RASD

Requirement Analysis and Specification Document

Data4Help, AutomatedSOS and Track4Run

Irene Nizzoli, Isabella Piacentini, Elio Salvini
10483798 10508831 10490058

POLITECNICO DI MILANO

Summary

1. Introduction 3
1.1 Purpose 3
1.1.1 Goals 3
1.2 Scope 4
1.2.1 Description of the given problem 4
1.2.2 World, shared and machine phenomena 4
1.3 Definitions, Acronyms, Abbreviations 5
1.3.1 Definitions 5
1.3.2 Acronyms 5
1.3.3 Abbreviations 5
1.4 Document Structure 5
2. Overall Description 7
2.1 Product perspective 7
2.2 Product Functions 9
2.2.1 Data Management 9
2.2.2 Request Management 9
2.2.3 Request in case of need 9
2.2.4 Track Runners 10
2.3 User Characteristics 10
2.4 Domain Assumptions 10
3. Specific Requirements 11
3.1 External Interface Requirements 11
3.1.1 User interfaces 11
3.1.2 Hardware interfaces 13
3.1.3 Software interfaces 13
3.1.4 Communication interfaces 13
3.2 Scenarios 14
3.2.1 Scenario 1 14
3.2.2 Scenario 2 14
3.2.3 Scenario 3 14
3.2.4 Scenario 4 14
3.2.5 Scenario 5 14
3.3 Functional Requirements 15

3.3.1 Goals, requirements and domain assumptions 15
3.3.2 Use case diagram 17
3.3.3 Use cases 17
3.3.4 Traceability Matrix 21
3.3.5 Sequence Diagrams 22
3.4 Performance Requirements 25
3.5 Design Constraints 25
3.5.1 Standards Compliance 25
3.5.2 Hardware Limitations 25
3.5.3 Other Constraints 25
3.6 Software system attributes 25
3.6.1 Reliability 25
3.6.2 Availability 26
3.6.3 Security 26
3.6.4 Maintainability 26
3.6.5 Portability 26
4. Formal Analysis Using Alloy 27
4.1 Alloy Results 31
4.2 World Generated 32
5.Effort Spent 33
5.1 33
5.2 33
5.3 33
6.References 34

1. Introduction

1.1 Purpose

This is a Requirement Analysis and Specification Document (RASD) which purpose is to describe the goals, requirements and domain assumptions of the system. It will contain the non-functional limitations and general use cases to better describe the client's needs and the typical usage of the application. This document will be used by the developers of the software during the implementation phase but also during analysis, testing and contractual discussions.

Data4Help general purpose is to give access of health status information to third parties either about groups or single individuals. The application should ensure tracked client's privacy and anonymity and also allow third parties to subscribe to certain request and receive new data as soon as they are updated. In general, third parties, such as big companies, could use this system to gain a better general knowledge of possible clients' needs and habits to improve their business. On the other hand, it could be used by individuals that have the desire to monitor the status of an elderly parent or a child, given the permission of the latter individuals.

AutomatedSOS will use the same data acquired by Data4Help with the goal of helping elderly people in need of immediate medical aid. The system will work by analysing the data obtained and checking the health parameters continuously. This should provide help for people in need way faster than they would have received without this system.

Track4Run will, as well as AutomatedSOS, use the features offered by Data4Help. The purpose in this case is to be able to track participants during a run. The system should also allow organizers to choose a path for the run and let spectators follow the athletes position on a map during the race.

1.1.1 Goals

[G1]. Third parties can request access to data of some specific individuals
[G2]. Third parties are allowed to request access to anonymized data of groups of individuals
[G3]. Third parties can access the data of specific individuals
[G4]. Third parties are able to access anonymized data of groups of individuals
[G5]. Subscribed users are given the choice to accept or refuse third parties' requests to access their data
[G6]. To subscribed users is guaranteed anonymity

[G7]. Subscribed costumers receive immediate help when they are displaying serious medical conditions

[G8]. Spectators are able to know the position of the athletes on the path of the run

1.2 Scope

1.2.1 Description of the given problem

Data4Help is a service that acquires data from subscribed individuals, after asking for their consent, using devices such as smartwatches or similar. Third parties can also benefit from this service. After registration, they can formulate two types of request. The first one is to access the data of a single individual: for this request the application gives the user the freedom to accept or refuse the request. The second request is to access the data of a group of individuals, in this case it's the application that gives the access to the data and must deny it if the number of individuals is below 1000 (the anonymity could not be guaranteed in this scenario). The application also gives third parties the possibility to subscribe to receive new data as soon as they are available.

TrackMe, through another service, called AutomatedSOS, gives the possibility to elder subscribers to receive immediate help in case of need. This application detects if the user's parameters are below a certain threshold and immediately contacts a medical center to send an ambulance to the exact location of the subscriber.

Another application created by TrackMe is Track4Run. This service allows spectator to track participants in a run using their position on a map. This application allows organizers to create a path for the run and participants to register to the run.

1.2.2 World, shared and machine phenomena

- World phenomena:
  - [W1]. Third parties monitor health status and location of individuals
  - [W2]. Individuals agree to provide their data to TrackMe
  - [W3]. Third parties access to data of individuals (who agreed to give them)
  - [W4]. Third parties access to data of groups of anonymous individuals
  - [W5]. Third parties recognize an individual by his/her CF or SSN
  - [W6]. Organizers define the path of the run
  - [W7]. Runners enroll to a run
  - [W8]. Users see on a map the runners' position

- Shared phenomena:
  - [S1]. Registration of users
  - [S2]. Users activates monitoring of their health status and location
  - [S3]. Users' data acquisition through smartwatches or similar devices
  - [S4]. Forwarding of a third party's request of access to personal data to a specific user
  - [S5]. Users accept or refuse the requests from third parties
  - [S6]. Let the previously saved user's data available to third parties (if the request of access is approved)
  - [S7]. AutomatedSOS acquires health status data of its users from Data4Help
  - [S8]. AutomatedSOS sends an ambulance in the location of its users (if their health status values are below a certain threshold)

- Machine phenomena:
  [M1]. Request to access to anonymous groups data are approved or disapproved by TrackMe's systems on the base of certain conditions
  [M2]. AutomatedSOS analyses users acquired data
  [M3]. AutomatedSOS sends requests to external medical service for an ambulance when the data go below the threshold

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- **Individual**: user of the application whose health status is monitored.
- **Third party**: user of the application who requests health status data acquired from individuals or groups.
- **Data Acquisition Device:** device that can provide data about the health status of an individual.
- **Individual Request**: request (advanced by a third party) to get access to a specific individual's data about his/her health status. In this case the identity of the individual is shown to the third party.
- **Group Request**: request (advanced by a third party) to get access to health status data of anonymous group of individuals.
- **Runner**: *individual* who has subscribed himself to a run.
- **Organizer**: user of the application who organises a run by defining a path.
- **Spectator**: user who can see the position of runners through the application.
- **Path**: path of a run.

1.3.2 Acronyms

- RASD: Requirement Analysis and Specification Document
- API: Application Programming Interface
- GPS: Global Positioning System
- DAD: Data Acquisition Device
- CF: "Codice Fiscale"
- SSN: Social Security Number

1.3.3 Abbreviations

- [Gn]: n^{th} goal
- [Dn]: n^{th} domain assumption
- [Rn]: n^{th} functional requirement
- [Wn]: n^{th} world phenomena
- [Sn]: n^{th} shared phenomena
- [Mn]: n^{th} machine phenomena

1.4 Document Structure

1. **Introduction**: this part presents *purpose* and *scope* of the applications Data4Help, AutomatedSOS and Track4Run. The applications goals are listed and described. In this section there is also an analysis of world, shared and machine phenomena that concerns these applications.

2. **Overall description**: further details about shared phenomena and model domain are provided. Also, most important requirements and domain assumptions are defined. The aim of this part is to focus user needs and to give an overall description of the application's model.
3. **Specific Requirements**: all aspects presented in the previous section are more deeply analysed in this part. All application requirements are listed and their link with goals and domain assumptions is stressed. Use cases are defined, and most important cases are analysed with the support of diagrams. Hardware and software interfaces are analysed. Constrains and limitations of the application are identified; necessary software attributes are enlightened.
4. **Formal Alloy analysis**: More relevant and critical parts of the model described in the previous sections are deeply analysed in this part with the use of Alloy.
5. **Effort spent**: this part lists for each member who worked at this document the number of hours dedicated to each section.
6. **References**: external resources used in the development of the current document.

2. Overall Description

2.1 Product perspective

Data4Help services, and their links with AutomatedSOS and Tarck4Run, are globally described in the class diagram in *figure 1*. This diagram shows the most important class of the applications from a requirement viewpoint.

