Help sends *User data* to the *Third party*.
Exceptions	1. *User* denies consent to their data access by the requesting *Third party*.

U_6

Name	**Third Party requests Group data**
Actors	*Third party*, Data4Help
Entry Conditions	*Third party* successfully registered to Data4Help.
Events Flow	1. *Third party* requests access to *Group data*. 2. Data4Help checks if the requested data refers to minimum 1000 *Users*.
Exit Condition	Data4Help sends *Group data* to the *Third party*.
Exceptions	1. *Group data* refers to less than 1000 *Users*. Data4Help denies *Group data* access to the *Third party*.

U_7

Name	**Third Party subscribes to New User data**
Actors	*Third party*, *User*, Data4Help
Entry Conditions	*Third party* successfully registered to Data4Help and obtained access to *User data*.
Events Flow	1. *Third party* requests subscription to *User data*. 2. *User* gives consent.
Exit Condition	Data4Help registers the *Third party* subscription to new data. Each time new data is produced, it is sent to the *Third party*.
Exceptions	No Exceptions

U_8

Name	**Third Party subscribes to New Group data**
Actors	*Third party*, Data4Help
Entry Conditions	*Third party* successfully registered to Data4Help and obtained access to *Group data*.
Events Flow	1. *Third party* requests subscription to *Group data*.
Exit Condition	Data4Help registers the *Third party* subscription to new data. Each time new data is produced, it is sent to the *Third party*.
Exceptions	No Exceptions

U_9

Name	**User adds a Service**
Actors	*Third party*, *User*, Data4Help
Entry Conditions	*User* is successfully registered to Data4Help and installed Data4Help application on their smartphone.
Events Flow	1. *User* navigates to "Discover" page. 2. *User* chooses which *Service* they would like add. 3. *User* gives consent to sharing their data with the specific *Third party*. 4. *User* taps on "Add" button.
Exit Condition	Data4Help registers the new *Service* for the *User*.
Exceptions	1. *User* does not give consent to sharing their data. The *Service* is not added and the *User* is invited to try again adding it.

U_{10}

Name	**User revokes consent to a Service**
Actors	*User*, Data4Help
Entry Conditions	*User* gave consent to sharing their data with a *Third party*.
Events Flow	1. *User* navigates to "My Services" page. 2. *User* chooses which service they would like to revoke consent. 3. *User* navigates to the *Service* dedicated page by tapping on its name. 4. *User* taps on "Revoke consent" button.
Exit Condition	Data4Help stops sharing the data of the *User* with the specific *Third party*.
Exceptions	No Exceptions.

U_{11}

AutomatedSOS

Name	**User in need assisted by AutomatedSOS**
Actors	*User*, Data4Help, AutomatedSOS, Local emergency services
Entry Conditions	*User* is subscribed to AutomatedSOS and installed Data4Help application on their smartphone.
Events Flow	1. Data4Help sends *User data* to AutomatedSOS as soon as it is produced. 2. AutomatedSOS identifies User data as *anomalous data*. 3. *User* is identified as *User in need*. 4. AutomatedSOS calls local emergency services requesting an ambulance. 5. AutomatedSOS sends *User data* including GPS location and health data to local emergency services. 6. Local emergency services send an ambulance to the location of the *User in need*.
Exit Condition	*User in need* taps on "Reactivate Monitoring" button.
Exceptions	1. Local emergency services don't answer the call. 2. The call to local emergency services is answered but the communication fails before giving all the necessary details. AutomatedSOS repeats the call.

U_{12}

Track4Run

Name	**Organizer Sign Up**
Actors	*Organizer*, Track4Run
Entry Conditions	*Organizer* is connected to the Track4Run website dedicated to organizers.
Events Flow	1. *Organizer* taps on "Sign Up" button. 2. *Organizer* fills in all required fields for *Organizer* registration, including *Username* and password. 3. *Organizer* checks the "Accept terms and conditions" checkbox. 4. *Organizer* taps on "Create an Account" button. 5. Track4Run saves *Organizer* information.
Exit Condition	*Organizer* successfully registered by Track4Run.
Exceptions	1. Inserted email already registered for another *Organizer*. 2. Inserted password is not valid. 3. Not all required fields are filled in. 4. "Accept terms and conditions" checkbox not checked. 5. *Organizer* already signed up. *Organizer* is invited to try again signing up, reporting which error(s) they have committed.

U_{13}

Name	**Organizer creates a Run**
Actors	*Organizer*, Track4Run
Entry Conditions	*Organizer* is logged in and connected to the *Run* dedicated website.
Events Flow	1. *Organizer* clicks on "Create Run" button. 2. *Organizer* fills in all required fields for *Run* creation. 3. *Organizer* clicks on "Confirm" button.
Exit Condition	Track4Run creates the *Run* defined by the *Organizer*.
Exceptions	1. Not all required fields are filled in. *Organizer* is invited to try again creating the *Run*.

U_{14}

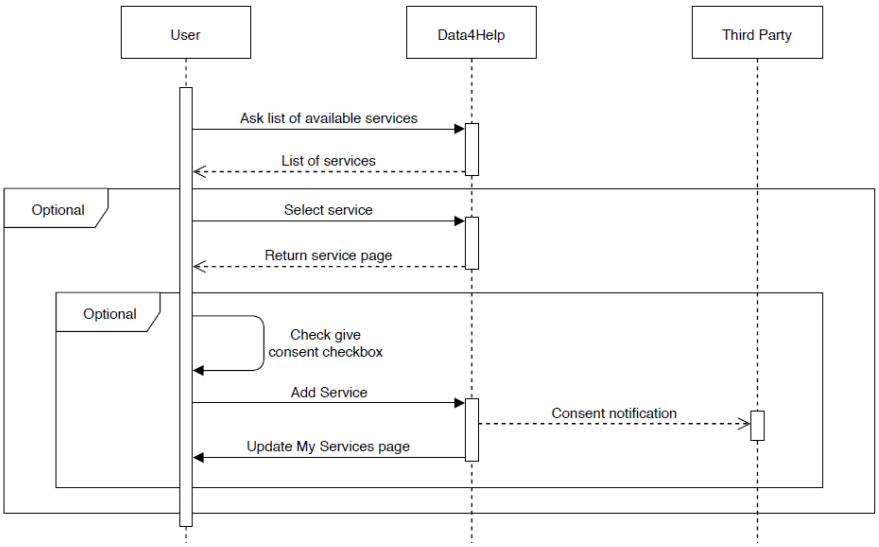
Name	**User enrolls in a Run**
Actors	*User*, Track4Run
Entry Conditions	*User* is subscribed to Track4Run and installed Data4Help application on their smartphone.
Events Flow	1. *User* navigates to "My Services" page. 2. *User* taps on Track4Run. 3. *User* taps on "Discover Runs". 4. *User* chooses the *Run* they wish to enroll in from the list or inserts the *Run* identifier or name in the search box. 5. *User* taps on "+" button to enroll in the *Run*.
Exit Condition	Track4Run registers the *User* as a *Participant* of the *Run*.
Exceptions	1. No *Runs* are listed. 2. The selected *Run* reached the maximum number of participants. 3. The selected *Run* is closed to enrollment. 4. There are no *Runs* associated to the inserted run identifier or name. *Participant* is invited to try again later enrolling in a *Run*.

