Politecnico di Milano
AA 2018-2019

Computer Science and Engineering
Software engineering 2

Data4Help RASD

Requirements Analysis and Specification Document
Version 2.0
09/12/18

Diego Piccinotti, Umberto Pietroni, Loris Rossi

Table of Contents

1 Introduction

1.1 Purpose

The purpose of this document is to analyze and illustrate specifications for the *Data4Help* system. This will be accomplished through a detailed analysis of the solution proposed and of the goals, assumptions and requirements that are necessary to realize it.
This document is intended to be used by the clients requiring the development of the system, its end users and all those who are involved in the development process, mainly project managers, analysts and development teams.

Data4Help is designed as a platform to enable third parties to access its users' health and location data. The system continuously collects these data through wearable devices and stores them to enable later access to third parties.
Third parties can request data of some specific individual, whose collection has to be approved by the user (which grants access to their data from this moment on), or data of a group of users which must be larger than 1000 individuals to allow for proper data anonymization.
Data4Help is then extended in its functionality by ***AutomatedSOS***, an additional service built on top of the existing data collection platform, which monitors subscribed elderly users' health parameters and automatically requests an ambulance to their location if their parameters get below a risk threshold determined by a preventive medical checkup.
Finally, *Data4Help* platform can be used also to enable an easy management of run competitions from organizers and to enable athletes to enroll to them through another additional service, ***Track4Run***, capable of broadcasting the live coverage to spectators through a map showing the live position of the runners.

1.2 Scope

In this section we describe the boundaries between the environment and the system to be.
In particular, we distinguish between world, shared and machine phenomena.
Moreover, we define the goals of the system in order to satisfy the stakeholders' needs.

1.2.1 World, shared and machine phenomena

World phenomena are the ones which the machine cannot observe.
In our context, world phenomena are, for instance:

- Movement of a user from a position to another one.
- A change of a user's health status.
- The dispatch of an ambulance from an external provider.

Phenomena entirely occurring in the machine are considered machine phenomena.
For instance, data manipulation from a database or checking for proper anonymization of a group of data shall be done by the system to be.

Regarding shared phenomena, which involve both the environment and the system, we have events like:

- A third party requests a specific user's data.
- A user signs up in the system.
- The system requires the external provider to dispatch an ambulance.

Further shared phenomena are discussed in the Product Perspective paragraph (2.1).

1.2.2 Goals

G1: The user can be recognized by providing a form of identification
G2: Allow third parties to monitor data about location and health status of individuals.
G3: Allow third parties to access data relative to specific users
G4: Allow third parties to access anonymized data of groups of users
G5: Allow third parties to offer a personalized and non-intrusive SOS service to elderly people so that an ambulance arrives to the location of the customer in case of emergency.
G6: Allow athletes to enroll in a run
G7: Allow organizers to manage runs
G8: Allow spectators to see on a map the position of all runners during the run

1.3 Definitions, acronyms, abbreviations

1.3.1 Definitions

- **User**: individual who allows *Data4Help* to monitor his location and health status.
- **Third party**: individual or organization registered to *Data4Help* which can request users' data.
- **Data collection**: gathering of users' data through a wearable device.
- **Anonymized data**: data about more than 1000 users whose personal information has been previously removed so that they are not directly relatable to the system's users.
- **Elderly**: user who is subscribed to *AutomatedSOS* and is older than 60 years old.
- **Risk threshold**: Set of boundary health parameters defined for each elderly. If monitored values of the user's health parameters get below these boundaries, an SOS request is placed to an external ambulance provider.
- **Athlete**: user who participates in a run.
- **Run organizer**: third party who can manage runs for athletes.
- **Spectator**: non-registered individual who follows a run through a map with runners' positions.
- **Wearable**: a personal device provided with biometric sensors and GPS given to each user for free after the registration process.

1.3.2 Acronyms

- **API**: Application Programming Interface
- **GPS**: Global Positioning System
- **GSR**: Galvanic Skin Response

1.3.3 Abbreviations

- **Gn**: n^{th} goal.
- **Dn**: n^{th} domain assumption.
- **Rn**: n^{th} functional requirement.
- **Rn-NF**: n^{th} nonfunctional requirement.
- **UCn**: n^{th} use case.

1.3.4 Conventions

We'll use "their" and "them" to indicate "his or her" in those cases when it is useful to be gender neutral in writing.

1.4 Revision History

Version 1: committed on 11/11/18. First version.
Version 2: updated on 9/12/18. Fixed RequestManager state diagram, added R31 to G3, created new requirement R32, fixed use case 3.

1.5 Reference Documents

- Specifications document: "Mandatory Project Assignment AY 2018-2019"
- IEEE Standard on Requirement Engineering (*ISO/IEC/IEEE 29148, Dec 2011*)
- Alloy reference - http://alloy.lcs.mit.edu/alloy/documentation/book-chapters/alloy-language-reference.pdf
- UML reference - https://www.uml-diagrams.org

1.6 Document Structure

Chapter 1 introduces the problem and describes the purpose of the application *Data4Help.* The scope of the application is defined by stating the goals and a brief description of phenomena.

Chapter 2 presents the overall description of the project. Product perspective includes details on the boundaries of the system and world, machine and shared phenomena. In Product Functions are described the main functions of the system to be and User Characteristics highlights the main actors. At last are defined the domain assumption on which the system relies on.

Chapter 3 contains all the information about the system requirements needed to satisfy each goal. Furthermore, in this section are highlighted the major functions and interactions between the actors and the system using use cases and sequence diagrams.

Chapter 4 includes the alloy model and the discussion of its purpose.

Chapter 5 shows the effort spent by each group member while working on this project.

2 Overall Description

2.1 Product perspective

2.1.1 Phenomena

World: (*world phenomena the machine cannot observe)*

- Users move (their position changes).
- Users' health parameters vary.
- The external provider dispatches an ambulance.

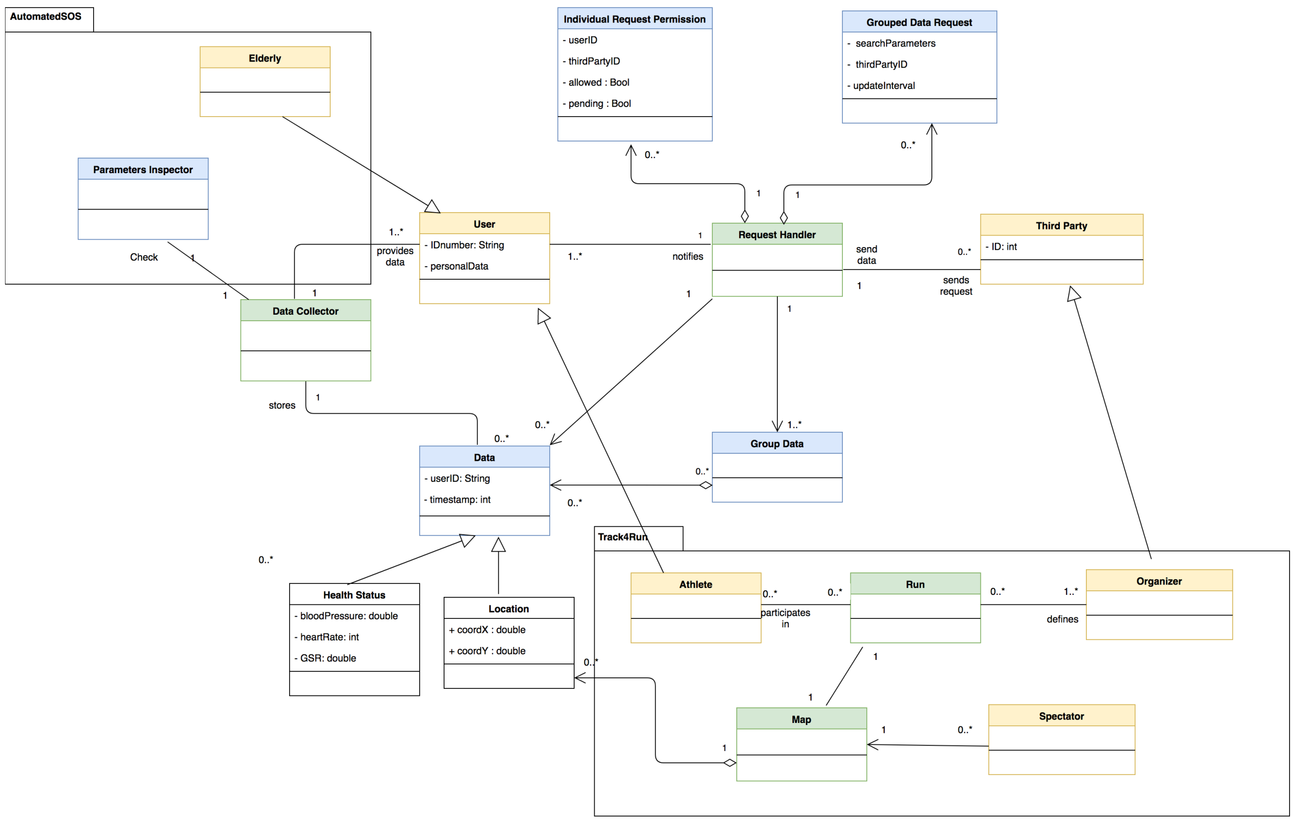
Shared: (*controlled by the world and observed by the machine or controlled by the machine and observed by the world)*

- User and third parties can sign up and login to the system.
- User data collection by the system: the data is collected from the sensor on the user's wearable, thus the phenomenon being shared.
- Data requests from third parties: the system acts as an interface between user data and third parties.
- System sends data to the third parties.
- Data collection acceptance/denial by the user: again, the system acts as an interface between two world entities and knows what is happening.
- The system asks for an ambulance dispatch to the external provider: the system produces an effect on the world, the dispatching.
- Position update on map for runners.
- Organizers input runs data to the system.
- Athletes enroll to runs available in the system: this phenomenon is shared because we assume that enrollment to runs happens only through the system and is not a copy of a physical form.
- Spectators connect to the system to watch runs: spectators are able to follow the live coverage through the system that is displaying a map.