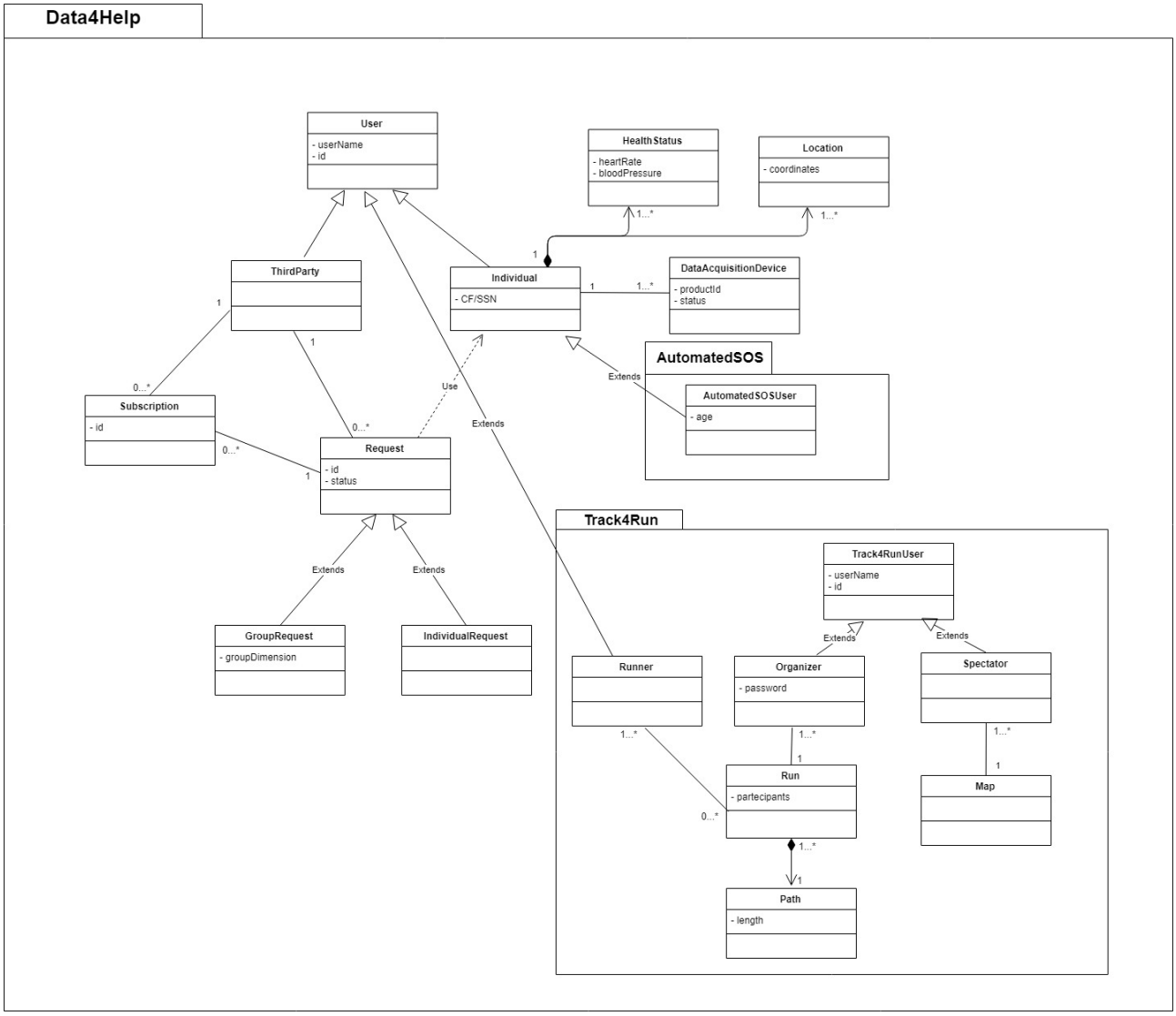


figure 1

Data4Help divides its users in two categories, individuals (those whose health status is monitored) and third parties (those who request access to individual health status data). Also, requests are divided in two categories, group request and individual request. It is possible to create subscription for both types of requests.
To subscribe to AutomatedSOS services it is necessary to subscribe also to Data4Help, indeed Data4Help users contains AutomatedSOS ones.

Track4Run users are split in three classes:

- Data4Help users:
  - *runners*, those who participate to a run;
- Track4Run users:
  - *organizers*, those who creates the path of a run;
  - *spectators*, anyone who want to see runners' position.

Organizers and spectators are exclusively users of Track4Run, while runners are also users of Data4Help.

It is possible for each individual monitored by Data4help to register more than one DAD.
Always for each individual it's possible to keep track of his/her different health statuses and locations in time.

One of the more crucial aspects of Data4Help is the handle of requests of access to individual's data, advanced by third parties. This process is shown in detail in the following diagram, *figure 2*.
As we can see there are three main cases handled by the system, the single request (both groups and individuals), group subscriptions and individual subscription.

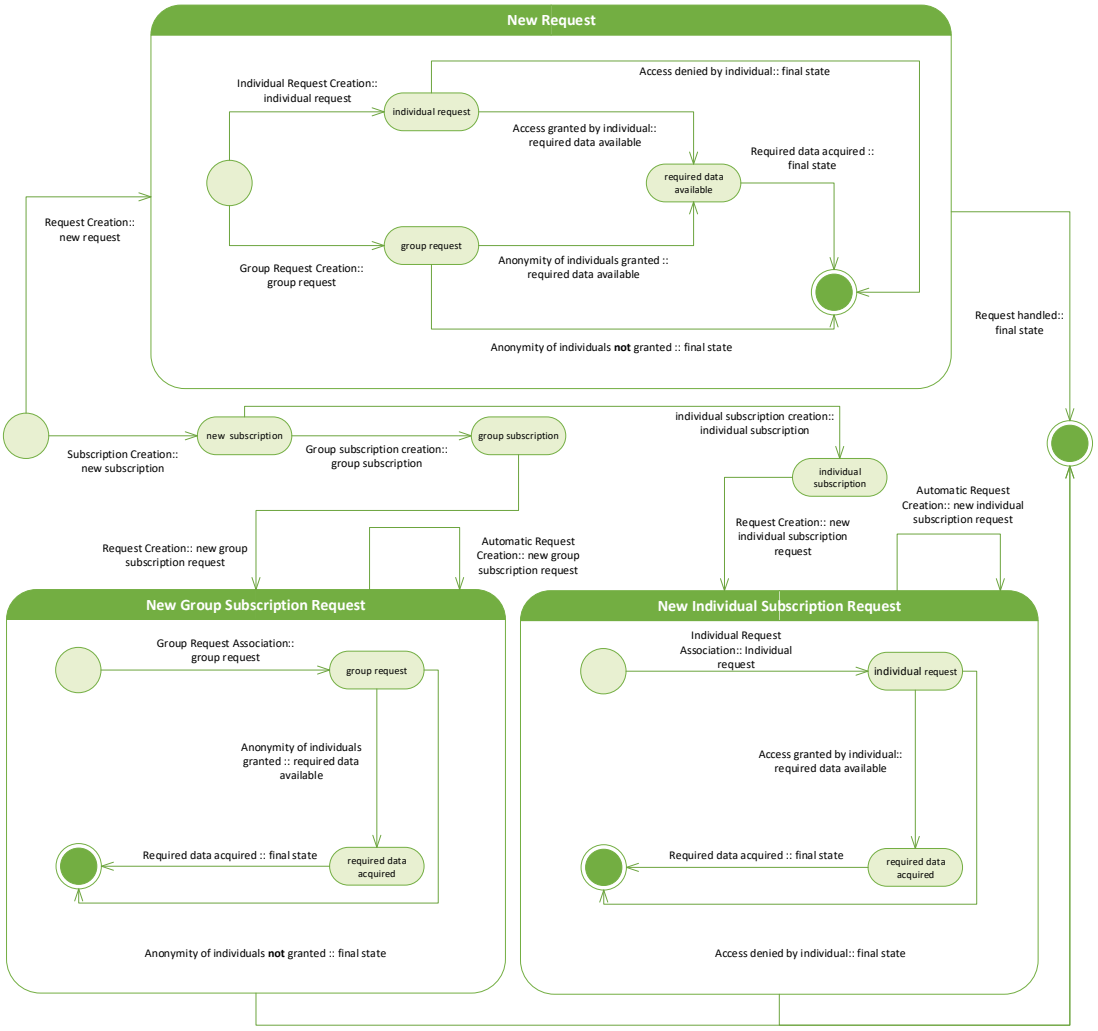


figure 2

2.2 Product Functions

Bearing in mind all the information about the three applications we can divide the most important functions of these systems in 4 groups: two for Data4Help, one for AutomatedSOS and one for Track4Run.

2.2.1 Data Management

This is an essential requirement. The users have to accept the request of the application to access their data. After that Data4Help is able to acquire individual's data and keep a storage of them. In case of accepted request by third parties the application can give them access to the data they need.

Another great characteristic is that the application also let two external services (AutomatedSOS and Track4Run) use the acquired data of the individuals subscribed to those applications. Thanks to this property AutomatedSOS is capable of analyse the data of the users and Track4Run can access to the location of the runners.

2.2.2 Request Management

One of the main functions of the application is the way request from third parties are handled. The system needs to be able to diversify between individual request and group request and proceed accordingly. In the former case the application has to pass the request to the individual and wait for an eventual approval before giving access to the data. In the latter it is instead needed a phase of analysis of the request where the system checks if the number of people that reflect the parameters given by the third parties are over or below the threshold that guarantees anonymity to users. Only in the first situation the third parties receive access to the data which they asked for. Another feature is the management of subscriptions of third parties in regard of specific request: in this case, for example, the application needs to check if the limitation to guarantee privacy are still respected after every update of the users' information.

2.2.3 Request in case of need

The principal function of AutomatedSOS is the ability to send an immediate request for help. First of all, the application needs to acquire the data of the subscribers thanks to Data4Help, then the service needs to constantly analyse them. If the data go below a certain threshold the service instantaneously has to contact an external medical facility providing it with the exact location of the user in need (the location is also acquired thanks to Data4Help).

2.2.4 Track Runners

The main requirement of Track4Run is the possibility to track the runners. The spectators, through a map, are able to locate the exact position of the athletes participating the run. The location of the runners is acquired thanks to the resources of Data4Help.

2.3 User Characteristics

In general, the actors of the application are these:

- Individuals: They represent all the people whose data are acquired by different sorts of tracking devices and stored inside the application.
- AutomatedSOSUsers: Individuals that activated their account on AutomatedSOS with the aim to be continuously monitored to check if there is any need for medical assistance.
- Third Parties: Companies or single individuals that made at least a request or a subscription to the data (health status/location) of other users.
- Organizers: In regard of Track4Run they are the people who choose the path for the run and they define it inside the application.
- Spectators: People who want to follow the position of the athletes during different runs inside the map provided by Track4Run.
- Runners: Individuals that are the participants of the runs that needs to be tracked by the application.

2.4Domain Assumptions

[D1].	By activating the monitoring feature inside the application, the individuals agree that TrackMe acquires their data
[D2].	Third parties know a way to identify the individuals whose data are of interest
[D3].	When the number of individuals whose data satisfy a request is less than 1000, third parties are able to identify the individuals
[D4].	Users have a unique identification code
[D5].	Device sensors can provide an accurate enough current location
[D6].	When the parameters go below a certain threshold AutomatedSOSUsers' health needs immediate medical help
[D7].	Individuals don't take the device sensor off
[D8].	Individuals uses the device sensor correctly
[D9].	Device sensors work correctly and without malfunctions
[D10].	The external medical services are available 24/7 and work correctly
[D11].	External medical services are able to receive automatic emergency reports
[D12].	Runners have a tracking device on their person during the run
[D13].	Organizers decide a path for the run
[D14].	Individuals enroll to the run

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User interfaces

The following mockups represent an idea of how the mobile application will look in the first release.