U_{15}

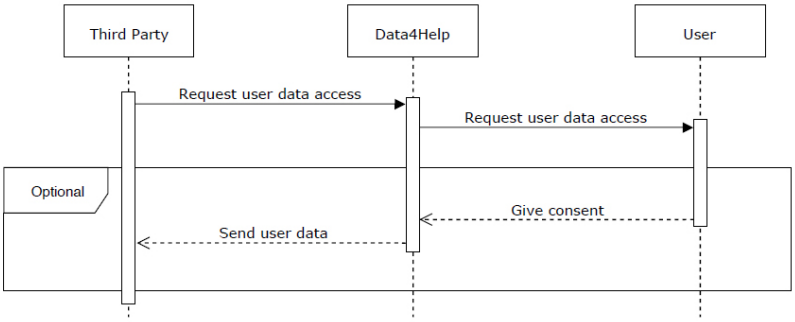
Name	**Spectator watches a Run**
Actors	Individual
Entry Conditions	Individual is connected to the *Spectators* dedicated website.
Events Flow	1. *Individual* clicks on the *Run* they would like to watch or inserts the run identifier or name. 2. *Individual* watches the *Run* as a *Spectator*.
Exit Condition	The *Run* is over.
Exceptions	1. No *Runs* are listed. 2. There is no *Run* associated to the inserted run identifier or name. 3. The *Spectator* disconnects from the *Spectators* dedicated website. Exception 1: *Spectator* is invited to try again later watching a *Run*. Exception 2: no action is taken.

U_{16}

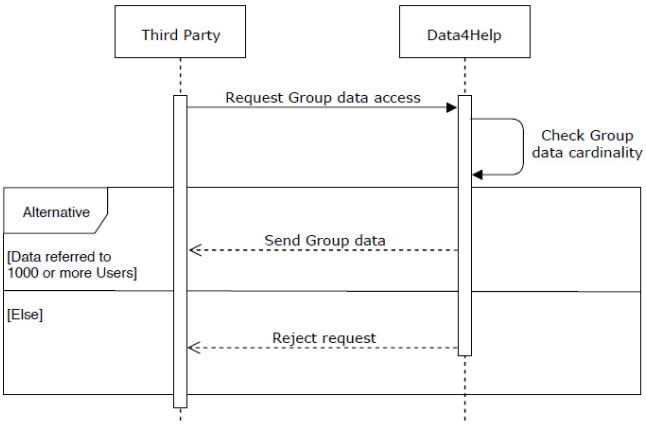
3.2.4 Sequence Diagrams



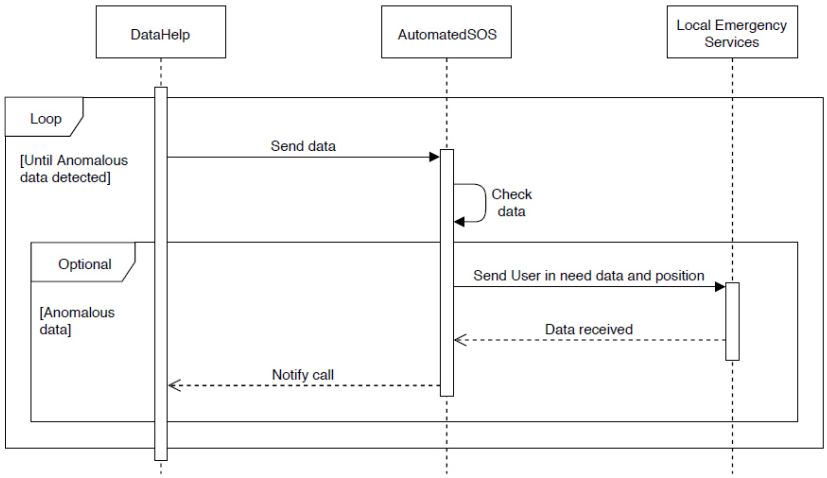
Add New Service Sequence Diagram



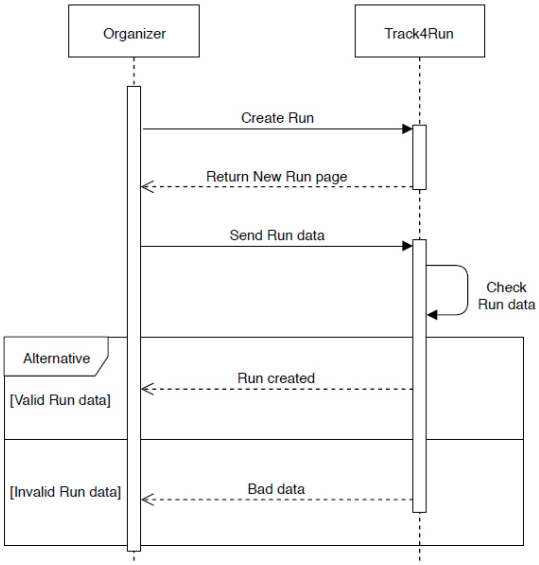
User Data Request Sequence Diagram



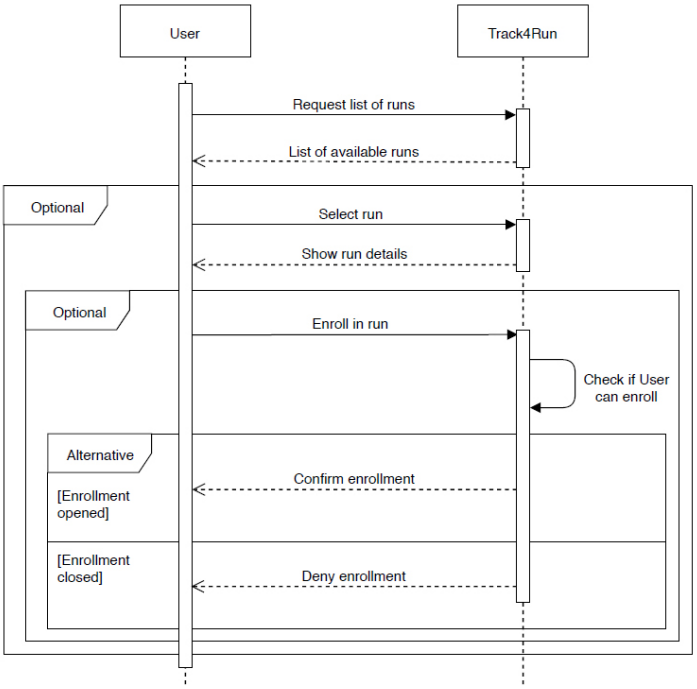
Group Data Request Sequence Diagram



User in Need Assistance Sequence Diagram



Run Organization Sequence Diagram



Participant Enrollment Sequence Diagram

3.2.5 Requirements

The following is a list of requirements for Data4Help, AutomatedSOS and Track4Run. Where not explicitly specified, the subject of the requirement is the system to which it refers to (Data4Help, AutomatedSOS or Track4Run).