Machine: (*phenomena located entirely in the machine)*

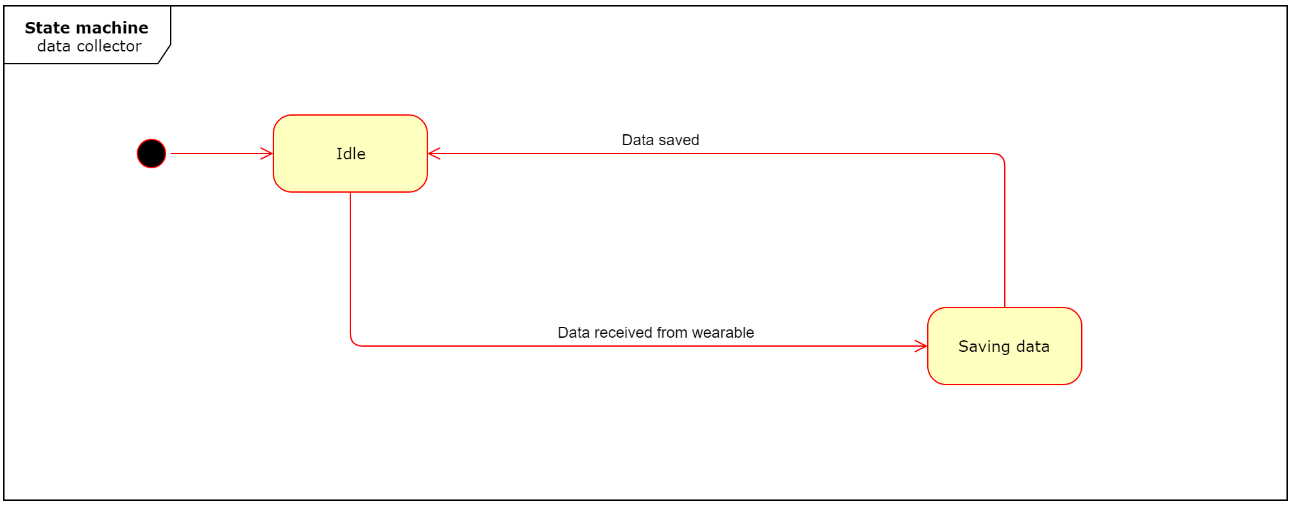
- Database queries.
- Anonymization check.
- Check of health parameters against risk thresholds.
- System enforces unicity of email addresses.

2.1.2 Diagrams

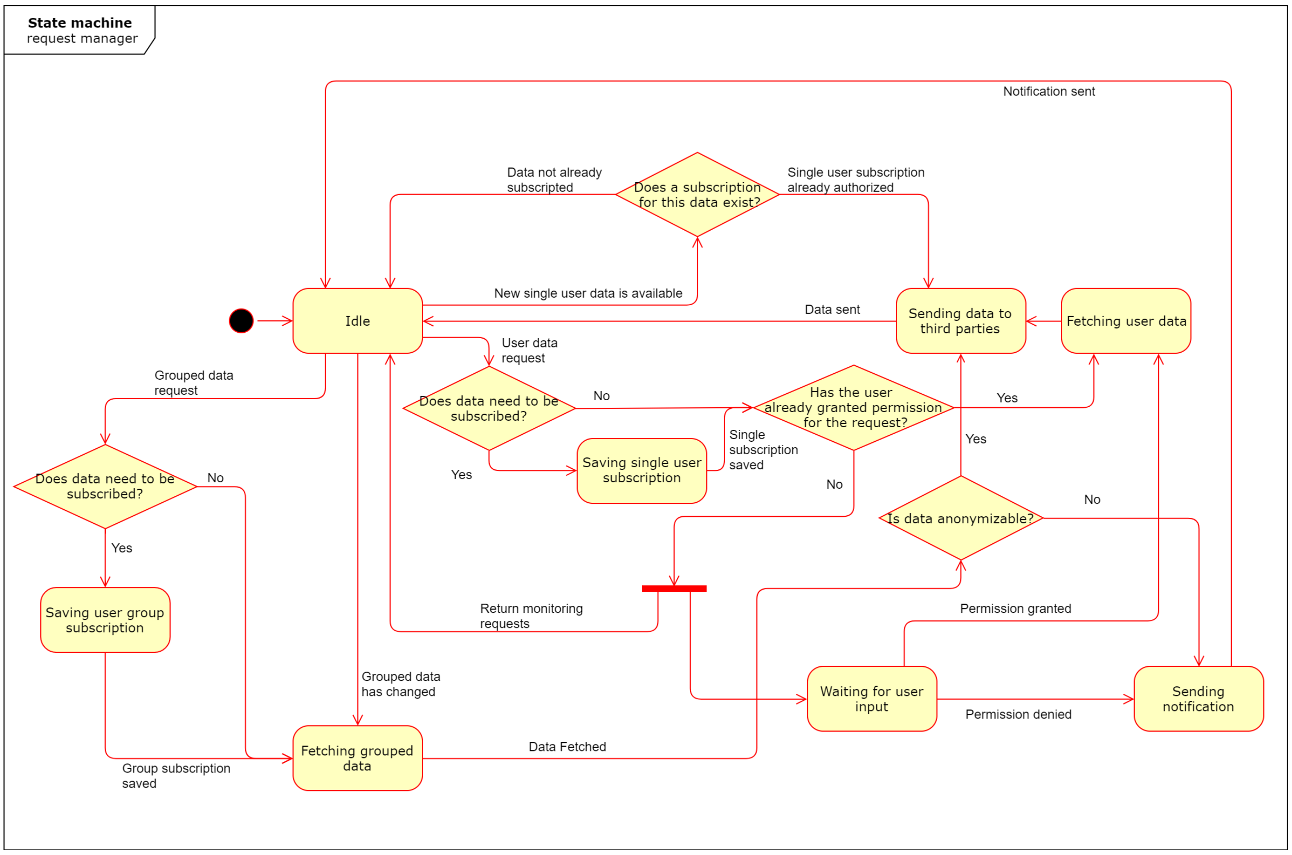


This *class diagram* models how the system is structured and the relations among its parts. One of the main objectives of this diagram is to highlight the system boundaries and in order to facilitate this, classes which are part of the World are colored in yellow, the classes related to Shared Phenomena in green and, finally, the part of the system which is observed only by the Machine is blue.
User and Third Party are the most important classes because the first one is the provider of Data gathered and stored by the Data Collector class, while the other one triggers the data

retrieval process regulated by the Request Handler class, whose task is to check if the Third-Party requests are accepted and fetch the desired data.
On top of this system two other services are built: *AutomatedSOS* and *Track4Run*. The former consists, in this diagram, of the Elderly class (which is a subclass of User) and of the Parameters Inspector, that monitors each elderly user's Health Status data to check if the parameters are below risk thresholds. The latter is composed by Athlete class, a subclass of User that represents all the Run participants, and by the Organizer, a subclass of Third Party that defines the Run. All runners' Locations can be seen on Map by the Spectator, another type of actor.



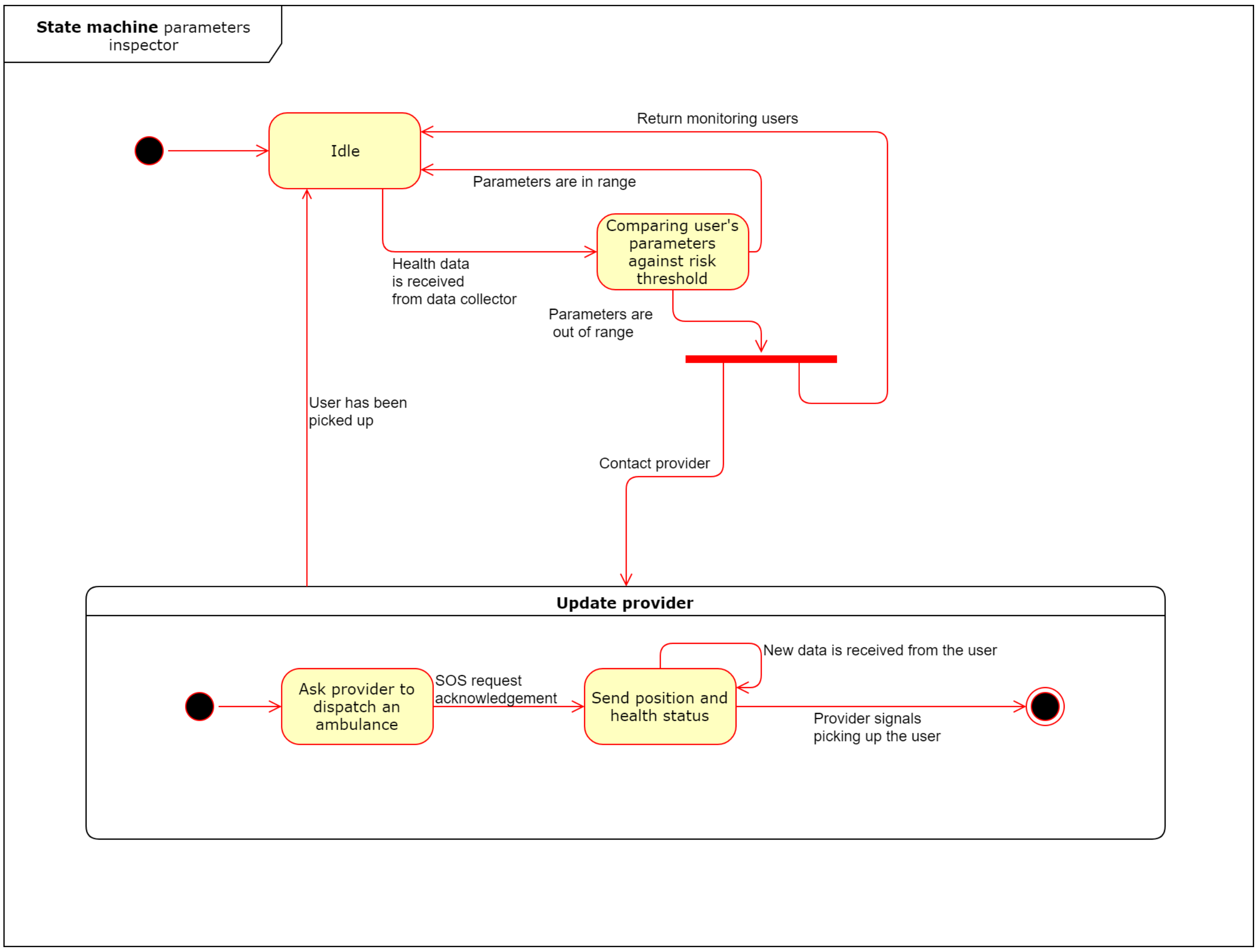
This state diagram shows the behavior of the *Data Collector* class, whose duty is to collect data from the sensors placed on the users' wearable devices. The state machine exhibits a cyclic behavior: starting from an idle state it awakens after data is received from a wearable device, then it proceeds to store it in the system. Once the save is completed, the state machine returns idle.