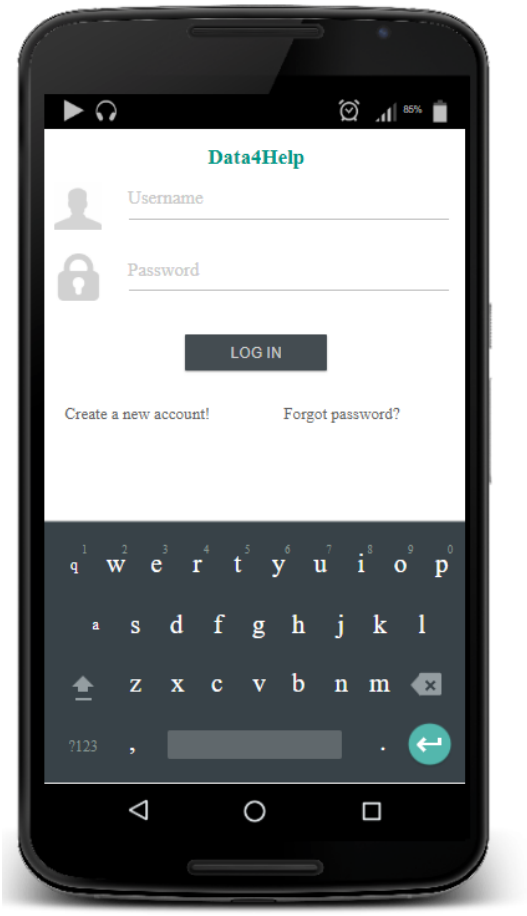


figure 3

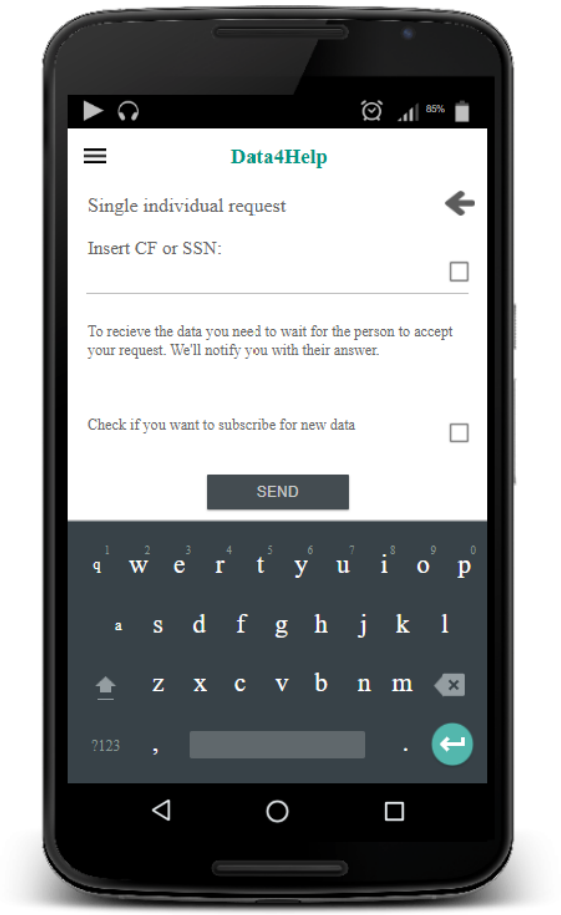


figure 4

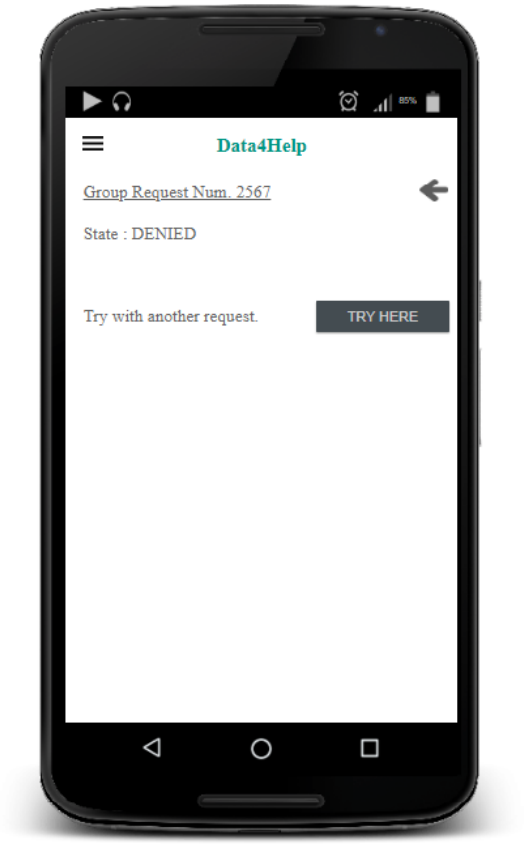


figure 5

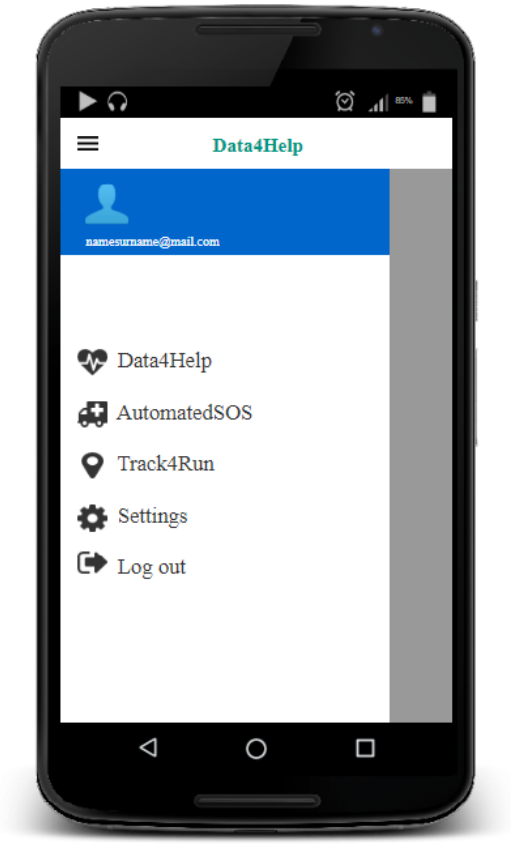


figure 6

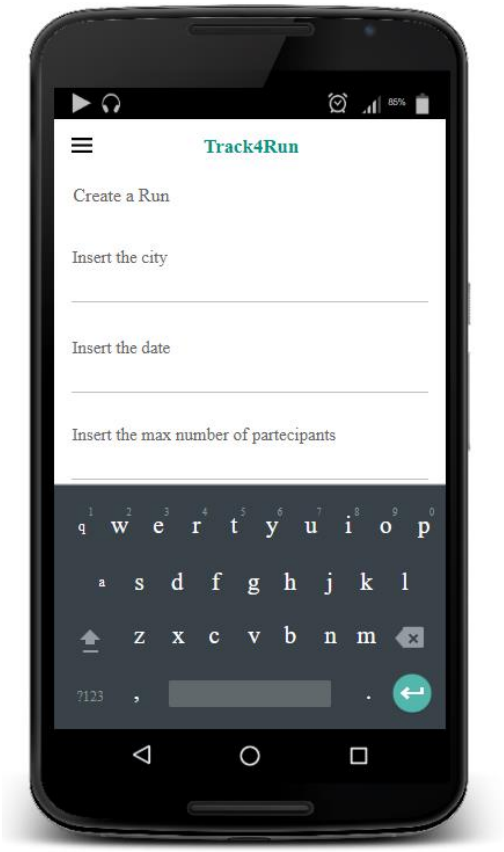


figure 7

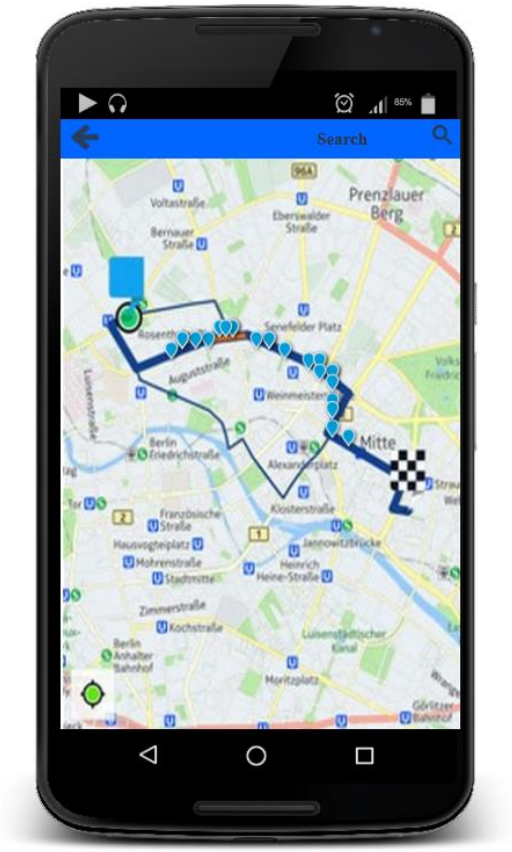


figure 8

3.1.2 Hardware interfaces

In order to provide a correct working of the application, devices that can monitor users' health status (like smartwatches) are needed. Most of these devices need also a smartphone to have access to internet services and to communicate the monitored data. Devices used to monitor user's status, or smartphones directly connected with them, must support GPS to provide Data4Help, AutomatedSOS and Track4Run services.

To gain access to individuals' data, or to follow runners' position during a run, it is possible to use any device able to provide access to web services.

3.1.3 Software interfaces

Services offered by AutomatedSOS and Track4Run needs the support of other external services. Interfacing with other software is essential if we think at the necessity of AutomatedSOS of contacting sanitary systems in case of emergency. While for Track4Run the use of an external software to show street, maps is an easy way to provide its service, avoiding the need of creating virtual maps of the interested areas.

Sanitary emergency systems: AutomatedSOS contacts the sanitary emergency systems if the monitored values of one of its users goes down a certain threshold. The application sends the coordinates of the user in danger and reports the emergency to the sanitary systems.