Data4Help

- R_1 Unregistered individuals and companies must not be able to use Data4Help.
- R_2 At sign up, *User* must provide: first name, last name, SSN, gender, date of birth, email and password.
- R_3 At sign up, *Third party* must provide a company name.
- R_4 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*.
- R_5 At sign up, *Third party* must accept *Terms and conditions*.
- R_6 Identify a *User* by their identifier.
- R_7 Query Data4Help Database for a *User* by their identifier.
- R_8 Receive *User data*.
- R_9 Validate *User data*.
- R_{10} Authenticate *User data*.
- R_{11} Store collected *User data* in Data4Help Database.
- R_{12} Retrieve specific *User data* by querying Data4Help Database.
- R_{13} Receive *Third party* data access request.
- R_{14} Validate *Third party* data access request.
- R_{15} Authenticate *Third party* data access request.
- R_{16} Forward *User data* access request to the specific *User*.
- R_{17} Receive *User* consent approval or denial with respect to *Third party* data access requests.
- R_{18} Check if a specific *User* gave consent to a specific *Service*.
- R_{19} Send specific *User data* to the requesting *Third party*.
- R_{20} Not send specific *User data* to the requesting *Third party* if the specific *User* denied consent.
- R_{21} *Third party* must be able to set specific constraints to define a group of *Users*: age, gender, residence.

R_{22} Check how many *Users* the requested *Group data* refers to.

R_{23} Properly anonymize *Group data*.

R_{24} Send *Group data* to the requesting *Third party*.

R_{25} Not send *Group data* if the group it refers to is made up of less than 1000 *Users*.

R_{26} Receive *Third party* subscription request.

R_{27} Validate *Third party* subscription request.

R_{28} Authenticate *Third party* subscription request.

R_{29} Automatically send new *User data* to subscribed authorized *Third parties* as soon as they are produced.

R_{30} Allow *Users* to add a *Service*.

R_{31} Allow *Users* to unsubscribe from a *Service*.

R_{32} Send to a specific *User* all their data stored by TrackMe.

R_{33} Delete all data of a specific *User*.

R_{34} Allow *Users* to request all their data stored by TrackMe at any time.

R_{35} Allow *Users* to request the deletion of all their data stored by TrackMe at any time.

AutomatedSOS

R_{36} Receive *User data* from Data4Help.

R_{37} Compare *User data* against certain thresholds.

R_{38} Call local emergency services providing necessary *User data* of *User in need*.

R_{39} *User* must be able to reactivate AutomatedSOS monitoring.

Track4Run

R_{40} Receive *User data* from Data4Help.

R_{41} At sign up, *Organizers* must provide: first name, last name, email and password.

R_{42} At sign up, *Organizers* must accept *Terms and conditions*.

R_{43} Allow *Organizers* to create a *Run*, defining: name, path, date, maximum number of *Participants* and enrollment closure date.

R_{44} Allow *Users* to enroll in an existing *Run*.

R_{45} Prevent a *User* from enrolling in a *Run* if the maximum number of *Participants* was already reached.

R_{46} Prevent a *User* from enrolling in a *Run* if it already started or finished.

R_{47} Prevent a *User* from enrolling in a *Run* if enrollment is closed.

R_{48} Show a *Run* by displaying the position of *Participants* on a map.

3.2.6 Satisfying Goals

The following is an analysis, for each goal of the system to be, of the requirements R_n and the domain assumptions D_n that satisfy the goal itself and of the use cases relative to it.

Data4Help

G_1 Collect *User data* through *Smart Wearables*.

- R_2 At sign up, *User* must provide: first name, last name, SSN, gender, date of birth, email and password.
- R_4 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*.
- R_6 Identify a *User* by their identifier.
- R_8 Receive *User data*.
- R_9 Validate *User data*.
- R_{10} Authenticate *User data*.
- R_{11} Store collected *User data* in Data4Help Database.

- D_2 *User data* collected at a certain instant corresponds to the actual status (GPS position and health data) of the *User* at that precise moment.
- D_3 The maps in use accurately represent the world.
- D_8 *Smart Wearables* are correctly worn by *Users*.
- D_9 Data4Help and all *Services*, including AutomatedSOS and Track4Run, are always online.
- D_{10} *Users* own a working smartphone which is always connected to the Internet.
- D_{11} *Users* own a working *Smart Wearable* which is always connected to the *User*'s smartphone.

D_{12} Either smartphones or *Smart wearable* have a working and active GPS.

U_1 User Sign Up

U_3 User Login

U_5 Data4Help collects User data

G_2 Send specific *User data* to *Third parties* only if *User* consent was given after *Third party* access request.

R_3 At sign up, *Third party* must provide a company name.

R_5 At sign up, *Third party* must accept *Terms and conditions*.

R_1 Unregistered individuals and companies must not be able to use Data4Help.

R_6 Identify a *User* by their identifier.

R_7 Query Data4Help Database for a *User* by their identifier.

R_{11} Store collected *User data* in Data4Help Database.

R_{12} Retrieve specific *User data* by querying Data4Help Database.

R_{13} Receive *Third party* data access request.

R_{14} Validate *Third party* data access request.

R_{15} Authenticate *Third party* data access request.

R_{16} Forward *User data* access request to the specific *User*.

R_{17} Receive *User* consent approval or denial with respect to *Third party* data access requests.

R_{18} Check if a specific *User* gave consent to a specific *Service*.

R_{19} Send specific *User data* to the requesting *Third party*.

R_{20} Not send specific *User data* to the requesting *Third party* if the specific *User* denied consent.

U_2 Third party Sign Up

U_4 Third party Login

U_5 Data4Help collects User data

U_6 Third party requests User data

U_{11} User revokes consent to a Service

G_3 Send anonymized requested *Group data* to *Third parties* if the group it refers to is made up of 1000 or more *Users*.

R_3 At sign up, *Third party* must provide a company name.

R_5 At sign up, *Third party* must accept *Terms and conditions*.

R_{11} Store collected *User data* in Data4Help Database.

R_{12} Retrieve specific *User data* by querying Data4Help Database.

R_{13} Receive *Third party* data access request.

R_{14} Validate *Third party* data access request.

R_{15} Authenticate *Third party* data access request.

R_{21} *Third party* must be able to set specific constraints to define a group of *Users*: age, sex, residence.

R_{22} Check how many *Users* the requested *Group data* refers to.

R_{23} Properly anonymize *Group data*.

R_{24} Send *Group data* to the requesting *Third party*.

R_{25} Not send *Group data* if the group it refers to is made up of less than 1000 *Users*.

U_2 Third party Sign Up

U_4 Third party Login

U_5 Data4Help collects User data

U_7 Third party requests Group data

G_4 Send *User data* and *Group data* to subscribed authorized *Third parties* as soon as they are produced.