State machine
request manager
Notification sent
Data not already subscripted
Does a subscription for this data exist?
Single user subscription already authorized
New single user data is available
Idle
Data sent
Sending data to third parties
Fetching user data
User data request
Grouped data request
Does data need to be subscribed?
No
Has the user already granted permission for the request?
Yes
Single subscription saved
Saving single user subscription
Yes
No
Is data anonymizable?
No
Does data need to be subscribed?
No
Yes
Saving user group subscription
Return monitoring requests
Permission granted
Grouped data has changed
Waiting for user input
Permission denied
Sending notification
Fetching grouped data
Data Fetched
Group subscription saved

The request manager class, as the name suggests, manages data requests from third parties and fulfils them when data is available. The availability of the data depends on different conditions regarding the nature of the request:

- If the data request concerns a *single user's data*, the Request Manager shall verify if the user has previously granted the permission to allow access to their data from that specific third party. If the permission was not already granted, the Request Manager shall proceed to ask the user for permission and fulfil the request if they grant it. The fork point indicates that the request does not block the machine on waiting for the reply. Otherwise it shall notify the third party of the unavailability of the data.
- If the request is concerning group data, the state machine shall verify that the data can be anonymized (more than 1000 users) and, if this is possible, send theproperly anonymized data. Otherwise it shall notify the third party of the unavailability of such data.

Finally, the class is able to manage incoming new single user's data and dispatch it to the third parties, if they already placed an authorized request for such data. Regarding group data subscriptions, instead, after the set time interval has passed the state machine proceeds to check if anonymization on that group data is still possible and then will proceed to satisfy or deny the request as usual.



State machine parameters inspector
Idle
Return monitoring users
Parameters are in range
Comparing user's parameters against risk threshold
Health data is received from data collector
Parameters are out of range
User has been picked up
Contact provider
Update provider
New data is received from the user
Ask provider to dispatch an ambulance
SOS request acknowledgement
Send position and health status
Provider signals picking up the user

The *parameters inspector* class has the responsibility to monitor single users' health parameters and, if they get below risk thresholds, require the external ambulance provider to dispatch an ambulance and pick the user up to transport him to an health center. The state machine stays in an idle state until it receives health and GPS data from *data collector* class, then it activates and compares the user's parameters against the risk thresholds stored in the system and acts as follows:

- If the parameters are in range, the class returns idle since there is no risk situation detected.
- Else, if the parameters are out of range, the state machine enters a composite state called *update provider* to communicate all necessary information to the ambulance provider: first it places a request for an ambulance dispatch and then, when it receives acknowledgement for such request, starts sending health and position data of the user until the ambulance provider signals the system to have picked up the user. Finally, the state machine goes back to idle, exiting the composite state.

The fork after parameters out of range are discovered means that the state machine can monitor multiple users simultaneously and does not get fully occupied by a single emergency. To prevent ignoring other users getting sick while that specific emergency is being serviced, the *update provider* composite state shall be handled as a separated task that does not block the machine waiting for its completion.

2.2 Product Functions

Data4Help

Data4Help's purpose is to allow third parties to search and collect data about specific individuals or groups for their business or research projects. To enable this, individuals, who are users of the system willingly providing their data for collection from *Data4Help*, must be uniquely identified to allow to retrieve their data. Since users' privacy is nowadays an important concern, *Data4Help* won't allow third parties to monitor a user unless he explicitly accepts such behavior.

Data queries

To perform a single user's data query, a third party must be registered and access the system with their credentials. Then the system shall require the desired user's identifier, specifically their ID card number, and check if they have already granted permission to the third party to collect their data. If this is not the case, the user shall be notified through an email and asked if they want to allow data collection from that specific third party. If the user grants access to their data, then the system shall immediately provide it to the third party and send new data as soon as it is available, if this is required by the third party. In case the user denies the data access request, the system shall notify the third party of the unavailability of such data.

Group data queries are also allowed by the system but *Data4Help*, aiming to protect the privacy of its users, will guarantee their satisfaction only if they are concerning a group larger than 1000 individuals. In this case, the third party will specify a criterion to select users being sampled (e.g. people living in Città Studi block, with age between 18 and 25) and the system will immediately provide anonymized data of users matching the criterion, without previously asking permission to them.
When the anonymization threshold is not reached, the query shall not be satisfied and the concerned third party notified of the unavailability of such data.

For both types of query, the system enables third parties to subscribe to updates on future data, allowing to continuously track a single user or a group of individuals and receive data as soon it is available in the system.

AutomatedSOS
AutomatedSOS aims to be a "smart" SOS service for elderly people. Taking advantage of modern biometric sensors and wireless connections, the service monitors its users' real-time health status and position. An elderly person, that has reached a minimum age, can subscribe

to *AutomatedSOS* service upgrading their *Data4Help* account by inputting their risk thresholds, that have been evaluated through a preventive hospital checkup.
Whenever an *AutomatedSOS* user's health parameters get below risk thresholds, an external ambulance provider is contacted by the service and the user's location and health status are immediately signaled to allow for a steady pick up and an easier on-site treatment. Furthermore, sick user's data get continuously broadcasted to the ambulance provider to keep him updated until he confirms picking up the user.

Track4Run

Track4Run is an additional service which exploits *Data4Help* platform to allow an easy management of running competitions and gathering of statistics about runners' performance. Race organizers can register to the system as third parties and, after logging in, can insert new competitions into the system. Another feature available to them is to manage participants, adding or removing them manually from the participants' list that is shown in the competition detail page.
Standard *Data4Help* users can upgrade their account to an athlete profile to participate to runs available in the system.
After logging in, the system shows them a list of available races, which they can search through and select a single competition to enroll to. After selecting the competition, the details are shown to the user, who can confirm the enrollment or go back to the run list to check other competitions.
After completing a run, each user can check their data from their personal page and the competition's result.
Spectators are unregistered people who can spectate a run through a dedicated page, showing a live map with positions of each runner and a live leaderboard with runners' times. The desired run can be selected from a list of competitions that are live or will begin shortly.

2.3 User characteristics

2.3.1 Actors

- **User**: an individual registered in the system who has accepted to give his location and health data, through a wearable device provided by the company.
- **Elderly:** user who is subscribed to *AutomatedSOS* service.
- **Third Party**: an individual or organization registered in the system. It can login to the system, and then request user data, either a specific user's data or data about group of users (properly anonymized).

- **Run Organizer**: third party that can schedule run competitions through the Track4Run system. It can schedule a new run, define the path for a new run, check the athletes participating in a run and add or delete them.
- **Athlete**: a user who participate to a run organized through the Track4Run system. This actor can login to the system, see a list of all the scheduled runs and participate to the desired ones and also see data relative to his past run competitions.
- **Spectator:** an unregistered individual who can see a list of active runs in a public website. After selecting the desired one, the spectator can see a live map with the runners' positions on the track.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain Assumptions

D1: Users are uniquely identified by their ID number. [G1, G2]
D2: Information provided by the user during the registration process is assumed to be true. [G1, G2, G3]
D3: User's position is available through GPS. [G1]
D4: User's health related data values are available through a wearable personal device [G1]
D5: A partner of TrackMe provides an ambulance service 24/7. [G4]
D6: The risk thresholds for each user are obtained through a preventive hospital check. [G5]
D7: There is an external provider offering a map service [G8]
D8: Any user whose health parameters get below the risk thresholds once, is considered to be sick until they get medical treatment [G5]

3 Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The system to be will be based upon a web application, in order to provide the user and third parties with a personal area allowing access to the system functionalities that are respectively available to them.

Since at this stage of system analysis and specification there is no particular need in constraining graphic interfaces, we will analyze them in the Design Document after considering all the architectural choices of the system to be.

3.1.2 Hardware Interfaces

The system to be relies heavily on biometric and location data acquisition. In order to accomplish this, we have considered employing a rechargeable wearable device equipped with energy efficient sensors.
As an example, these sensors could include GPS, heart rate sensor, accelerometer, blood pressure sensor and GSR sensor.
To transmit the collected data to the system, the wearable devices must be equipped with a mobile radio system, allowing access to the communication network.

3.1.3 Software Interfaces

- **Maps service**: the system to be, specifically for its *Track4Run* service, needs access to a maps service in order to show runs spectators a live map of the participants' positions.
- **Ambulance Provider service**: the system to be communicates with an external Ambulance Provider through its API in order to require an ambulance dispatch, send sick users' data and receive confirmation of SOS requests completion.

3.1.4 Communication Interfaces

The system uses the usual standard communication protocols to transmit data over the internet.