Street maps services: in order to give to the organizers, the possibility to define the path of the run, Track4Run needs a knowledge of the street map of the area designated for the event. Through the API of an external street maps service the application allows organizers to specify the path and to see its preview in the map. In addition, spectators of the run can see the street map of the area interested by the run, and the position of runners on the map; information required to recreate the map on spectator's device are provided by the external street maps service.

3.1.4 Communication interfaces

Data4Help and Track4Run use a high-level protocol based on TCP to communicate with users' devices (like HTTP for example). While AutomatedSOS exploits communication, protocols based on UDP; this choice is due to the need to be promptly in contacting the emergency systems if a user is in danger.

3.2 Scenarios

3.2.1 Scenario 1

Hannah is working full-time inside an important company and hasn't got much time to go visit her elderly father Mike anymore. She still wishes to check on is conditions sometimes during the day, usually when he goes out to play chess with his friends in park near his house. His father is affected by dementia and could easily loose his way and end up not knowing where to go. She downloads Data4Help for herself and her father and sign him up to be monitored. Then she creates a request using Mike's SSN and sends the requests in the right time slot. This way, if she sees on Data4Help that his father location indicates that he's not going in the right direction she just calls him and helps him find his way to the park without any problems or accidents that may have occurred otherwise.

3.2.2 Scenario 2

Samsung is looking to understand the needs of a group of teenagers in an area of New York. It starts by creating a request inside Data4Help inserting these parameters. Data4Help checks if the number of people reflecting the boundaries are over 1000, which means that it is over the threshold to ensure privacy to the users. Seeing that the response is positive, Data4Help accepts Samsung request and sends the data of the group selected. Samsung decides to subscribe to the data to obtain a better understanding of the habits of the group and so it selects the possibility inside the application. Now every time the data are updated Samsung receives them, given that they still respect the anonymity limitations.

3.2.3 Scenario 3

Henry is 75, lives alone and he's prone to hearth failures. Being worried for his health he subscribes to Data4Help and AutomatedSOS and starts wearing his new smartwatch 24/7. The device is linked to the application, which constantly monitors the data received. After a few months Henry isn't feeling very well, but he collapses before he can call an emergency number. AutomatedSOS detects that his health status is not among normal values and contacts immediately the medical services. After a few minutes an ambulance arrives at Henry's locations and he's given the medical help he needs in time.

3.2.4 Scenario 4

Mark likes to go see runs very much but he's very busy and sometimes his schedule overlaps with the races. He opens Track4Run and starts by searching for the right run to follow. Then, after selecting it he sees every athletes' positions on the map. He wants to only see the position of his favourite runner, so he inserts inside the search bar the name of the athlete. Now he can see the positions and stats of the race all while comfortably completing his other errands.

3.2.5 Scenario 5

Sarah is organizing a new race in her town. She wants to give the possibility to people to follow the run in real-time. She opens Track4Run and starts to create a new race. The application asks her some information like the name of the run, the date, the city, the starting time and the maximum number of participants. At the end Track4Run allows her to add the precise path of the run. Days later during the race lots of spectators are able to follow the entire run she organized just by looking at an online map inside the application and without having to be near the path of the competition.

3.3 Functional Requirements

3.3.1 Goals, requirements and domain assumptions

[G1]. Third parties can request access to data of some specific individuals

- [D2] Third parties know a way to identify the individuals whose data are of interest
- [D4] Users have a unique identification code
- [R1] Individual can activate the monitor function inside the application
- [R2] Users can register to the application

[G2]. Third parties are allowed to request access to anonymized data of groups of individuals

- [R1] Individual can activate the monitor function inside the application
- [R2] Users can register to the application

[G3]. Third parties can access the data of specific individuals

- [D1] By activating the monitoring feature inside the application, the individuals agree that TrackMe acquires their data
- [D5] Device sensors can provide an accurate enough current location
- [D7] Individuals don't take the device sensor off
- [D8] Individuals uses the device sensor correctly
- [D9] Device sensors work correctly and without malfunctions
- [R1] Individual can activate the monitor function inside the application
- [R3] Track me passes request of third parties to the specific individuals
- [R4] Individuals are allowed to accept the request from the third parties
- [R5] Third parties can download data of individuals/groups from the application

[G4]. Third parties are able to access anonymized data of groups of individuals

- [D1] By activating the monitoring feature inside the application, the individuals agree that TrackMe acquires their data
- [D5] Device sensors can provide an accurate enough current location
- [D7] Individuals don't take the device sensor off
- [D8] Individuals uses the device sensor correctly
- [D9] Device sensors work correctly and without malfunctions
- [R5] Third parties can download data of individuals/groups from the application
- [R6] TrackMe accept any request for which the number of individuals whose data satisfy the request is higher than 1000

[G5]. Individuals are given the choice to accept or refuse third parties' requests to access their data

- [D1] By activating the monitoring feature inside the application, the individuals agree that TrackMe acquires their data
- [D2] Third parties know a way to identify the individuals whose data are of interest
- [D4] Users have a unique identification code
- [R1] Individual can activate the monitor function inside the application
- [R3] Track me passes request of third parties to the specific individuals
- [R4] Individuals are allowed to accept the request from the third parties

[G6]. To subscribed users is guaranteed anonymity

- [D3] When the number of individuals whose data satisfy a request is less than 1000, third parties are able to identify the individuals
- [R6] TrackMe accept any request for which the number of individuals whose data satisfy the request is higher than 1000

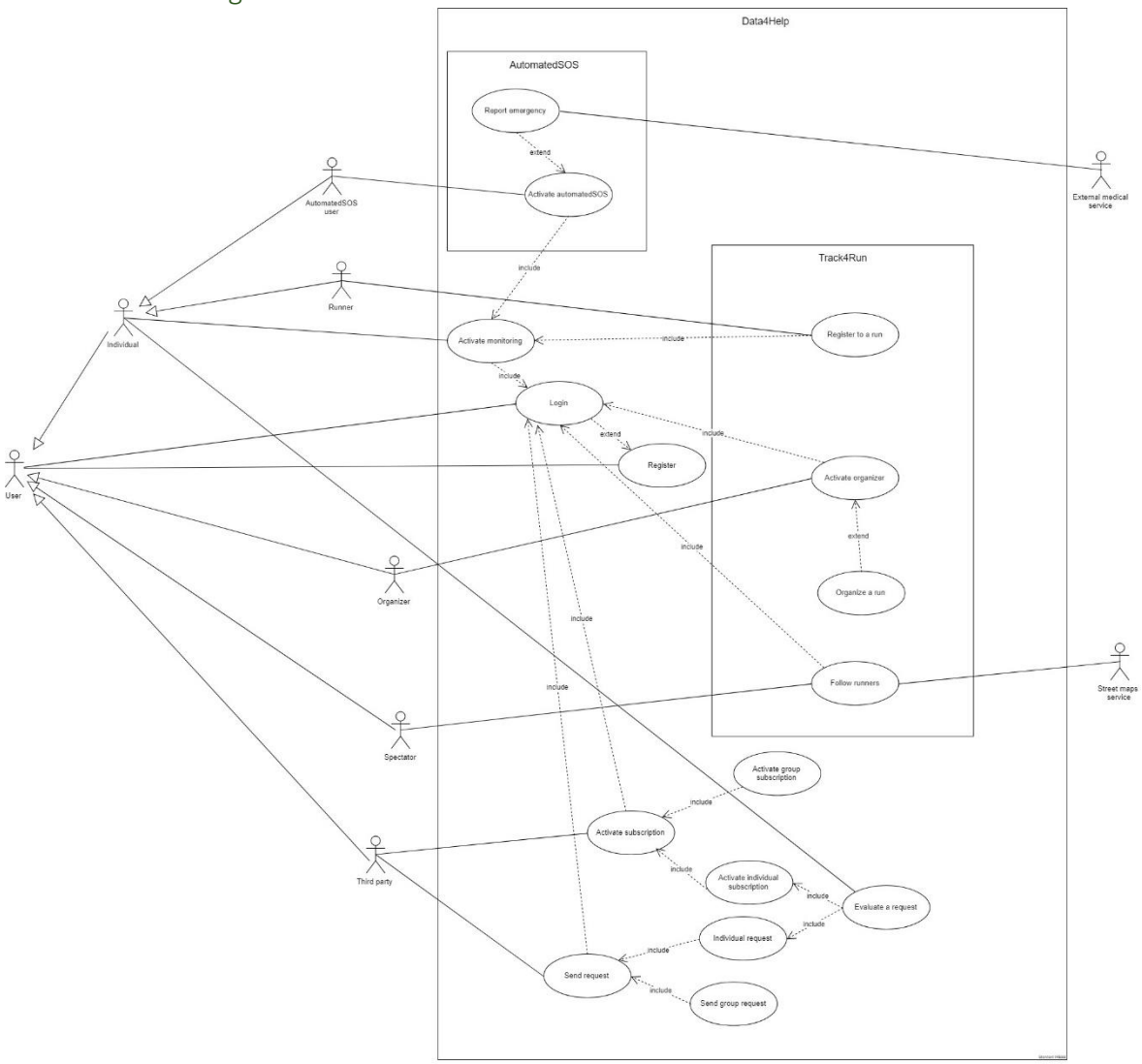
[G7]. Subscribed costumers receive immediate help when they are displaying serious medical conditions

- [D6] When the parameters go below a certain threshold AutomatedSOSUsers' health needs immediate medical help
- [D11] External medical services are able to receive automatic emergency reports
- [D12] Runners have a tracking device on their person during the run
- [D13] Organizers decide a path for the run
- [D14] Individuals enroll to the run
- [R7] AutomatedSOS analyses users acquired data
- [R8] AutomatedSOS sends requests to external medical service for an ambulance when the data go below the threshold

[G8]. Spectators are able to know the position of the athletes on the path of the run

- [D12] Runners have a tracking device on their person during the run
- [D13] Organizers decide a path for the run
- [D14] Individuals enroll to the run
- [R9] Track4Run allows organizers to load path for the run
- [R10] Track4Run allows individuals to sign up to a run as participants
- [R11] Track4Run shows the position of the runners on the map

3.3.2 Use case diagram



3.3.3 Use cases

Name	UC1 - Register
Participant actors	User
Entry conditions	The user has installed the application or he's trying to access Data4Help's web services for the first time.
Events flow	1. User enters in registration section 2. User fills all necessary fields to accomplish the registration 3. User confirms registration 4. Data4Help systems save new user's account data
Exit condition	User has successfully registered to Data4Help and now he's allowed to login
Exceptions	1. User is already registered 2. User doesn't fill all necessary fields to accomplish the registration 3. Provided username isn't available 4. Provided e-mail does not exist