R_3 At sign up, *Third party* must provide a company name.

R_5 At sign up, *Third party* must accept *Terms and conditions*.

R_7 Query Data4Help Database for a *User* by their identifier.

R_8 Receive *User data*.

R_9 Validate *User data*.

R_{10} Authenticate *User data*.

R_{18} Check if a specific *User* gave consent to a specific *Service*.

R_{19} Send specific *User data* to the requesting *Third party*.

R_{20} Not send specific *User data* to the requesting *Third party* if the specific *User* denied consent.

R_{21} *Third party* must be able to set specific constraints to define a group of *Users*: age, sex, residence.

R_{22} Check how many *Users* the requested *Group data* refers to.

R_{23} Properly anonymize *Group data*.

R_{24} Send *Group data* to the requesting *Third party*.

R_{25} Not send *Group data* if the group it refers to is made up of less than 1000 *Users*.

R_{26} Receive *Third party* subscription request.

R_{27} Validate *Third party* subscription request.

R_{28} Authenticate *Third party* subscription request.

R_{29} Automatically send new *User data* to subscribed authorized *Third parties* as soon as they are produced.

D_9 Data4Help and all *Services*, including AutomatedSOS and Track4Run, are always online.

D_{10} *Users* own a working smartphone which is always connected to the Internet.

D_{11} *Users* own a working *Smart Wearable* which is always connected to the *User*'s smartphone.

U_5 Data4Help collects User data

U_8 Third party subscribes to New User data

U_9 Third party subscribes to New Group data

U_{11} User revokes consent to a Service

G_5 Allow *Users* to manage their subscription to *Services* and to Data4Help.

R_2 At sign up, *User* must provide: first name, last name, SSN, gender, date of birth, email and password.

R_4 At sign up, *User* must accept *Terms and conditions*, including the *Privacy statement*.

R_6 Identify a *User* by their identifier.

R_7 Query Data4Help Database for a *User* by their identifier.

R_{12} Retrieve specific *User data* by querying Data4Help Database.

R_{30} Allow *Users* to add a *Service*.

R_{31} Allow *Users* to unsubscribe from a *Service*.

R_{32} Send to a specific *User* all their data stored by TrackMe.

R_{33} Delete all data of a specific *User*.

R_{34} Allow *Users* to request all their data stored by TrackMe at any time.

R_{35} Allow *Users* to request the deletion of all their data stored by TrackMe at any time.

U_{10} User adds a Service

U_{11} User revokes consent to a Service

AutomatedSOS

G_6 Analyze *User data* to check whether or not a *User* is a *User in need*.

- R_6 Identify a *User* by their identifier.
- R_7 Query Data4Help Database for a *User* by their identifier.
- R_8 Receive *User data*.
- R_9 Validate *User data*.
- R_{10} Authenticate *User data*.
- R_{29} Automatically send new *User data* to subscribed authorized *Third parties* as soon as they are produced.
- R_{36} Receive *User data* from Data4Help.
- R_{37} Compare *User data* against certain thresholds.
- R_{39} *User* must be able to reactivate AutomatedSOS monitoring.

- D_2 *User data* collected at a certain instant corresponds to the actual status (GPS position and health data) of the *User* at that precise moment.
- D_6 AutomatedSOS and Track4Run are subscribed to new *User data* of all their *Users*.
- D_9 Data4Help and all *Services*, including AutomatedSOS and Track4Run, are always online.
- D_{10} *Users* own a working smartphone which is always connected to the Internet.
- D_{11} *Users* own a working *Smart Wearable* which is always connected to the *User*'s smartphone.
- D_{12} Either smartphones or *Smart wearable* have a working and active GPS.

- U_5 Data4Help collects User data
- U_{12} User in need assisted by AutomatedSOS

G_7 Send an ambulance to the last position of a *User in need*.

- R_6 Identify a *User* by their identifier.
- R_7 Query Data4Help Database for a *User* by their identifier.
- R_{11} Store collected *User data* in Data4Help Database.
- R_{12} Retrieve specific *User data* by querying Data4Help Database.
- R_{38} Call local emergency services providing necessary *User data* of *User in need*.

D_3 The maps in use accurately represent the world.

D_6 AutomatedSOS and Track4Run are subscribed to new *User data* of all their *Users*.

D_7 When AutomatedSOS needs to send an ambulance to a *User in need* it forwards the request to local emergency services, which eventually dispatch an ambulance.

U_5 Data4Help collects User data

U_{12} User in need assisted by AutomatedSOS

Track4Run

G_8 Allow *Organizers* to create a *Run*.

R_{41} At sign up, *Organizers* must provide: first name, last name, email and password.

R_{42} At sign up, *Organizers* must accept *Terms and conditions*.

R_{43} Allow *Organizers* to create a *Run*, defining: name, path, date, maximum number of *Participants* and enrollment closure date.

U_{13} Organizer Sign Up

U_{14} Organizer creates a Run

G_9 Allow *Users* to enroll in a *Run* as *Participants*.

R_2 At sign up, *User* must provide: first name, last name, SSN, gender, date of birth, email and password.

R_{44} Allow *Users* to enroll in an existing *Run*.

R_{45} Prevent a *User* from enrolling in a *Run* if the maximum number of *Participants* was already reached.

R_{46} Prevent a *User* from enrolling in a *Run* if it already started or finished.

R_{47} Prevent a *User* from enrolling in a *Run* if enrollment is closed.

U_{15} User enrolls in a Run

G_{10} Allow *Spectators* to watch a *Run*.

R_{40} Receive *User data* from Data4Help.

R_{48} Show a *Run* by displaying the position of *Participants* on a map.

D_3 The maps in use accurately represent the world.

D_6 AutomatedSOS and Track4Run are subscribed to new *User data* of all their *Users*.

D_8 *Smart Wearables* are correctly worn by *Users*.

D_9 Data4Help and all *Services*, including AutomatedSOS and Track4Run, are always online.

D_{10} *Users* own a working smartphone which is always connected to the Internet.

D_{11} *Users* own a working *Smart Wearable* which is always connected to the *User*'s smartphone.

D_{12} Either smartphones or *Smart wearable* have a working and active GPS.

U_5 Data4Help collects User data

U_{16} Spectator watches a Run

3.3 Performance Requirements

PR_1 AutomatedSOS must call local emergency services within 5 seconds from the moment it identified a *User* as *User in need*.

Requirement PR_1 refers to the reaction time of the system to be. Being able to respect this requirement can mean saving a person's life, besides making the system to be reliable. This is why AutomatedSOS must always be ready to call local emergency services as soon as needed.

The system to be depends on external parties and pieces of hardware. These have a great impact on the performance and on the functions of the system to be. It is possible to put constraints on the quality of external pieces of hardware compliant with the system to be (e.g. sensors quality and accuracy, *Smart wearable* connection to smartphone). By doing this, the system will be able to send accurate and precise data to *Services*, positively contributing to their performance level.