3.2 Scenarios

Scenario 1
Kate would like to watch his friend Umberto running in the Run4Fun run competition organized by the company HealthFun. She opens the *Track4Run* website and scrolls through the list of active runs, until she finds Run4Fun's. Then she clicks on the "Watch on live map" button, and the browser shows a live map with the current runners' position, including Umberto's.

Scenario 2
The HealthAnalytica company would like to periodically monitor the health status of people aged from 18 to 25 and living in the Città Studi block in Milan. Jacob, an HealthAnalytica employee, logs into the *Data4Help* website and enters the "Group Data" section.
Then he selects the parameters for the desired data: mininum and maximum age, and the geographical area to consider. Next, he sets a Daily Update for this data, and eventually he clicks on Subscribe. Since there are not enough users satisfying the chosen criteria, the website shows an error message because it can't properly anonymize the data. So, the HealthAnalytica employee changes the search parameters, and clicks again on Subscribe. Now *Data4Help* shows a confirmation message to let Johnny know the request has been accepted successfully.

Scenario 3
Andy, a teenager registered to *Data4Help*, found out that *Data4Help* can help elderly people in case of need. Therefore, he tells his grandpa Mike, which is 66 years old, to subscribe to the *AutomatedSOS* service, so that Mike can feel safer when he is alone.
Mike goes to the doctor for a medical checkup, and after the visit he registers to *Data4Help* and subscribes to *AutomatedSOS* by inserting his risk thresholds obtained from the checkup into his *Data4Help* personal area.

Scenario 4
Arnold, a personal trainer working at the StayFit gym based in Milan, wants to write a personal workout schedule for a gym member.

Since each member has included in his membership also a subscription to *Data4Help* and has already agreed to send data to StayFit, Arnold wants to look up the member's data. In particular, in order to give him a schedule with exercises adequate to his level, Arnold is looking for the member's health status. Therefore, he opens *Data4Help*'s web page, logs in with the gym account and searches for the member's data using his ID number and now he is able to write the right work schedule for him.

3.3 Functional requirements

3.3.1 Data4Help

G1: The user can be recognized by providing a form of identification
[D1]: Users are uniquely identified by their ID number.
[D2]: Information provided by the user during the registration process are assumed to be true
[R1]: The system shall allow registration of individuals through an email and a password.
[R2]: The system shall guarantee the unicity of email addresses.
[R3]: The system shall ask users to provide their personal data (birthdate, gender, residency address, ID number).
[R4]: The system shall ask the user to agree to a policy that specifies that, by registering, users agree that TrackMe acquires their data.

G2: Allow third parties to monitor data about location and health status of individuals.
[D3]: User's position is available through GPS.
[D4]: User's health data (heart rate and blood pressure) are available through a wearable personal device.
[R5]: The system shall store past position and health data of every single user.
[R6]: The system shall support the registration of third parties and provide them with a unique identifier (ID).
[R30]: The system shall support the login of third parties with their ID and password.
[R7]: Third parties shall be allowed to subscribe to new data and the system shall send data as soon as they are produced.

G3: Allow third parties to access data relative to specific individuals
[D2]: Users are uniquely identified by their ID number.
[R30]: The system shall support the login of third parties with their ID and password.

[R31]: The system shall support the login of user with their email and password.
[R32]: User shall be able to grant or deny data collection permission to a ThirdParty by logging into his personal page
[R7]: Third parties shall be allowed to subscribe to new data and the system shall send data as soon as they are produced.
[R8]: Third parties shall be able to request and receive a specific user's data by providing user's ID number.
[R9]: Upon every data collection request, the system shall check if the permission to access that data has already been granted by the user, otherwise it shall ask them.

G4: Allow third parties to access anonymized data of groups of individuals
[R30]: The system shall support the login of third parties with their ID and password.
[R7]: Third parties shall be allowed to subscribe to new data and the system shall send data as soon as they are produced.
[R10]: The system shall allow third parties to search for the desired group of individuals.
[R11]: The system shall accept a group data request only if the data can be properly anonymized. Anonymization is considered proper if the number of people involved in the request is greater than 1000.
[R12]: The system shall allow third parties to subscribe to periodic updates relative to a certain group of individuals provided that the data contained in each update can be properly anonymized.

3.3.2 AutomatedSOS

G5: Allow third parties to offer a personalized and non-intrusive SOS service to elderly people so that an ambulance arrives to the location of the customer in case of emergency.
[D6]: The risk threshold for each user is obtained through a preventive hospital check.
[D8]: Any user whose health parameters get below the risk thresholds once is considered to be sick until they get medical treatment.
[R31]: The system shall support the login of user with their email and password.
[R13]: The system, in order to allow *Data4Help* users to subscribe to *AutomatedSOS*, shall ask them to input their risk threshold.
[R14]: Frequently enough, health parameters of each user are monitored by the system and compared against their threshold to detect risk situations.
[R15]: If a user's parameters get below risk thresholds the system shall contact the ambulance provider and require the dispatch of an ambulance.
[R16]: After an emergency request has been placed the system shall send data about the user and keep sending them regularly to the ambulance provider.

[R17]: The system shall allow the ambulance provider to notify when the user has been picked up, in order to stop data transmission.

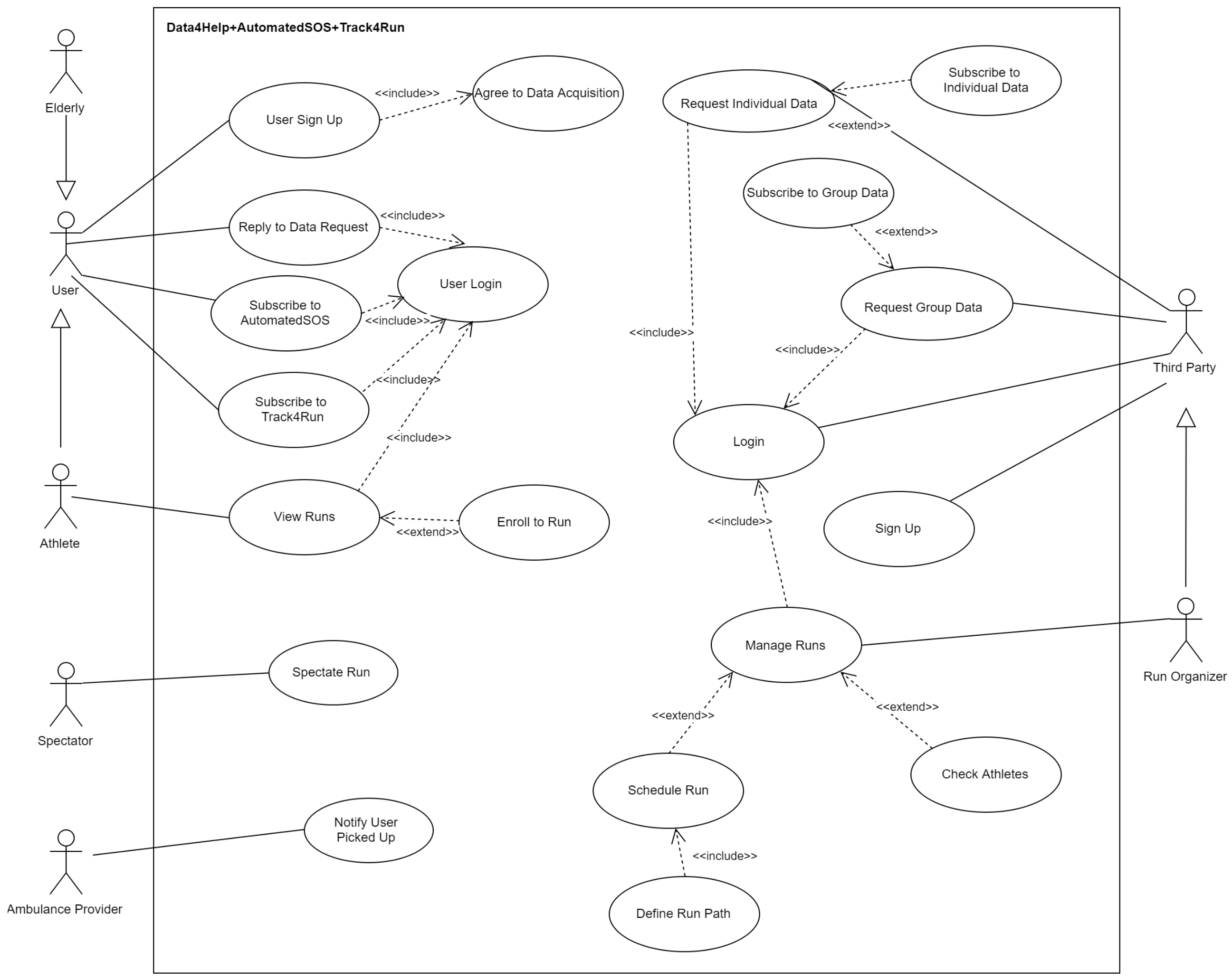
3.3.3 Track4Run

G6: Allow athletes to enroll in a run
[R18]: The system shall allow athletes to register to the system.
[R31]: The system shall support the login of user with their email and password.
[R19]: The system shall allow athletes to check a list of available runs.
[R20]: The system shall provide the ability to enroll to the desired run only to registered athletes.
[R21]: The system shall allow enrolling only if the user has already agreed to share publicly his location for the duration of the run.

G7: Allow organizers to manage runs
[R30]: The system shall support the login of third parties with their ID and password.
[R22] The system shall allow organizers to register to *Track4Run* platform.
[R23] The system shall allow organizers to create and delete races.
[R24] The system shall allow organizers to add a path for the run.
[R25] The system shall allow organizers to check a participants list.
[R26] The system shall allow organizers to add or remove participants before the start of the race.