Name	UC2 - Login
Participant actors	User

Entry conditions	User is registered to Data4Help
Events flow	1. User access to the application 2. User enters his username and password 3. User confirms the login
Exit condition	User is successfully logged in, and now he's allowed to use/active its services
Exceptions	1. User enters an inexistent username 2. User enters an inexistent or a wrong password

Name	UC3 - Send individual request
Participant actors	Third party
Entry conditions	User has successfully logged in
Events flow	1. User opens the section dedicated to single requests 2. User access to the section dedicated to individual requests 3. User enters a code (like SSN) to identify the individual whose data are requested 4. User's application sends the request to Data4Help system 5. The request is forwarded to the interested individual
Exit condition	The request has been forwarded to the interested individual and the application is waiting for the individual to evaluate it
Exceptions	1. The entered identifying code is inexistent or invalid 2. The request can't be forwarded to the user because he has deleted his registration to the service

Name	UC4 - Send group request
Participant actors	Third party
Entry conditions	User has successfully logged in
Events flow	1. User opens the section dedicated to group request 2. User access to the section dedicated to group requests 3. User enters parameters and conditions that the request must fulfil 4. User's application sends the request to Data4Help system 5. The request is evaluated by the system (in order to grant individuals' anonymity), then it is approved or rejected 6. User receives an answer to his request
Exit condition	User has received an answer containing the requested data or a warning that reports the rejection of his request
Exceptions	1. Parameters and conditions entered by the user aren't valid 2. Parameters and conditions entered by the user refers to an empty set of individuals

Name	UC5 - Activate individual subscription
Participant actors	Third party
Entry conditions	User has successfully logged in
Events flow	1. User opens the section dedicated to single requests 2. User access to the section dedicated to individual requests 3. User flags the mark for the subscription 4. User enters a code (like SSN) to identify the individual whose data are requested 5. User's application sends the request to Data4Help system 6. The request is forwarded to the interested individual 7. Individual's data are periodically sent to the user who made the subscription if the request is approved

Exit condition	The requested data are periodically sent to the third party who has made the subscription, or the request has been rejected
Exceptions	1. The entered identifying code is inexistent or invalid 2. The request can't be forwarded to the user because he has deleted his registration to the service

Name	UC6 - Activate group subscription
Participant actors	Third party
Entry conditions	User has successfully logged in
Events flow	1. User opens the section dedicated to group request 2. User access to the section dedicated to group requests 3. User flags the mark for the subscription 4. User enters parameters and conditions that the request must fulfil 5. User's application periodically sends the request to Data4Help system 6. The request is evaluated by the system (in order to grant individuals' anonymity), then it is approved or rejected 7. User periodically receives an answer to his request
Exit condition	User periodically receives an answer containing the requested data or a warning that reports the rejection of his request
Exceptions	1. Parameters and conditions entered by the user aren't valid 2. Parameters and conditions entered by the user refers to an empty set of individuals

Name	UC7 - Evaluate a request
Participant actors	Individual, third party
Entry conditions	A request to access individual's health status data is been sent
Events flow	1. Individual receives the request of access to his data from a third party 2. Individual approves or rejects the request 3. An answer is sent to the third party who made the request
Exit condition	Third party has received an answer containing the requested data or a warning that reports the rejection of his request
Exceptions	1. Individual doesn't evaluate the request

Name	UC8 - Activate monitoring
Participants actors	Individual
Entry conditions	The individual has already logged in
Events flow	1. The individual selects the "Data4Help" option on the menu 2. The individual selects the "Activate the monitoring" option 3. The individual inserts his/her CF or SSN 4. The individual connects his/her device 5. The individual clicks on the "Activate" button
Exit conditions	The individual is now correctly monitored
Exceptions	1. The individual's CF or SSN isn't valid

Name	UC9 - Activate AutomatedSOS
Participants actors	AutomatedSOS user

Entry conditions	The AutomatedSOS user is already monitored
Events flow	1. The AutomatedSOS user selects the “AutomatedSOS” option in the menu 2. The AutomatedSOS user selects the “Subscribe to the AutomatedSOS service” option 3. The AutomatedSOS user inserts his/her age 4. The AutomatedSOS user clicks on the “Subscribe” button
Exit conditions	The AutomatedSOS user is correctly subscribed to the AutomatedSOS service
Exceptions	1. The inserted age is below 65 or is invalid

Name	UC10 - Report emergency
Participants actors	AutomatedSOS user, External medical service
Entry conditions	The AutomatedSOS user has already activated the AutomatedSOS service
Events flow	1. The AutomatedSOS user’s data go below the threshold 2. The application immediately contacts an external medical service
Exit conditions	The application has correctly contacted an external medical service
Exceptions	\
Special requirements	The external medical service is successfully contacted in 5 second

Name	UC11 - Activate organizer
Participants actors	Organizer
Entry conditions	The organizer has already logged in
Events flow	1. The organizer selects the “Track4Run” option in the menu 2. The organizer selects the “Subscribe as an organizer” option 3. The organizer inserts his/her organizer’s data 4. The organizer clicks on the “Subscribe” button
Exit conditions	The organizer is subscribed also as an organizer
Exceptions	1. The organizer’s data aren’t correct

Name	UC12 - Organize a run
Participants actors	Organizer
Entry conditions	The organizer has already activated the organizer service
Events flow	1. The organizer selects the “Track4Run” option in the menu 2. The organizer selects the “Organize a run” option 3. The organizer inserts the city 4. The organizer inserts the date 5. The organizer inserts the max number of participants 6. The organizer inserts the start point and the end point of the path 7. The organizer clicks the “Create” button
Exit conditions	The organizer has successfully created a run
Exceptions	1. The organizer did not enter a valid data

Name	UC13 - Follow runners
Participants actors	Spectator, Street maps service

Entry conditions	The spectator has already logged in
Events flow	1. The spectator selects the “Track4Run” option in the menu 2. The spectator selects the “Watch a run” option 3. The spectator inserts the name of the run he/she wants to watch 4. The spectator clicks the “Ok” button
Exit conditions	The spectator is correctly watching the run on the map
Exceptions	1. The name of the run inserted by the spectator doesn’t exist

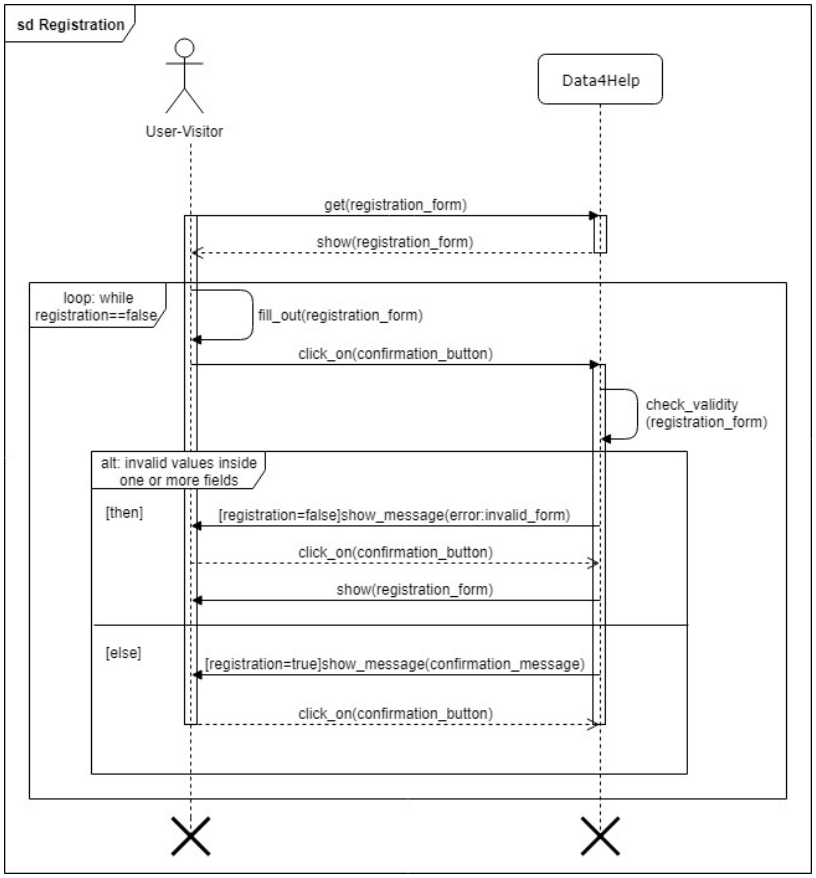
Name	UC14 - Register to a run
Participants actors	Runner
Entry conditions	The runner is already monitored
Events flow	1. The runner selects the “Track4Run” option in the menu 2. The runner selects the “Register to a run option” 3. The runner inserts the name of the run he/she wants to register to 4. The runner clicks the “Register” button
Exit conditions	The runner is correctly registered to the run
Exceptions	1. The name of the run inserted by the runner doesn’t exist 2. The runner is already subscribed to the selected run