3.4 Design Constraints

3.4.1 Standards Compliance

- GDPR 2016/679 (EU) - General Data Protection Regulation
- ISO/IEC/IEEE 29148:2011 - Standard on requirement engineering

3.4.2 Hardware Limitations

Data4Help and the services built on top of it are software applications with no strict hardware limitations. Therefore the only ones that are noteworthy are the following:

- Working smartphone with Internet connectivity and GPS.
- Working *Smart wearable* able to send *User data* to the *User*'s smartphone.

3.5 Software System Attributes

3.5.1 Reliability

The reliability of the system mainly depends on that of *Smart Wearables* in use whose purpose is to collect *User data*. It also depends on the reliability of the Internet connection when it comes to sending data to Data4Help and to *Third parties*.
AutomatedSOS downtime must be minimal, in order to provide assistance to all *Users in need*. Fault detection time must be kept minimum so as to promptly identify a fault and repair it as soon as possible.

3.5.2 Availability

The system must offer the maximum availability, granting its service every day at any time. The lack of service must be minimal.

AutomatedSOS must be active every day at any time. The lack of service is acceptable only if it is due to maintenance. AutomatedSOS *Users* must have received a warning 48 hours before, and must be noticed again one hour before. Even in this case, the lack of service must be kept to a minimum.

3.5.3 Security

The system to be does not have particular security concerns except the ones related to privacy. The login of *Users* and especially of *Third parties* must be very safe (using state of the art login techniques is recommended) to avoid unauthorized individuals to access private information of *Users*. Moreover, the means of communication must be encrypted to save the confidentiality of information sent to Data4Help and to *Third parties*.

3.5.4 Portability

Portability of *User data* from a device to another is possible by entering personal login data, also for devices with different operating systems. Personal data and settings are stored in a database and they are downloaded when a new device is connected.

Chapter 4

Formal Analysis using Alloy

4.1 Data4Help

```
-- SIGNATURES

sig Individual{
    age: one Int
}{age >=0 }

one sig Data4Help{
    userdata: Data one -> one User,
    groupdata: Data -> AnonUsersGroup
}

sig HealthData{
    timestamp: one Int,
    data: one Data
}{timestamp>0}

sig Position{
    timestamp: one Int,
    data: one Data,
    lat: one Int,
    long: one Int
}{timestamp>0}
```

```
sig Data {
    healthData: set HealthData,
    positions: set Position
}

sig User {
    individual: one Individual
}

sig AnonUsersGroup {
    filter: one Filter
}

sig Filter{
    age: one Int
}

sig Service{
    thirdParty: one ThirdParty,
    users: set User,
    subscribedUsers: set User
}{subscribedUsers in users}

sig ThirdParty {
    services: set Service,
    userdata: Data lone -> lone User,
    groupdata: Data -> AnonUsersGroup
}{services.users = Data.userdata}

-- FUNCTIONS

fun retrieveUserData[d4h: Data4Help, u: User] : Data->User {
    d4h.userdata :> u
}
```