G8: Allow spectators to see on a map the position of all runners during the run
[D7] There is an external provider offering a map service.
[R27] The system shall provide a public list of live runs and allow the spectator to select one of them.
[R28] The system shall allow the spectator to see a map of the desired run, with live participants' position.
[R29] The system shall update the positions of the runners on the map as soon as new data is received.

3.3.4 Use Case Diagram



3.3.5 Use Cases

Name	Sign Up *(UC1)*
Actor	Third Party
Entry Condition	This use case starts when Third Party clicks the “Sign up as third party” button on *Data4Help* webpage
Flow of Events	1. The system shows the registration form to Third Party 2. Third Party fills all the mandatory fields and provides the necessary information. 3. They click on the “Confirm” button. 4. The system generates an unique ID which identifies the Third Party
Exit Condition	This use case terminates when the registration is completed and from now on the Third Party is able to login with the ID provided by the system
Exceptions	1. Third Party misses to fill some mandatory information or the data inserted is not valid. This exception is handled by the system who notifies the Third Party and disable the confirm button while the data are not complete or valid. 2. The Third Party is already registered to the system. This exception is handled by notifying the Third Party and taking him back to the homepage.

Name	Login *(UC2)*
Actor	Third Party
Entry Condition	The third party clicks on "Login" button on *Data4Help* webpage
Flow of Events	1. The system shows the login form 2. The third party enters the requested credentials 3. The third party clicks on "Login as a Third Party" button
Exit Condition	The login is successful, and the system shows the third party's personal area.
Exceptions	1. The third party enters invalid data in the form. The system shows an error message and then returns to the form page.

Name	Request Individual Data *(UC3)*
Actor	Third Party
Entry Condition	The Third Party clicks on “Search Individual” button on their *Data4Help* webpage
Flow of Events	1. Third Party inserts the single individual’s ID number or fiscal code on an input form 2. The system shows the searched User 3. Third Party clicks on “Request Data” 4. The system checks if the searched User has already granted access to Third Party 5. If not, the system sends a notification to User
Alternative flow	5. The desired User has already granted access to their data 6. Third Party receives user’s data
Exit Condition	The request is successful
Exceptions	1. There is no individual related to the input and no user is shown to the Third Party. The system shows again the input form to them. 2. The user has not granted access to their data, the system notifies the third party

Name	Request Group Data *(UC4)*
Actor	Third Party
Entry Condition	The Third Party clicks on “Search Group” button on their *Data4Help* webpage
Flow of Events	1. Third Party inserts the search parameters (geographical area, gender, age and so on) 2. The system fetches all the data which satisfy the parameters 3. The system verifies that it is able to anonymize the requested data 4. The system sends group data to Third Party
Exit Condition	Group data is sent to Third Part

Exceptions	1. Group data can’t be anonymized. The system reacts by notifying the Third Party with an error message.

Name	Subscribe to Individual Data (*UC5)*
Actor	Third Party
Entry Condition	A data request has just completed successfully
Flow of Events	1. A dialog window asking if the Third Party wants to subscribe to new data for the same request is shown by the system 2. The Third Party clicks on the “Yes” button
Exit Condition	The dialog window is closed, and the subscription is registered in the system
Exceptions	1. The Third-Party clicks “No”. In this case the system will not send new data to the Third Party about that specific individual

Name	Subscribe to Group Data *(UC6)*
Actor	Third Party
Entry Condition	A group data request has just completed successfully
Flow of Events	1. A dialog asking if the Third Party wants to subscribe to new data for the same request is shown by the system 2. The Third Party specifies the time interval between each new data update 3. Third Party clicks “Confirm” button
Exit Condition	The dialog window is closed, and the subscription is registered in the system
Exceptions	1. Third Party selects “No Update” option. In this case the system will not send new data to the Third Party about that group

Name	User Sign Up *(UC7)*
Actor	User
Entry Condition	The user clicks on the "Sign up as user" button on *Data4Help* webpage
Flow of Events	1. The system shows the user registration form 2. The user fills all the mandatory fields and provides necessary information 3. The user clicks on the "Submit data" button
Exit Condition	The registration is successful and from now on the user can grant access to their data
Exceptions	1. The user misses to fill some mandatory information or the data inserted is not valid. This exception is handled by the system who notifies the User and disables the confirm button as long the data are not complete or valid 2. The username provided by the user is not available. The system shows an error message to inform the user and asks them to insert a new one

Name	Agree to Data Acquisition *(UC8)*
Actor	User
Entry Condition	The user is about to complete the registration process and has clicked the "Submit data" button
Flow of Events	1. The system shows the user a dialog asking to confirm their will to agree to personal and biometric data collection, along with a "Yes" and a "No" button 2. The user clicks "Yes" 3. The user is shown a confirmation page
Alternative flow	1. The system shows the user a dialog asking to confirm their will to agree to personal and biometric data collection, along with a "Yes" and a "No" button 2. The user clicks "No" 3. The user is shown a warning message stating that he has to accept data collection in order to complete the registration 4. The user dismisses the message 5. The user clicks "Yes" 6. The user is shown a confirmation page
Exit Condition	The user exits the confirmation page
Exceptions	1. The user closes the web page before clicking "Yes", the data is deleted and any new sign up for that user will start from scratch

Name	User Login *(UC9)*
Actor	User
Entry Condition	User has landed on the login page
Flow of Events	1. The user types login credentials in the login form 2. The user clicks on the "Login as user" button
Exit Condition	The system recognizes the credentials inserted by the user
Exceptions	1. The credentials inserted by the user are invalid, the system notifies the user and asks to try again.

Name	Reply Data Request *(UC10)*
Actor	User
Entry Condition	The user receives an email containing a request from a third party to access his data
Flow of Events	1. The user login on his personal page 2. The user opens the page containing all the third parties' requests 3. User replies to the third party's request
Exit Condition	The system notifies the involved third party about the user's choice
Exceptions	1. The user doesn't make any choice. The system does not provide any data (or denial notification) to the third party until a decision is made.

Name	Subscribe to *AutomatedSOS (UC11)*
Actor	User
Entry Condition	User has clicked on the "Subscribe to *AutomatedSOS*" button, located in their personal area of the website
Flow of Events	1. The system shows the user a page explaining the *AutomatedSOS* terms of service. 2. The user clicks on the "Accept" button 3. The system updates the enabled services for this specific user
Exit Condition	The system has correctly added *AutomatedSOS* to the user's services
Exceptions	1. The user clicks on "Cancel" or closes the web page before confirming the terms of service, the system does not update the enabled services for that user.

Name	Subscribe to *Track4Run (UC12)*
Actor	User
Entry Condition	User has clicked on the "Subscribe to *Track4Run*" button, located in their personal area of the website
Flow of Events	1. The system shows the user a page explaining the *Track4Run* terms of service. 2. The user clicks on the "Accept" button 3. The system updates the enabled services for this specific user
Exit Condition	The system has correctly added *Track4Run* to the user's services
Exceptions	1. The user clicks on "Cancel" or closes the web page before confirming the terms of service, the system does not update the enabled services for that user.

Name	View Runs *(UC13)*
Actor	Athlete
Entry Condition	The Athlete has clicked on the "Available runs" button in his personal area
Flow of Events	1. The system provides an available run list to the user 2. The user clicks on the "Show details" button associated with a single run 3. The system shows a page containing the run details to the user
Exit Condition	The user can view the details page
Exceptions	1. The user does not select a run to detail, no run is shown

Name	Enroll to Run *(UC14)*
Actor	Athlete
Entry Condition	The athlete clicks on the "Enroll" button in the run details page
Flow of Events	1. The system shows the enrollment form 2. The athlete provides any additional required data, apart from the information already collected from Track4Me 3. The athlete clicks on "Confirm" button
Exit Condition	The enrollment is successful
Exceptions	1. The athlete fails to fill in some mandatory information, or the data inserted is not valid. The system notifies the athlete and disables the confirm button as long the data are not complete or valid. 2. The athlete inserted invalid credentials. The system shows an error message and asks to re-input correct credentials.

Name	Manage Runs *(UC15)*
Actor	Run Organizer
Entry Condition	Run Organizer has successfully logged in
Flow of Events	1. A page containing all the runs managed by the organizer is shown by the system 2. The organizer selects a run to manage through the “Select” button relative to it 3. The system shows an administration page for the desired run
Exit Condition	The page is successfully shown to the organizer
Exceptions	1. The organizer does not select any run to manage, no run is shown

Name	Schedule Run *(UC16)*
Actor	Run Organizer
Entry Condition	The run organizer has clicked the “Add new run” button in the personal area
Flow of Events	1. The run scheduling form is presented 2. The run organizer fills in the required fields (such as start and end time and date of the run) 3. The run organizer clicks the “Define run path” button 4. The run organizer has correctly defined the run path
Exit Condition	The run data is stored successfully
Exceptions	1. The organizer inputs past dates or starting time for the run, an error message is shown, and the organizer is asked to input correct data 2. The organizer clicks the “Cancel” button or closes the page before confirming data, the run data is not saved to the system 3. The organizer does not define the run path, the run is not stored in the system

Name	Define Run Path *(UC17)*
Actor	Run Organizer
Entry Condition	The run organizer has clicked the “Define run path” button while adding a new run
Flow of Events	1. The run path definition dialog is shown 2. The organizer inputs the run path 3. The organizer clicks the “Save path” button
Exit Condition	The path data is correctly stored in the system
Exceptions	1. The organizer clicks cancel or closes the dialog before having specified the run path, the schedule run page is shown again

Name	Check Athletes *(UC18)*
Actor	Run Organizer
Entry Condition	The run organizer has clicked the "Show Athletes" button in the administration page of a selected run
Flow of Events	1. The system shows a list of the Athletes participating in the run 2. The Run Organizer select an Athlete by clicking on his name 3. The system shows the Athlete's detail to the Run Organizer
Exit Condition	The Run Organizer close the Athlete's list and return to the previous page
Exceptions	1. No one is participating to the run. The system shows a message and returns to the previous page.