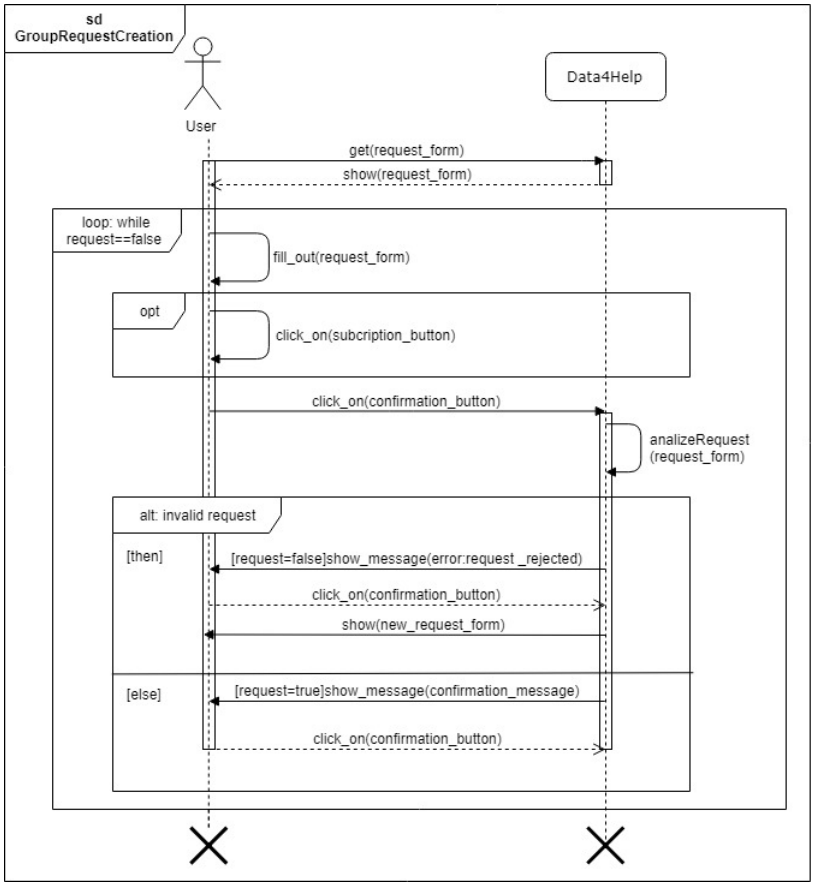
3.3.4 Traceability Matrix

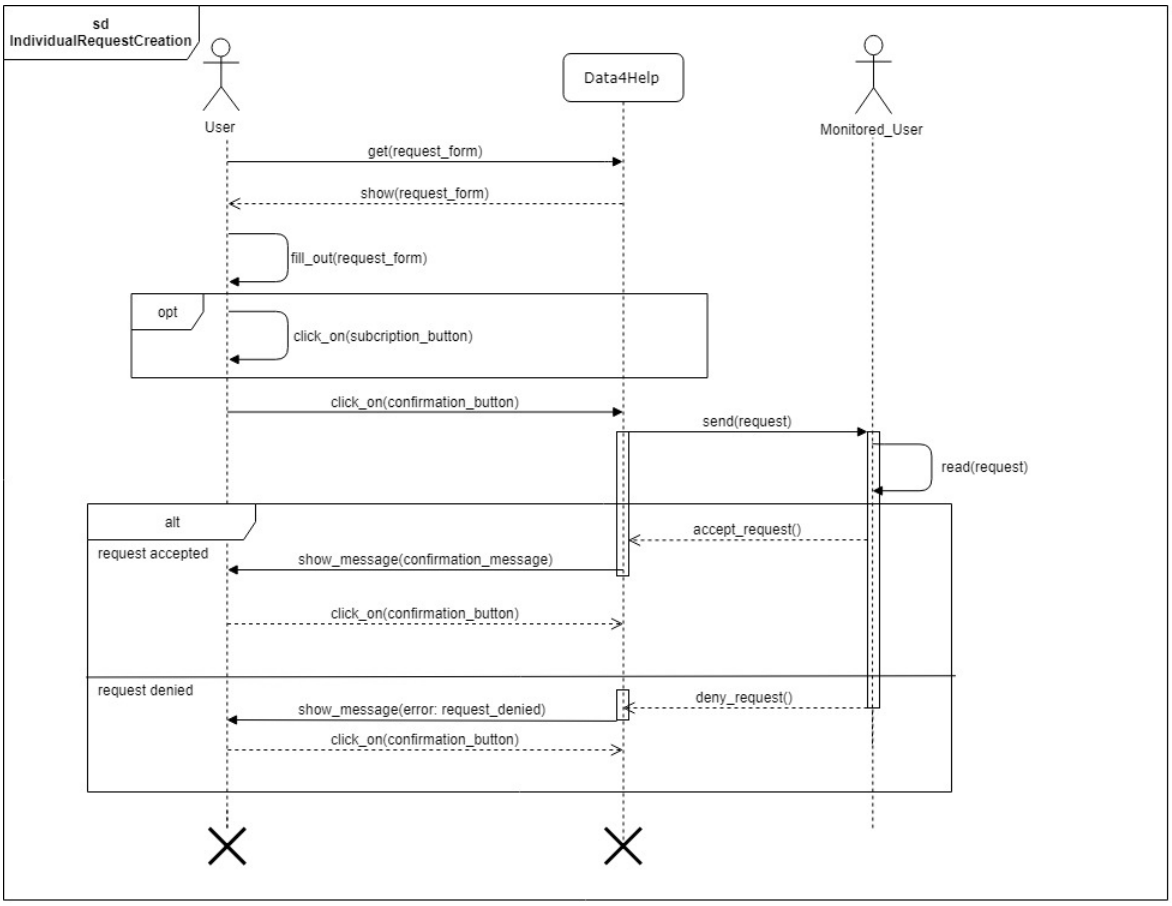
The following table shows the links among goals, requirements and use cases of whole application.

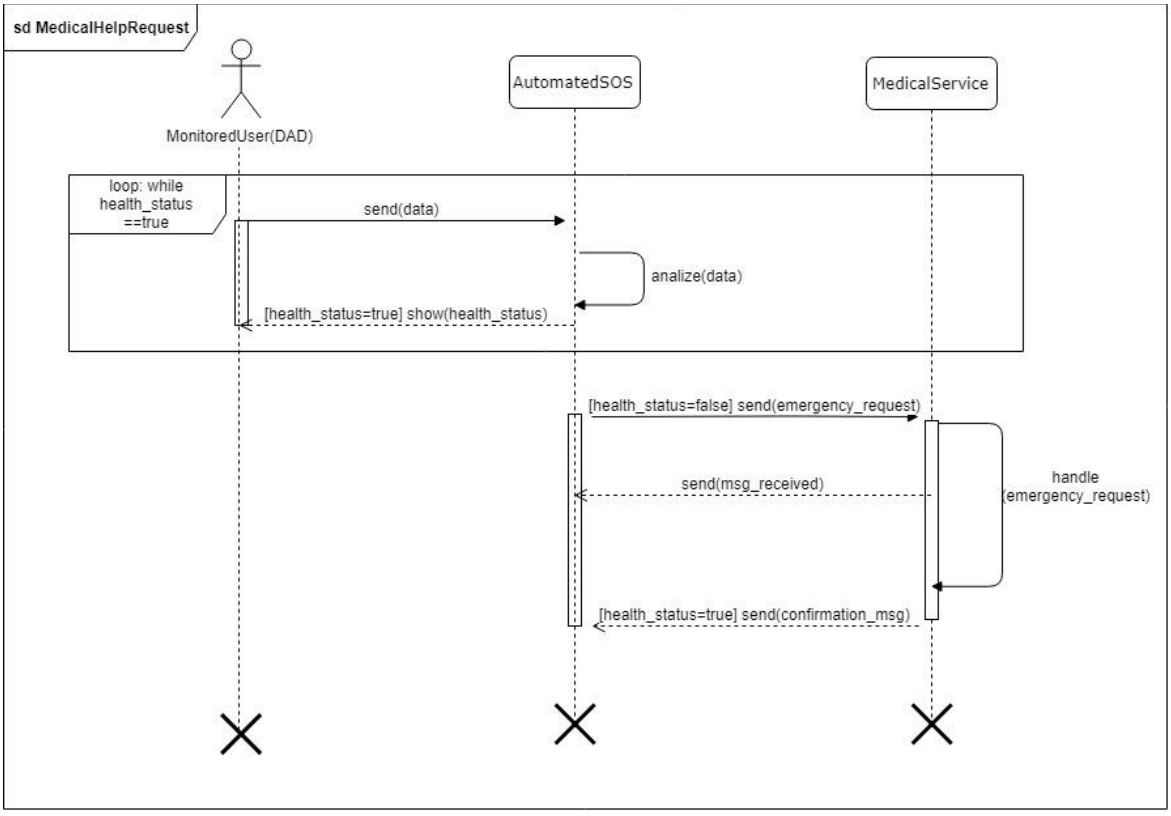
Goals	Requirements	Use cases
G1	R1	UC8 - Activate monitoring
	R2	UC1 - Register, UC2 - Login
G2	R1	UC8 - Activate monitoring
	R2	UC1 - Register, UC2 - Login
G3	R1	UC8 - Activate monitoring
	R3	UC3 - Send individual request
	R4	UC7 - Evaluate a request
	R5	UC3 - Send individual request, UC4 - Send group request, UC5 - Activate individual subscription, UC6 - Activate group subscription
G4	R5	UC3 - Send individual request, UC4 - Send group request, UC5 - Activate individual subscription, UC6 - Activate group subscription
	R6	UC4 - Send group request, UC6 - Activate group subscription
G5	R1	UC8 - Activate monitoring
	R3	UC3 - Send individual request
	R4	UC7 - Evaluate a request
G6	R6	UC4 - Send group request, UC6 - Activate group subscription
G7	R7	UC9 - Activate AutomatedSOS
	R8	UC10 - Report Emergency
G8	R9	UC11 - Activate organizer, UC12 - Organize a Run
	R10	UC14 - Register to a run
	R11	UC13 - Follow runners