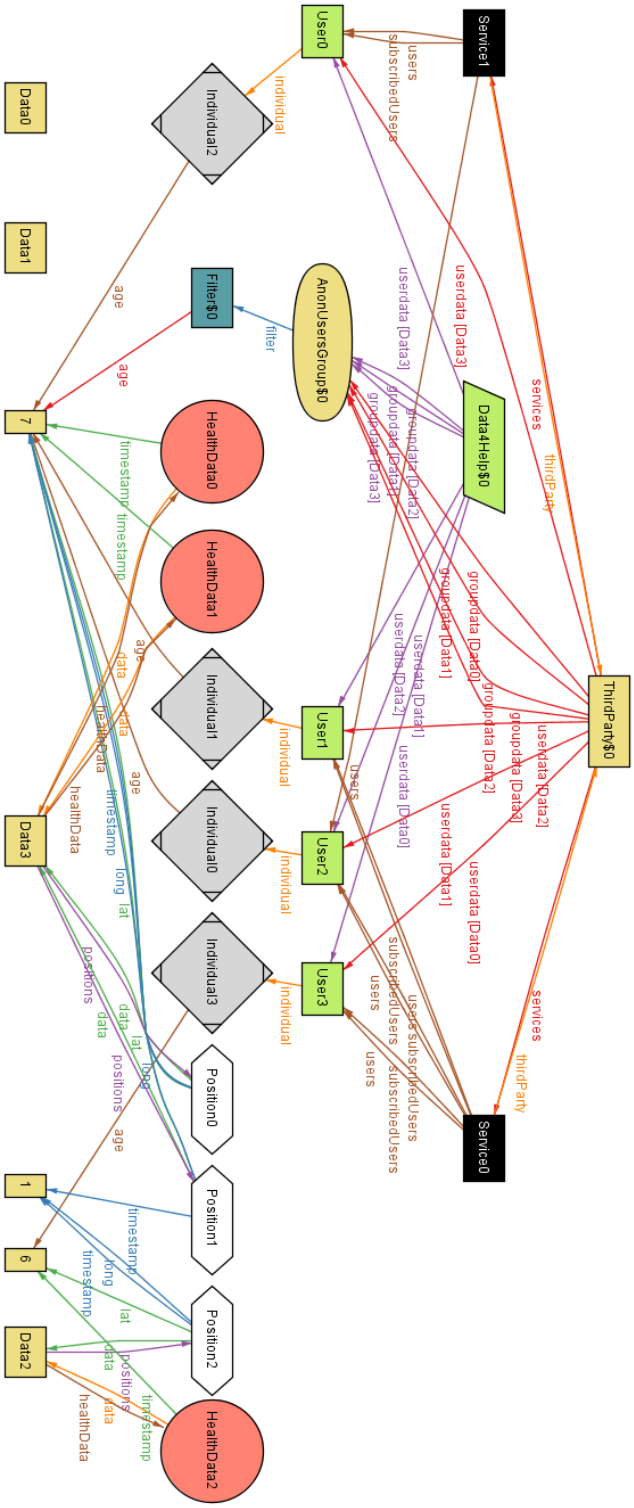
```
fun getAllDataOfAnonUsersGroup[d4h: Data4Help, aug: AnonUsersGroup] : set Data {
    (d4h.groupdata).aug
}

fun getAllUsersFromAnonUsersGroup[d4h: Data4Help, aug: AnonUsersGroup] : set User{
    (getAllDataOfAnonUsersGroup[d4h, aug]).(d4h.userdata)
}

fun retrieveDataToAnonUsersGroupByGroup[d4h: Data4Help, aug:  AnonUsersGroup] :
    Data -> AnonUsersGroup{
    d4h.groupdata :> aug
}

-- FACTS

fact userIndividualRelationshipIsUnique{
    all disj u1, u2:User, i:Individual |
    not (u1.individual = i and u2.individual = i)
}

fact healthDataToDataConnection{
    all d:Data, hd: HealthData |
    hd in d.healthData iff hd.data = d
}

fact postionToDataConnection{
    all d:Data, p: Position |
    p in d.positions iff p.data = d
}

fact userdataInThirdPartyImpliesUserdataInData4Help {
    all d:Data, u: User, d4h:Data4Help, tp:ThirdParty |
    d->u in tp.userdata implies d->u in d4h.userdata
}
```

```
fact userdataInThirdPartyImpliesUserdataInData4Help {
    all d:Data, u: User, d4h:Data4Help, tp:ThirdParty |
    d->u in tp.userdata implies d->u in d4h.userdata
}

fact allHealthDataBelongsToOnlyOneData{
    all hd: HealthData | all disj d1,d2: Data |
    not (hd in d1.healthData and hd in d2.healthData)
}

fact allPositionBelongsToOnlyOneData{
    all p:Position | all disj d1,d2: Data |
    not (p in d1.positions and p in d2.positions)
}

fact serviceCanBeOfOnlyOneThirdParty{
    all s:Service | all disj t1,t2: ThirdParty |
    not (s in t1.services and s in t2.services)
}

fact serviceThirdPartyRelationshipIsUnique{
    all s:Service, tp:ThirdParty |
    s.thirdParty = tp iff s in tp.services
}

fact anonUsersGroupsAreMadeUpOfMoreThan2Users {
    all aug: AnonUsersGroup, d4h: Data4Help |
    #(getAllUsersFromAnonUsersGroup[d4h, aug]) > 2
    -- #(getAllUsersFromAnonUsersGroup[d4h, aug]) >= 1000
}

fact anonUsersGroupsAreMadeOfUsersThatRespectFilter {
    all aug: AnonUsersGroup, d4h: Data4Help |
    (getAllUsersFromAnonUsersGroup[d4h, aug]).individual.age = aug.filter.age
```

```
}

fact dataIsOfAnonUsersGroup {
    all tp:ThirdParty, aug: AnonUsersGroup, d4h: Data4Help |
    aug in Data.(tp.groupdata) implies getAllDataOfAnonUsersGroup[d4h, aug]
        in tp.groupdata.AnonUsersGroup
}

-- PREDICATES

pred sendUserData [u: User, s: Service, tp:ThirdParty, tp': ThirdParty, d4h: Data4Help] {
    u in s.users
    tp = s.thirdParty
    tp'.services = tp.services
    tp'.groupdata = tp.groupdata
    tp'.userdata = tp.userdata + retrieveUserData[d4h, u]
}

pred sendGroupData [f: Filter, s: Service, tp:ThirdParty, tp': ThirdParty, d4h: Data4Help,
    aug: AnonUsersGroup] {
    aug in Data.(d4h.groupdata)
    f in aug.filter
    tp = s.thirdParty
    tp'.services = tp.services
    tp'.userdata = tp.userdata
    tp'.groupdata = tp.groupdata + retrieveDataToAnonUsersGroupByGroup[d4h,aug]
}

pred show {}

-- ASSERTIONS

assert sendUserDataOkay{
    all tp', tp: ThirdParty, u:User, s:Service, d4h:Data4Help |
```

```
    retrieveUserData[d4h,u] not in tp.userdata and sendUserData[u, s, tp, tp', d4h]
        implies retrieveUserData[d4h, u] in tp'.userdata
}

assert sendGroupDataOkay{
    all tp,tp': ThirdParty, f: Filter, s: Service, d4h: Data4Help, aug: AnonUsersGroup |
    sendGroupData[f, s, tp, tp', d4h, aug] implies ( aug in Data.(tp'.groupdata) and
        getAllDataOfAnonUsersGroup[d4h, aug] in tp'.groupdata.AnonUsersGroup)
}

-- CHECKS and RUN

check sendUserDataOkay for 10
check sendGroupDataOkay for 10

run show for 4 but exactly 4 User, 4 Data, 1 ThirdParty, 2 Service, 1 AnonUsersGroup,
    3 HealthData, 3 Position
```

```
3 commands were executed. The results are:
  #1: No counterexample found. sendUserDataOkay may be valid.
  #2: No counterexample found. sendGroupDataOkay may be valid.
  #3: Instance found. show is consistent.
```