Name	Spectate Run *(UC19)*
Actor	Spectator
Entry Condition	Spectator lands on "Live runs" page
Flow of Events	1. The system shows a list of live runs or about to start 2. The spectator selects a run through the relative "Select" button 3. The system shows a page relative to the run containing the live map and the scoreboard.
Exit Condition	The spectator leaves the page
Exceptions	1. The spectator does not select a run to spectate, no live map page is shown

Name	Notify User Picked Up *(UC20)*
Actor	Ambulance Provider
Entry Condition	The ambulance crew informs the external provider that they have picked up the user
Flow of Events	1. The operator opens the API interface 2. The operator inputs the identifier of the user that has been picked up 3. The operator confirms that the emergency has been taken care of
Exit Condition	The system acknowledges the SOS request ending
Exceptions	1. The inserted user number does not correspond to any SOS request or does not exist, the API replies with an error code

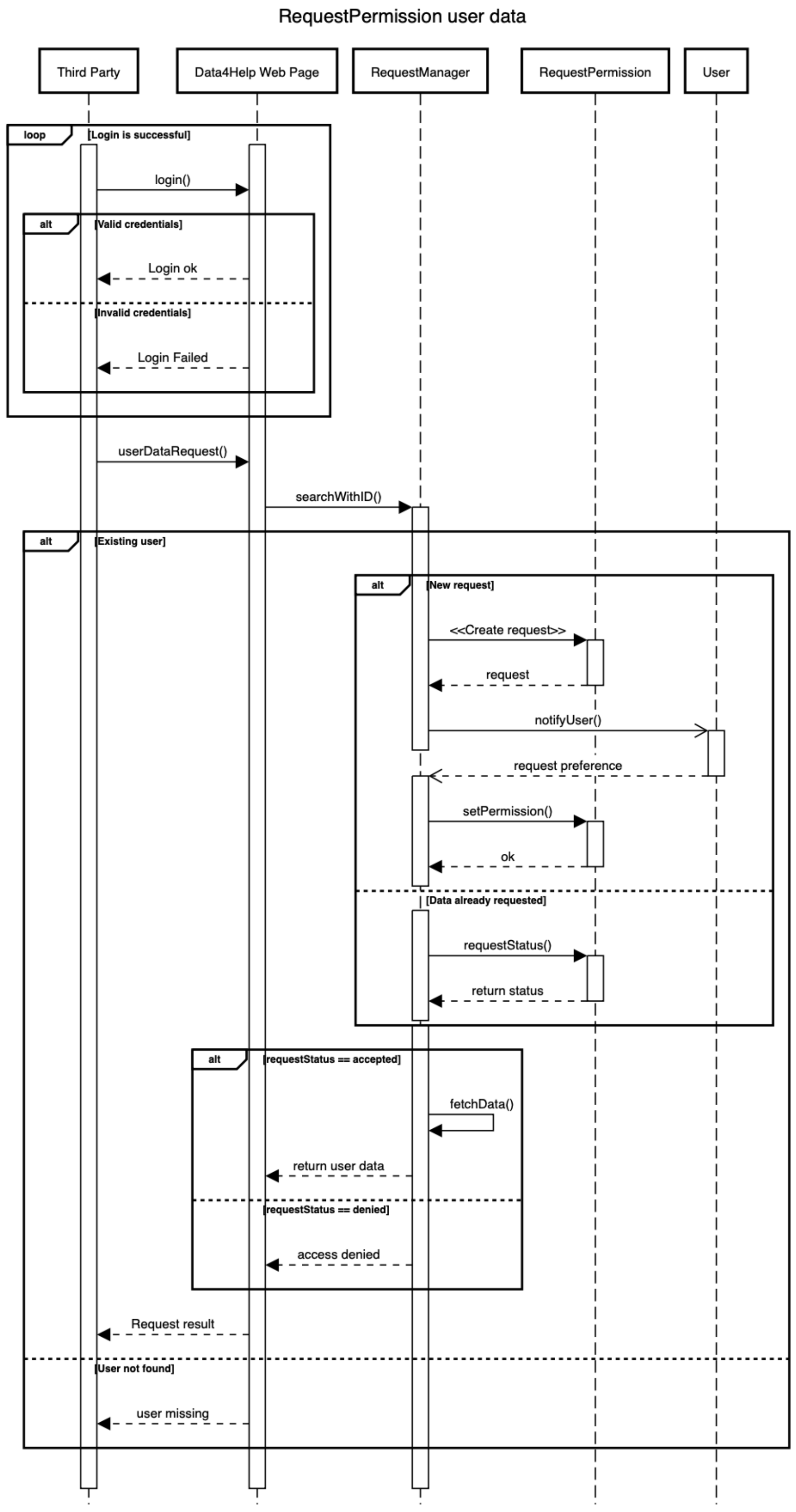
3.3.6 Traceability Matrix

Goal ID	**Req ID**	**Use Case ID**
G1	R1, R2, R3, R4	UC7
G2	R6	UC1
G2, G3, G4	R7	UC5, UC6
G2, G3, G4, G7	R30	UC2
G3	R8	UC3
G3	R9	UC3
G3	R31	UC9
G3	R32	UC10
G4	R10, R11	UC4
G4	R12	UC6
G5	R31	UC9
G5	R13	UC11
G5	R17	UC20
G6	R18	UC7, UC12
G6	R31	UC9
G6	R19	UC13
G6	R20, R21	UC14
G7	R22	UC1
G7	R23	UC16

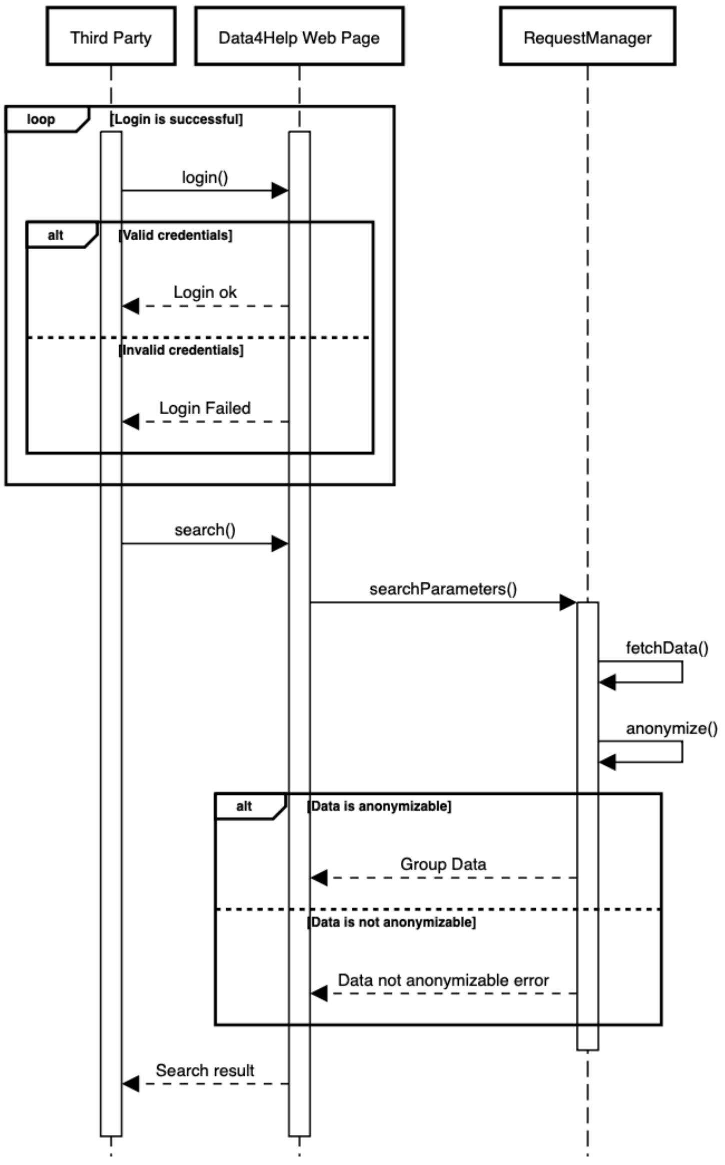
G7	R24	UC17
G7	R25, R26	UC18
G8	R27, R28, R29	UC19