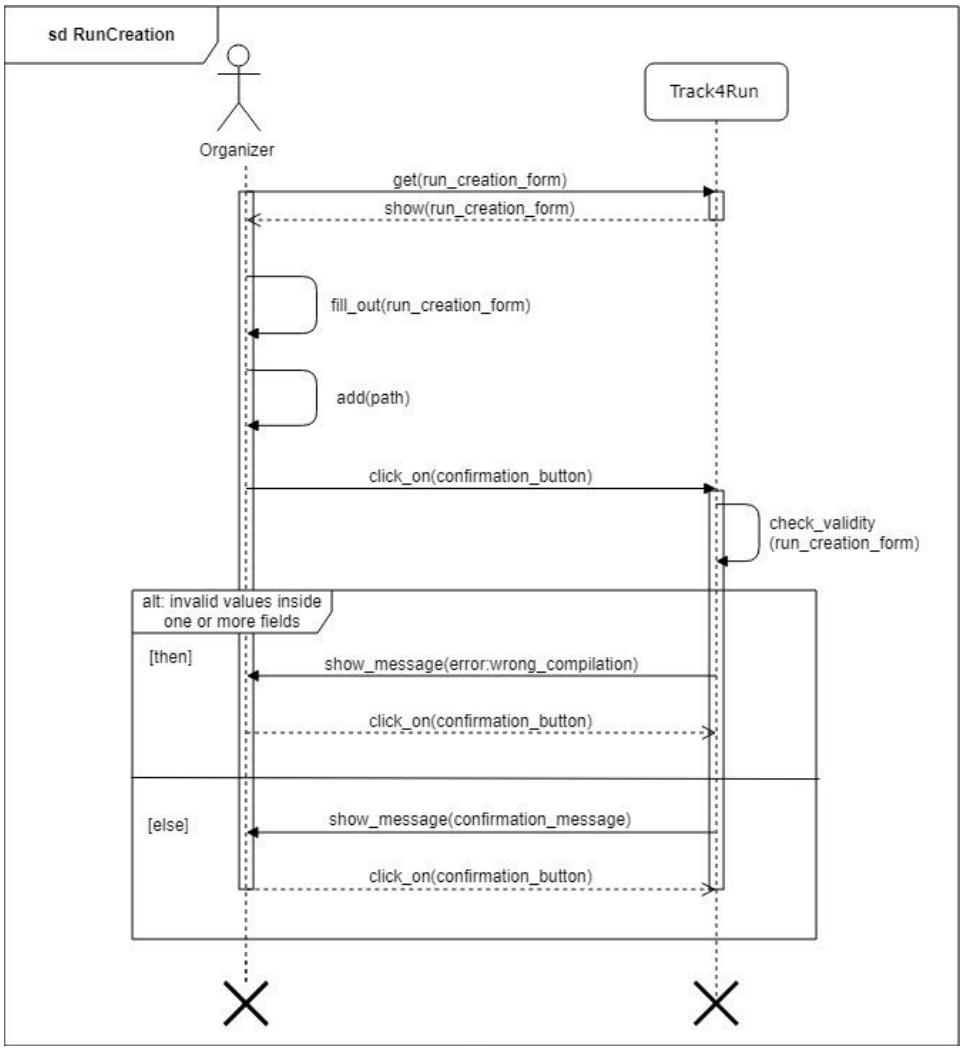
3.3.5 Sequence Diagrams

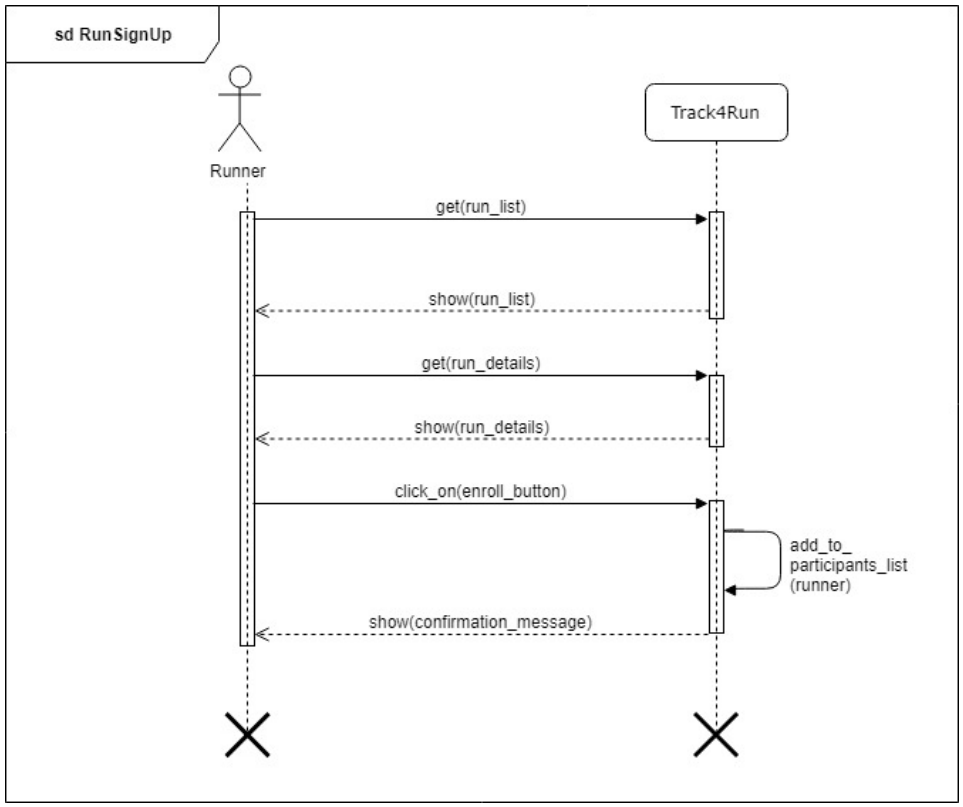





sd
IndividualRequestCreation
User
Data4Help
Monitored_User
get(request_form)
show(request_form)
fill_out(request_form)
opt
click_on(subcription_button)
click_on(confirmation_button)
send(request)
read(request)
alt
accept_request()
request accepted
show_message(confirmation_message)
click_on(confirmation_button)
request denied
deny_request()
show_message(error: request_denied)
click_on(confirmation_button)


sd MedicalHelpRequest
MonitoredUser(DAD)
AutomatedSOS
MedicalService
loop: while
health_status
==true
send(data)
analize(data)
[health_status=true] show(health_status)
[health_status=false] send(emergency_request)
send(msg_received)
handle
(emergency_request)
[health_status=true] send(confirmation_msg)


sd RunCreation
Organizer
Track4Run
get(run_creation_form)
show(run_creation_form)
fill_out(run_creation_form)
add(path)
click_on(confirmation_button)
check_validity
(run_creation_form)
alt: invalid values inside
one or more fields
[then]
show_message(error:wrong_compilation)
click_on(confirmation_button)
[else]
show_message(confirmation_message)
click_on(confirmation_button)


sd RunSignUp
Runner
Track4Run
get(run_list)
show(run_list)
get(run_details)
show(run_details)
click_on(enroll_button)
add_to_
participants_list
(runner)
show(confirmation_message)

3.4 Performance Requirements

The application should be able to respond to lot of user request at the same time and to constantly monitor a large number of subscribers (at the beginning it should be able to handle around 25000/30000 consumers). The users, in particular regarding AutomatedSOS, will rely on a constant monitoring done by the application so the system will have to guarantee a good reliability.

3.5 Design Constraints

3.5.1 Standards Compliance

- The application can be used either with landscape or portrait orientations. In the latter case the visualization of the content may change slightly.
- The application works correctly in an SD card, given that the space available is enough.
- The application saves the current state when is working in background or when the smartphone is put on standby and restore all the saved states and data when is called back to the foreground.
- The application saves private data in an internal database and that information won't be used for any commercial uses or distributed to third parties without user's confirmation.

3.5.2 Hardware Limitations

The main hardware limitations are:

- A functioning smartphone (iOS, Android)
- A working internet connection(2G/3G/4G)
- Bluetooth (only if the monitoring is activated)
- Memory space for the application

3.5.3 Other Constraints

The presence of a tracking device is needed for the application to properly monitor subscribed users. The device should be correctly handled and set up by the user. It should also be guaranteed a complete functionality by the company who produced it.

3.6 Software system attributes

3.6.1 Reliability

Reliability of TrackMe's systems depends on the type of service offered. AutomatedSOS service must rely upon a strong reliability, it has to be available 24/7 since it could be crucial to save a user's life. Data4Help and Track4Run services don't offer a vital service like AutomatedSOS's one, so they are offered with a less strong reliability, especially during late night (except for Data4Help's monitoring of individuals).

3.6.2 Availability

As for reliability, also availability of AutomatedSOS's service must be higher than the other services, it is required an availability of 5-nines to guarantee constant assistance to elderly people. Availability of Data4Help is important to be sufficient to correctly monitor individuals, so a 3-nines availability it is required. To offer a satisfying service toTrack4Run users it's enough an availability of 2-nines.

3.6.3 Security

User's passwords and personal data monitored by the system must be stored and encrypted. Also, the transfer of this data from user to TrackMe's system must be done in a secure way.

3.6.4 Maintainability

The application will be designed to be flexible and new features will be easy to add to the existing system since the service of Data4Help could be used as a base or a support for other services (as already happened for AutomatedSOS or Track4Run).

3.6.5 Portability

TrackMe's application is compatible with a big variety of mobile devices, especially many DADs must be able to run the application in order to offer a big amount of data to interested third parties.

4. Formal Analysis Using Alloy

In this chapter we described some of the features that characterises the system using an Alloy model. The model describes the following aspects:

- The creation of individual requests from third parties.
- Individual and group request acceptance conditions.
- Request evaluation from individuals.
- When third parties are allowed to access to individual's data.
- Emergency report conditions for AutomatedSOS users.
- Track4Run organization.

```
open util/boolean
open util/integer


sig Individual {
   id: one Int,
   currentLocation: lone Location,
   currentHealthStatus : lone HealthStatus
}
{id >= 0}

sig Location{
   user : one Individual
}

sig HealthStatus{
   currentHealthCondition : one Int,
   healthThreshold : lone Int,
   user : one Individual
}
{currentHealthCondition >= 0 and healthThreshold >= 0}

sig AutomatedSOSUser extends Individual{
   normalHealthCondition : one Bool,
   emergencyMessage : lone EmergencyMessage
}

sig EmergencyMessage{
   user : one AutomatedSOSUser,
}
```

```
sig Runner extends Individual {
   races : some Run
}

sig Run{
   participants : set Runner,
   organizer : one Organizer,
   dateTime : one DateTime,
   path : one Path
}

sig DateTime{}

sig Path{
   races : some Run
}

sig Organizer{
   races : set Run
}

sig Spectator{
   races : set Run
}

sig ThirdParty{
   request : set Request,
   subscription : set Subscription
}

abstract sig Request{
   idRequest : one Int,
   requester : one ThirdParty,
   approved : one Bool
}
{idRequest >= 0}

sig GroupRequest extends Request{
   userCardinality : one Int,
}
{userCardinality >= 0 }

sig IndividualRequest extends Request{
   individual : one Individual,
   individualConfirmation : one Bool
}

sig Subscription{
   idSubscription : one Int,
   idRequest : one Int,
   subscriber : one ThirdParty
}
{idRequest >= 0 and idSubscription>=0}



fact PathRun{
   no disj p1,p2 : Path | some r : Run | r in p1.races and r in p2.races
}
```

```
fact HealthStatusIndividualAssociation{
   all i : Individual | all h : HealthStatus | ( (h in i.currentHealthStatus) implies (h.user = i)) and (h.user = i implies h in i.currentHealthStatus)
}

fact  LocationIndividualAssociation{
   all i : Individual | all l : Location | ( l in i.currentLocation implies l.user = i) and (l.user = i implies l in i.currentLocation)
}

fact RunnerRunAssociation{
   all r1 : Runner | all r2 : Run | (r2 in r1.races implies r1 in r2.participants) and ( r1 in r2.participants implies r2 in r1.races)
}

fact OrganizerRunAssociation{
   all o : Organizer | all r : Run | (r in o.races implies r.organizer = o) and (r.organizer = o  implies r in o.races)
}

fact RequestThirdPartyAssociation{
   all r : Request | all t : ThirdParty | (r.requester = t implies r in t.request) and ( r in t.request implies r.requester = t)
}

fact SubscriptionThirdPartyAssociation{
   all s : Subscription | all t : ThirdParty | (s.subscriber = t implies s in t.subscription) and ( s in t.subscription implies s.subscriber = t)
}

fact IdCardinality{
   (no disj r1,r2 : Request | r1.idRequest = r2.idRequest) and (no disj s1,s2 : Subscription | s1.idSubscription = s2.idSubscription) and (no disj i1,i2 :Int | i1=i2)
}

-- A third party is considered as such only if it has done at least one request or one subscription
fact ThirdPartySubscriptionRequestCardinality{
   all t : ThirdParty | (some s: Subscription | s in t.subscription and s.subscriber=t ) or (some r : Request | r in t.request and r.requester=t)
}


fact WhenIsAnEmergencyMessageSent{
   no h : HealthStatus | (h.currentHealthCondition > h.healthThreshold and h.user.normalHealthCondition=False and h.user.emergencyMessage=none)
   and (h.currentHealthCondition <= h.healthThreshold and h.user.normalHealthCondition=True and #h.user.emergencyMessage=1)

}

--AutomatedSOSUser always have a current location and a current health status saved
fact AutomatedSOSUserCardinality{
   no a : AutomatedSOSUser | a.currentLocation = none or a.currentHealthStatus = none
}

fact RequestEvaluation{
   (all g : GroupRequest | (acceptGroupRequest[g] implies g.approved=True) and (not acceptGroupRequest[g] implies g.approved=False))
   and
   (all i : IndividualRequest | (acceptIndividualRequest[i] implies i.approved=True) and (not acceptIndividualRequest[i] implies i.approved=False))
}



pred acceptGroupRequest[g : GroupRequest] {
   g.userCardinality>6
}

pred acceptIndividualRequest[i : IndividualRequest]{
     i.individualConfirmation = True
}

pred whenThirdPartyCanDownloadData[t : ThirdParty, r : Request]{
   r in t.request and r.approved = True
}
```

```
-- ir is the created individual request, t is the third party who will create the request ir, but they hasn't still created it
-- t' is the third party after he has created the request ir
pred createIndividualRequest[ t, t' : ThirdParty, i : Individual, ir : IndividualRequest,  id : Int, a : Bool, ic : Bool]{
   --precondition
   ir not in t.request
   --body
   ir.requester = t'
   ir.individual = i
   t'.request = t.request + ir
   ir.idRequest = id
   ir.approved = a
   ir.individualConfirmation = ic
   --postcondition
   ir in t'.request

}

pred sendEmergencyMessage[a : AutomatedSOSUser]{
   a.currentHealthStatus.currentHealthCondition > a.currentHealthStatus.healthThreshold
}

pred show {
   (some t : ThirdParty | #(t.request) = 2) and
   (one a : AutomatedSOSUser | #(a.emergencyMessage) = 1) and
   (one r : Run | #(r.participants) = 1)
}




run acceptGroupRequest for 2
run acceptIndividualRequest for 3 but exactly 2 Individual
run whenThirdPartyCanDownloadData for 2
run createIndividualRequest for 4
run sendEmergencyMessage for 2
run show for 2 but exactly 2 Spectator
```

4.1 Alloy Results

Executing "Run acceptGroupRequest for 2"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
4406 vars. 344 primary vars. 10396 clauses. 83ms.
Instance found. Predicate is consistent. 47ms.

Executing "Run acceptIndividualRequest for 3 but exactly 2 Individual"
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
6672 vars. 523 primary vars. 15236 clauses. 172ms.
Instance found. Predicate is consistent. 104ms.

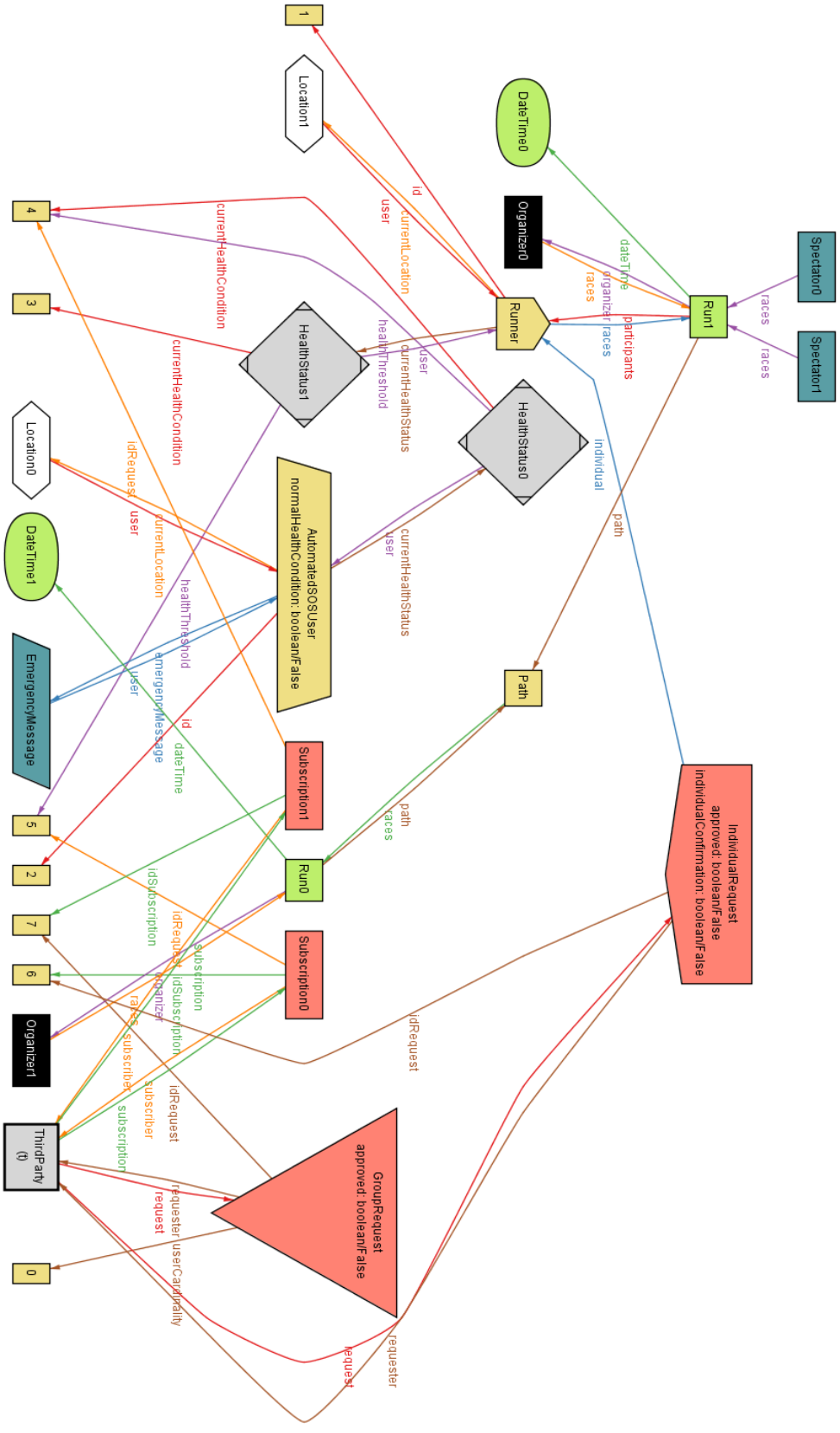
Executing "Run whenThirdPartyCanDownloadData for 2"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
4247 vars. 346 primary vars. 9864 clauses. 90ms.
Instance found. Predicate is consistent. 105ms.

Executing "Run createIndividualRequest for 4"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
11038 vars. 872 primary vars. 24863 clauses. 198ms.
Instance found. Predicate is consistent. 123ms.

Executing "Run sendEmergencyMessage for 2"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
4605 vars. 344 primary vars. 11015 clauses. 90ms.
Instance found. Predicate is consistent. 77ms.

Executing "Run show for 2 but exactly 2 Spectator"
Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
4238 vars. 342 primary vars. 9897 clauses. 83ms.
Instance found. Predicate is consistent. 71ms.

4.2 World Generated



5.Effort Spent

5.1

Irene Nizzoli	
Purpose, Scope	4
Definition, Document Structure	1,5
Product Perspective and Functions	2
Domain Assumptions	4,5
External Interface Requirements	1
Scenarios, User Characteristics	2,5
Functional Requirements	5
Performance Requirements	3,5
Design Constraints, Software Attributes	2
Alloy	6

5.2

Isabella Piacentini	
Purpose, Scope	3
Definition, Document Structure	1
Product Perspective and Functions	1,5
Domain Assumptions	4
External Interface Requirements	1
Scenarios, User Characteristics	4,5
Functional Requirements	6
Performance Requirements	3,5
Design Constraints, Software Attributes	2,5
Alloy	5

5.3

Elio Salvini	
Purpose, Scope	1,5
Definition, Document Structure	3,5
Product Perspective and Functions	3
Domain Assumptions	3,5
External Interface Requirements	3
Scenarios, User Characteristics	1
Functional Requirements	6
Performance Requirements	3
Design Constraints, Software Attributes	2
Alloy	5,5

6.References

- Specification document “Mandatory Project Assignment AY 2018-2019”
- ISO/IEC/IEEE 29148 - Standard on requirement engineering
- Slides – “Requirements engineering” Part I and Part II, “Alloy”, “Use of Alloy in RE”