4.2 AutomatedSOS

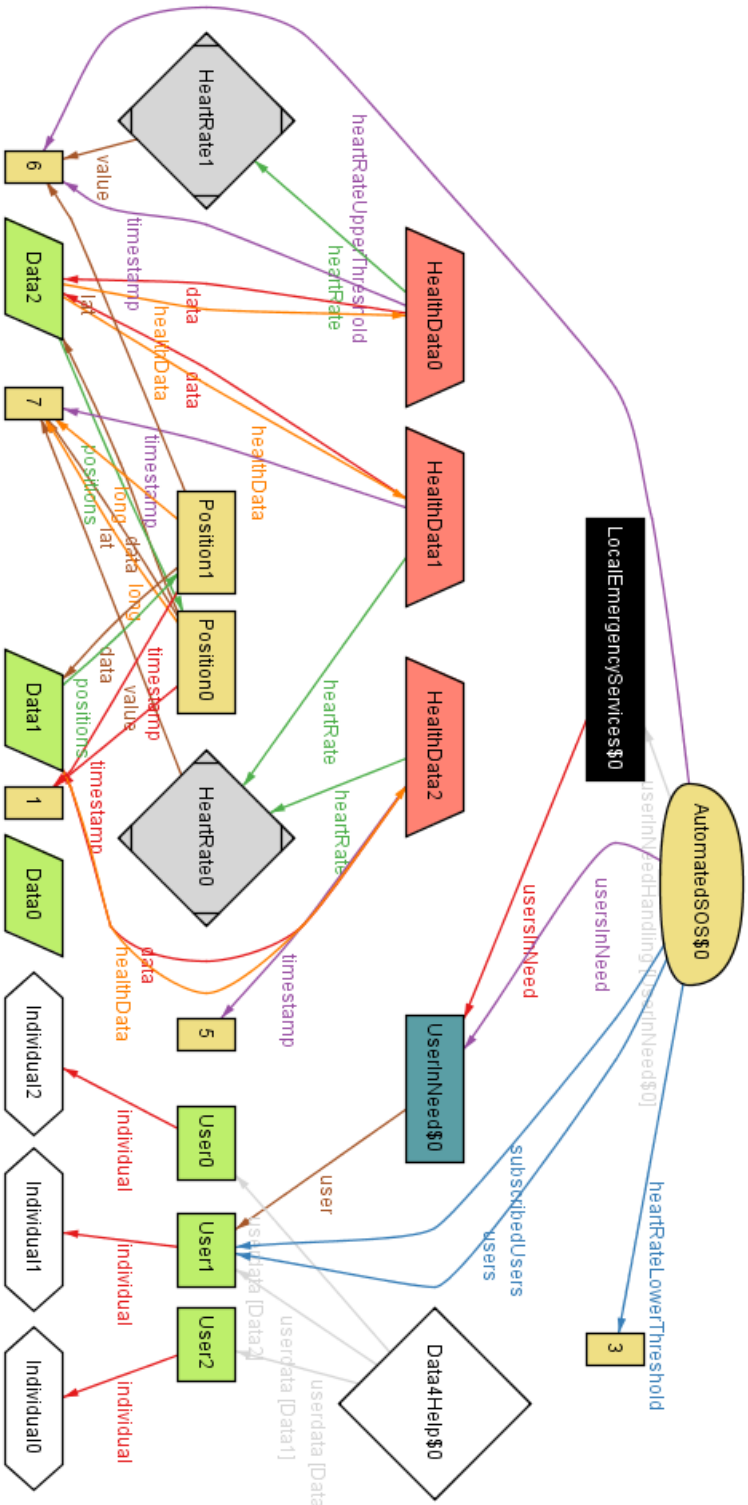
```
-- SIGNATURES

sig Individual {}

sig HealthData {
    timestamp: one Int,
    heartRate: one HeartRate,
    data: one Data
}{timestamp>0}

sig HeartRate{
    value: one Int
}{value >=0}

sig Position{
    timestamp: one Int,
    data: one Data,
    lat: one Int,
    long: one Int
}{timestamp>0}

sig Data {
    healthData: set HealthData,
    positions: set Position
}

sig User {
    individual: one Individual
}

one sig Data4Help{
    userdata: Data one -> one User
}
```

```
sig Service{
    users: set User,
    subscribedUsers: set User
}{subscribedUsers in users}

one sig AutomatedSOS extends Service{
    heartRateLowerThreshold: one Int,
    heartRateUpperThreshold: one Int,
    usersInNeed: set UserInNeed,
    userInNeedHandling: UserInNeed -> one LocalEmergencyServices
}{ heartRateLowerThreshold = 3 and heartRateUpperThreshold = 6
    and subscribedUsers = users
    and (userInNeedHandling.LocalEmergencyServices) = usersInNeed}
-- { heartRateLowerThreshold = 40 and heartRateUpperThreshold = 200}

sig LocalEmergencyServices{
    usersInNeed: set UserInNeed
}

sig UserInNeed{
    user: one User
}

-- FUNCTIONS

fun retrieveUserData[d4h: Data4Help, u: User] : set Data {
    d4h.userdata.u
}

-- FACTS

fact userIndividualRelationshipIsUnique{
    all disj u1, u2:User, i:Individual |
    not (u1.individual = i and u2.individual = i)
```

```
}

fact userUserInNeedRelationshipIsUnique{
    all disj uin1, uin2: UserInNeed, u: User |
    not (uin1.user = u and uin2.user = u)
}

fact healthDataToDataConnection{
    all d:Data, hd: HealthData |
    hd in d.healthData iff hd.data = d
}

fact postionToDataConnection{
    all d:Data, p: Position |
    p in d.positions iff p.data = d
}

fact allHealthDataBelongsToOnlyOneData{
    all hd: HealthData | all disj d1,d2: Data |
    not (hd in d1.healthData and hd in d2.healthData)
}

fact allPositionBelongsToOnlyOneData{
    all p:Position | all disj d1,d2: Data |
    not (p in d1.positions and p in d2.positions)
}

fact userInNeedDefinition{
    all uin: UserInNeed, asos: AutomatedSOS, d4h: Data4Help| some hr: HeartRate |
    ((hr.value < asos.heartRateLowerThreshold
        or hr.value > asos.heartRateUpperThreshold)
    and hr in retrieveUserData[d4h, uin.user].healthData.heartRate)
    iff
    (uin.user in asos.subscribedUsers and uin in asos.usersInNeed)
```

```
}

fact ifUserInNeedIsAssignedToLESTheyAreInLESUsersInNeed{
    all uin: UserInNeed, asos: AutomatedSOS, les: LocalEmergencyServices |
    uin in asos.userInNeedHandling.les iff uin in les.usersInNeed
}

pred show {}

-- RUN

run show for 3 but exactly 3 User, 2 HeartRate, 0 Service,
    1 UserInNeed, 1 LocalEmergencyServices
```

```
Executing "Run show for 3 but exactly 3 User, 2 HeartRate, 0 Service, 1 UserInNeed, 1 LocalEmergencyServices"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  3562 vars. 344 primary vars. 9407 clauses. 168ms.
  Instance found. Predicate is consistent. 71ms.
```