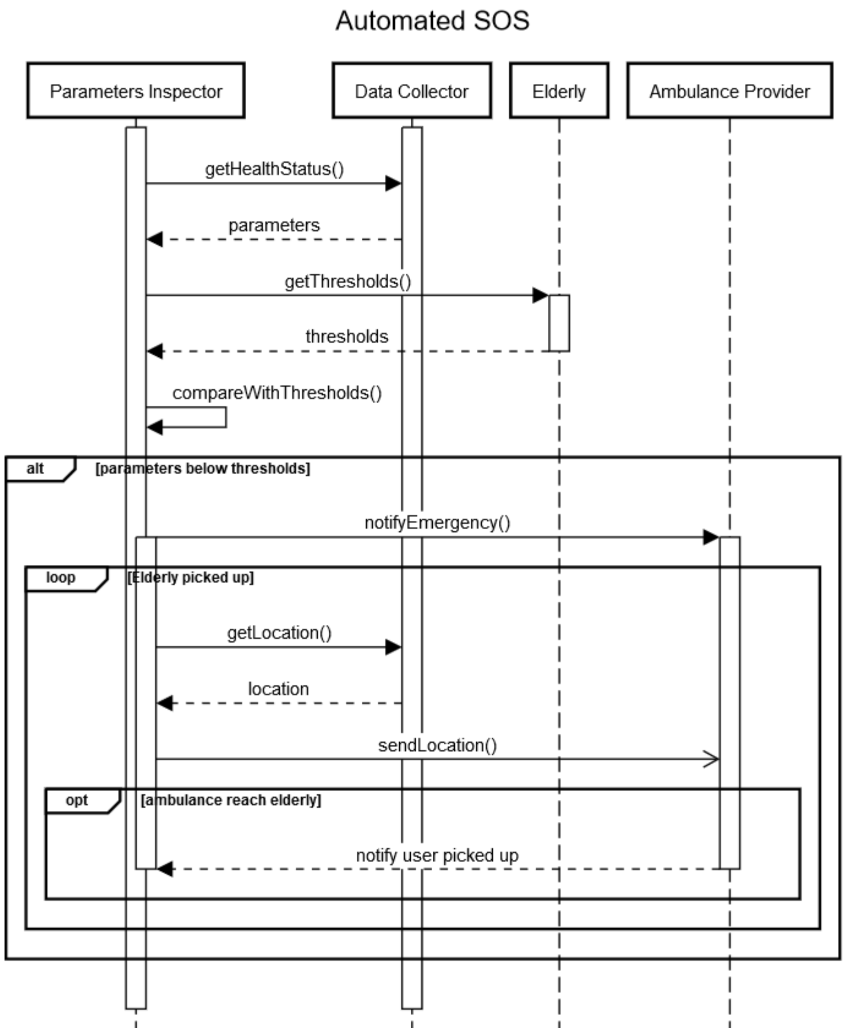
UC1: Sign Up
UC2: Login
UC3: Request Individual Data
UC4: Request Group Data
UC5: Subscribe to Individual Data
UC6: Subscribe to Group Data
UC7: User Sign Up
UC8: Agree to Data Acquisition
UC 9: User Login
UC10: Reply Data Request
UC11: Subscribe to *AutomatedSOS*
UC12: Subscribe to *Track4Run*
UC13: View Runs
UC14: Enroll to Run
UC15: Manage Runs
UC16: Schedule Run
UC17: Define Run Path
UC18: Check Athletes
UC19: Spectate Run
UC20: Notify User Picked Up

3.3.7 Sequence Diagrams

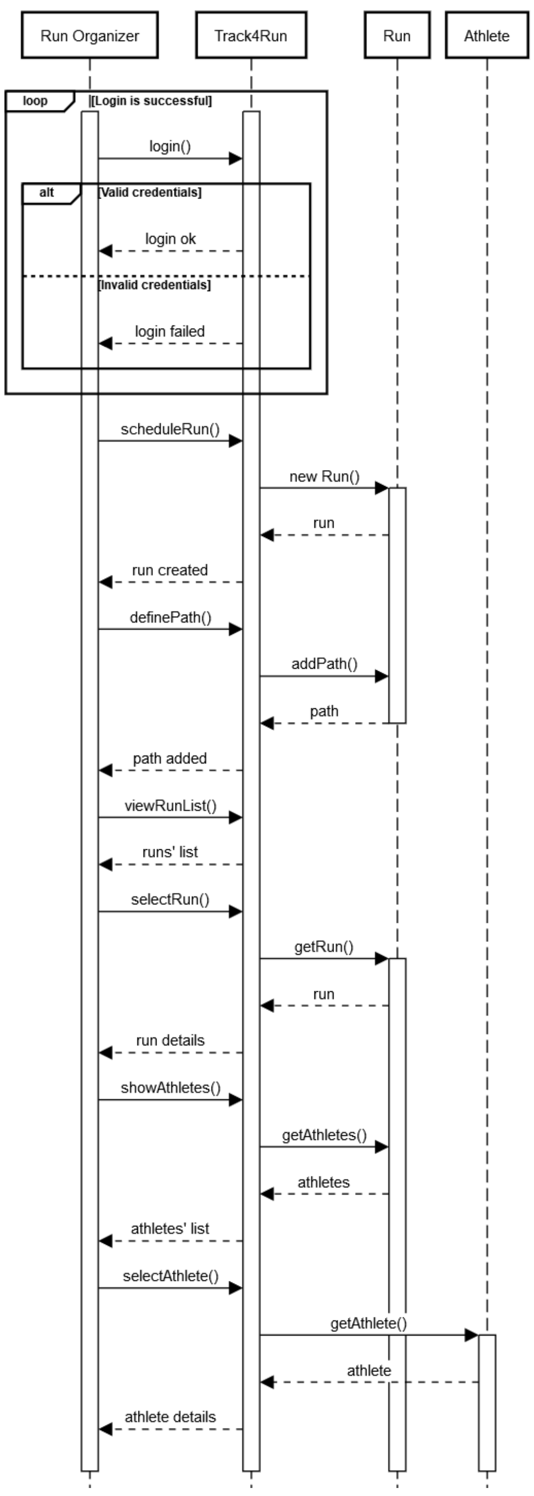


Request group data




Automated SOS
Parameters Inspector
Data Collector
Elderly
Ambulance Provider
getHealthStatus()
parameters
getThresholds()
thresholds
compareWithThresholds()
alt
[parameters below thresholds]
notifyEmergency()
loop
[Elderly picked up]
getLocation()
location
sendLocation()
opt
[ambulance reach elderly]
notify user picked up

Manage Run


Run Organizer
Track4Run
Run
Athlete
loop
[Login is successful]
login()
alt
[Valid credentials]
login ok
[Invalid credentials]
login failed
scheduleRun()
new Run()
run
run created
definePath()
addPath()
path
path added
viewRunList()
runs' list
selectRun()
getRun()
run
run details
showAthletes()
getAthletes()
athletes
athletes' list
selectAthlete()
getAthlete()
athlete
athlete details

3.4 Performance requirements

G5: Allow third parties to offer a personalized and non-intrusive SOS service to elderly people so that an ambulance arrives to the location of the customer in case of emergency.
[R1-NF]: An SOS request shall be placed to the external provider in less than 5 seconds from the time the health parameters are detected to be below the threshold shall be guaranteed by the system.

3.5 Design constraints

3.5.1 Standards Compliance

At this level of analysis, the system does not appear to need to be directly compliant to any standard, since it does not interface or interact with standardized data or systems. Any further compliance will be defined after design analysis.

3.5.2 Hardware limitations

This is a software application with no strict hardware limitations. However, to use the system functionalities, Users and Third Parties must have a device connected to the internet with a web browser.
Regarding the users' wearable there is no hardware limitation for them because they are provided for free by TrackMe after the registration.

3.5.3 Other Constraints

Each user must have an email client to receive third party's data requests notifications and accept or deny them.

3.6 Software System Attributes

3.6.1 Reliability

The *AutomatedSOS* and *Data4Help* services, being involved in emergency handling, should be available 24/7.
Track4Run, instead, can be allowed to have downtime periods, due to its non-critical nature.

3.6.2 Availability

AutomatedSOS is a risk-critical service and should be 99.999% available, since the users' lives depend on it.
Since *AutomatedSOS* relies on the Data Collector class (which represent just a part of the *Data4Help* service) it should be 99.999% available too.
Larger tolerance can be admitted for *Track4Run*, for which a 90% availability is considered to be sufficient, and *Data4Help*, with a 99% availability rate.

3.6.3 Security

User data should be strongly encrypted. At this stage of the analysis the algorithm is not relevant and will be discussed in the DD.

3.6.4 Portability

Due to being web based, the system is expected to have the maximum grade of portability, requiring only a web browser to be interacted with.

4 Formal Analysis using Alloy

4.1 What we want to model?

In this section we want to test the correctness of some of the core components and features of the system to be.
In particular, we are interested in modeling the behavior of:

- A third party has access to specific users' data only if the users have granted the permission to access their own data.
- A third party has access to group data only if they are anonymized.
- When an elderly's health status gets below risk thresholds, the elderly is inserted in the Ambulance Provider's emergency list.
- An athlete can enroll only to runs that are not yet begun.

4.1.1 Why are we modeling these things?

We model the constraints regarding the third-party access to data because they constitute one of the core functionalities of the system to be, so it is important that they are satisfied. The same holds for the behavior of the system when an elderly is in a risk status, which is the main goal of AutomatedSOS service.
About the athlete enrollment to a run, we wanted to highlight one of the features of Track4Run service, in order to have a global view of the system to be.

4.1.2 Explanation of some choices

- The RequestHandler does not have a GroupDataRequest attribute, as shown in class diagram in section 2.1. We assigned userData and groupData directly to ThirdParty in order to check more easily constraints on Alloy. userData and groupData simulate the access to data from third parties.
- Enrolled runs are only the future runs in which the athlete will run. To represent the present, so that we know which are future and past runs, we used the constant value *3*.
- For simplicity reasons, and without loss of generality, we used the Int type to model all the users' data values (HealthStatus and Location attributes). The same holds for User

and ThirdParty ID (which is not so far from reality: usually each row of a database table is associated with an incremental integer index).