4.3 Track4Run

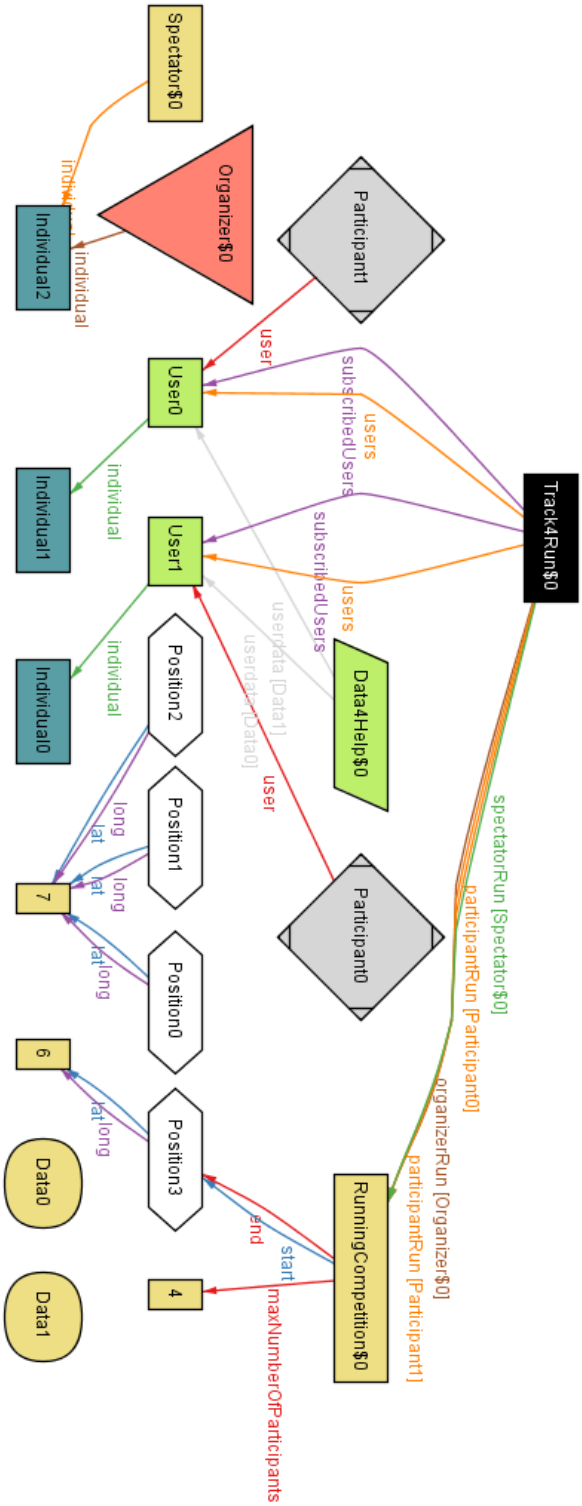
```
-- SIGNATURES
sig Individual {}

sig User {
    individual: one Individual
}

sig Participant {
    user: one User
}

sig Spectator {
    individual: one Individual
}

sig Organizer {
    individual: one Individual
}

sig Data{}

sig Position{
    lat: one Int,
    long: one Int
}

one sig Data4Help{
    userdata: Data one -> one User
}

sig Service{
    users: set User,
    subscribedUsers: set User
```

```
}{subscribedUsers in users}

one sig Track4Run extends Service {
    organizerRun: Organizer one -> RunningCompetition,
    participantRun: Participant -> one RunningCompetition,
    spectatorRun: Spectator -> one RunningCompetition
} {(participantRun.RunningCompetition).user in subscribedUsers}

sig RunningCompetition {
    start: one Position,
    end: one Position,
    maxNumberOfParticipants: one Int
}

-- FUNCTIONS

fun getRunningCompetitionForParticipant[t4r: Track4Run, p:Participant] :
    RunningCompetition {
    (t4r.participantRun)[p]
}

fun getRunningCompetitionForSpectator[t4r: Track4Run, s:Spectator] :
    RunningCompetition {
    (t4r.spectatorRun)[s]
}

fun getNumberOfParticipantsForRunningCompetition[t4r: Track4Run,
    r: RunningCompetition] : Int {
    #(t4r.participantRun.r)
}

-- FACTS

fact userIndividualRelationshipIsUnique{
```

```
    all disj u1, u2:User, i:Individual |
    not (u1.individual = i and u2.individual = i)
}

fact organizerIndividualRelationshipIsUnique{
    all disj o1, o2:Organizer, i:Individual |
    not (o1.individual = i and o2.individual = i)
}

fact userCantAppearTwiceInRunningCompetition{
    all disj p1, p2:Participant, t4r:Track4Run |
    p1.user = p2.user implies getRunningCompetitionForParticipant[t4r, p1]
        != getRunningCompetitionForParticipant[t4r, p2]
}

fact individualCantAppearTwiceAsSpectatorOfRunningCompetition{
    all disj s1, s2:Spectator, t4r:Track4Run |
    s1.individual = s2.individual implies getRunningCompetitionForSpectator[t4r, s1]
        != getRunningCompetitionForSpectator[t4r, s2]
}

fact limitParticipantsToMaxNumberOfParticipants{
    all r:RunningCompetition, t4r:Track4Run |
    getNumberOfParticipantsForRunningCompetition[t4r, r]
        <= r.maxNumberOfParticipants
}

fact participantCannotBeSpectator{
    all p:Participant, s:Spectator,t4r:Track4Run |
    p.user.individual = s.individual implies getRunningCompetitionForParticipant[t4r,p]
        != getRunningCompetitionForSpectator[t4r,s]
}
```

```
-- PREDICATES
pred userEnrollsInRunningCompetition[p:Participant, u:User, t4r,t4r':Track4Run,
    r:RunningCompetition] {
    u in t4r.users
    getNumberOfParticipantsForRunningCompetition[t4r, r]
        < r.maxNumberOfParticipants
    p.user = u
    t4r'.organizerRun = t4r.organizerRun
    t4r'.spectatorRun = t4r.spectatorRun
    t4r'.participantRun = t4r.participantRun + p->r
}

pred show{}

-- ASSERTIONS
assert userCanEnrollWhenNumberOfParticipantsIsNotMax{
    all u: User, t4r, t4r':Track4Run, p:Participant, r:RunningCompetition |
    p->r not in t4r.participantRun and userEnrollsInRunningCompetition[p,u,t4r, t4r',r]
        implies p->r in t4r'.participantRun
}

-- CHECKS and RUN

check userCanEnrollWhenNumberOfParticipantsIsNotMax for 10

run show for 4 but exactly 0 Service, 2 Participant, 2 Data, 2 Spectator,
    1 Organizer, 1 RunningCompetition
```

```
2 commands were executed. The results are:
  #1: No counterexample found. userCanEnrollWhenNumberOfParticipantsIsNotMax may be valid.
  #2: Instance found. show is consistent.
```



Chapter 5

Effort Spent

Gargano Jacopo Pio Total hours of work: 64h

- 2h Reading of Project Delivery Document, General LaTex setting.
- 1h Scope
- 3h RASD Review homework
- 3h Product functions - Data4Help
- 1h Creating subfiles structure in Latex
- 4h Functional Requirements and Goals definition
- 4h Class diagrams, Goals redefinition and revision, Document Structure fix
- 5h Use Cases
- 3h Scenarios
- 2h General revision
- 1h State Charts
- 3h Requirements
- 3h Satisfying Goals
- 3h Alloy, General revision
- 3h Add Constraints, Shared phenomena review
- 2h Class Diagrams
- 3h General revision, Performance Requirements

- 1h Meeting with Professor
- 5h Alloy
- 10h Alloy
- 2h General Revision

Giannetti Cristian Total hours of work: 62.5h

- 2h Reading of Project Delivery Document, General LaTex setting.
- 1h Goals
- 3h RASD Review homework
- 1h Assumptions
- 3h Functional Requirements, Goals
- 2h Class diagrams
- 3h Goals revision, Domain Assumptions
- 3h Use Cases
- 1h State charts
- 3h Mockup
- 0.5h Availability, Portability
- 3h State charts
- 1h Requirements
- 1h Use cases diagrams
- 3h Sequence diagrams
- 5h Mockup
- 1h Sequence Diagrams
- 2h Mockup
- 4h Mockup
- 3h Sequence Diagrams, Use Case Diagrams
- 2h General Revision
- 3h Diagrams Revision
- 10h Alloy
- 2h General Revision

Haag Federico Total hours of work: 47h

- 2h Reading of Project Delivery Document, General LaTex setting.
- 0.5h Purpose
- 3h RASD Review homework
- 1h User characteristics
- 4h Functional Requirements, Goals
- 1h Class diagrams
- 3h General revision,Product perspective
- 1h Revision of domain assumptions, Sse cases
- 1h Software Interfaces
- 0.5h Hardware Interfaces
- 6h General revision, Use Case diagrams, Sequence diagrams, Requirements
- 2h Revision of sequence diagram, Alloy
- 1h Meeting with Professor
- 1h Revision of Requirements
- 1h Revision of Class diagrams
- 3h Alloy
- 2h General revision, Effort Spent, Communication Interfaces, Fix of Definitions
- 2h Document Structure, Software System Attributes
- 10h Alloy
- 2h General Revision

Chapter 6

References

1 E. Di Nitto. *Lecture Slides*. Politecnico di Milano.

2 E. Di Nitto. *Mandatory Project Assignment AY 2018-2019*. Politecnico di Milano.

3 ISO/IEC/IEEE 29148:2011. *Standard on requirement engineering*. https://standards.ieee.org/standard/29148-2011.html.