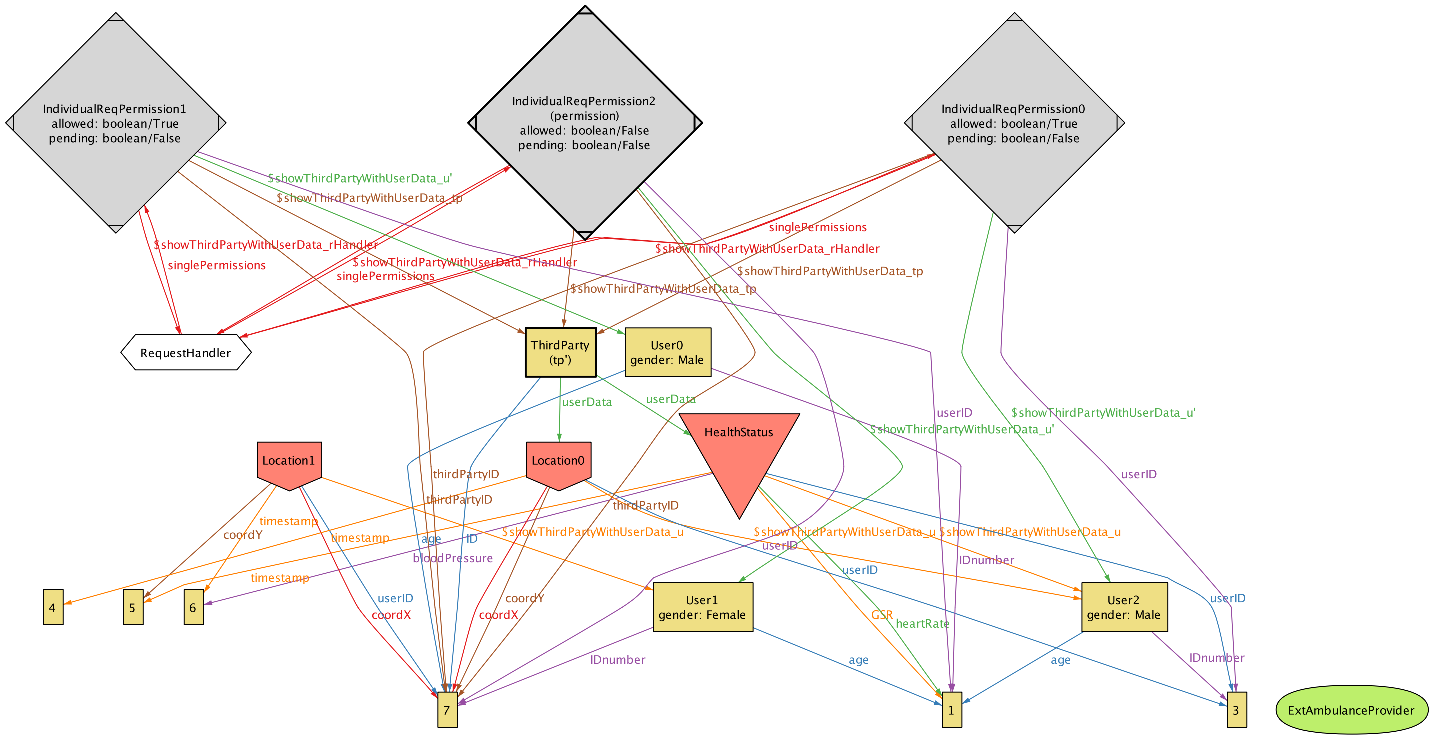
- Thresholds have a constant timestamp, which we assigned to value 0, in order to distinguish them from non-threshold values.
- For GroupData to be anonymized, we used the value *3* instead of *1000* to limit the size of Alloy generated worlds, without loss of generality.

4.2 Alloy Model

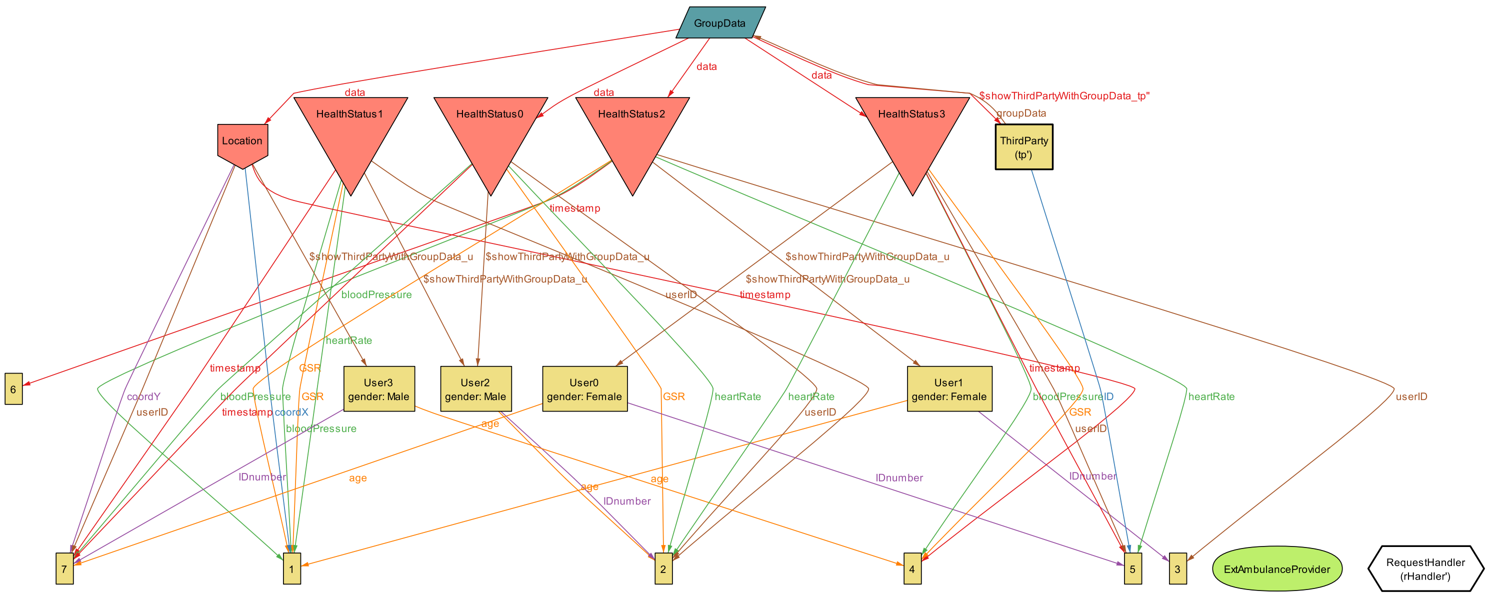
```
open util/integer
open util/boolean

abstract sig Data {
  userID: Int,
  timestamp: Int
} { timestamp >= 0 }

-- Int used for simplicity, instead of float
sig HealthStatus extends Data {
  bloodPressure: Int,
  heartRate: Int,
  GSR: Int
} { GSR > 0 and heartRate > 0 and bloodPressure > 0 }

-- Int used for simplicity, instead of float
sig Location extends Data {
  coordX: Int,
  coordY: Int
}

sig GroupData {
  data: set Data
}

abstract sig Gender {}
one sig Male extends Gender {}
one sig Female extends Gender {}

-- IDnumber: Int used for simplicity, instead of String
sig User {
  IDnumber: Int,
  age: Int,
  gender: Gender
} { IDnumber > 0 and age > 0 }

-- timestamp = 0 used to assign a constant value to thresholds
sig Elderly extends User {
  threshold: HealthStatus
}{ age > 5 and threshold.timestamp = 0 }
```

```
-- enrolledRuns: runs in which athlete will participate (in the future)
sig Athlete extends User {
  enrolledRuns: set Run
}

sig ThirdParty {
  ID: Int,
  userData: set Data,
  groupData: set GroupData
} { ID > 0 }

one sig ExtAmbulanceProvider {
  peopleToRescue: set Elderly
}

one sig RequestHandler {
  singlePermissions: set IndividualReqPermission
}

sig IndividualReqPermission {
  userID: Int,
  thirdPartyID: Int,
  allowed: Bool,
  pending: Bool
} { pending = True implies allowed = False }

sig Run {
  startTime: Int,
  endTime: Int
} { startTime > 0 and startTime < endTime }


fact usersAreUnique {
  no disj u1, u2: User | u1.IDnumber = u2.IDnumber
}

fact thirdPartiesAreUnique {
  no disj tp1, tp2: ThirdParty | tp1.ID = tp2.ID
}

fact OnlyThresholdsHaveTimestampZero {
  all data: Data | !(all old: Elderly | old.threshold.userID = data.userID) implies
data.timestamp > 0
}

-- every instance of Data is associated to one user
fact UserDataConnection {
  all data: Data | (some u: User | data.userID = u.IDnumber)
}

-- two health status data related to the same user can't have the same timestamp
-- the same holds for location data
fact DataHaveUniqueTimestamp {
  no disj loc1, loc2: Location | loc1.userID = loc2.userID and loc1.timestamp = loc2.timestamp
  and
  no disj hStatus1, hStatus2: HealthStatus |
    hStatus1.userID = hStatus2.userID and hStatus1.timestamp = hStatus2.timestamp
}

-- every IndividualReqPermission belongs to the RequestHandler
fact IndividualPermissionsBelongToRequestHandler {
  all permission: IndividualReqPermission |
    (some rHandler: RequestHandler | permission in rHandler.singlePermissions)
}

-- every IndividualReqPermission is associated to a User and a ThirdParty
fact IndividualPermissionsThirdPartyUserConnection {
  all permission: IndividualReqPermission |
    (some u: User, tp: ThirdParty | permission.userID = u.IDnumber and permission.thirdPartyID
= tp.ID)
}

-- there are no two or more IndividualReqPermission associated to the same User and ThirdParty
fact IndividualPermissionsThirdPartyUserAreUnique {
  no disj perm1, perm2: IndividualReqPermission |
    perm1.userID = perm2.userID and perm1.thirdPartyID = perm2.thirdPartyID
}

-- every ThirdParty has access only to specific users' data
-- for which they have given permission
fact thirdPartyCanAccessOnlyToGrantedData {
  all tp: ThirdParty | (
    all data: tp.userData | (
      some permission: IndividualReqPermission |
         permission.userID = data.userID and permission.thirdPartyID = tp.ID and
permission.allowed = True
    )
  )
}

-- GroupData is a collection of different users' data
fact DataWithinGroupDataBelongToDifferentUsers {
  all groupData: GroupData |
    (no disj sData1, sData2: groupData.data | sData1.userID = sData2.userID)
}

/*
Every ThirdParty can access only to group data that is anonymized.
The constant '3' represents the minimum number of data such that it is possible to anonymize
it.
In this model the procedure that removes personal information from users data is not
considered
*/
fact ThirdPartyCanAccessOnlyToAnonymizedGroupData {
  all tp: ThirdParty | (all groupData: tp.groupData | #groupData.data > 3)
}

-- every Elderly, whose last HealthStatus exceed risk thresholds,
-- is in ExtAmbulanceProvider.peopleToRescue
fact ElderlyInNeedAreKnownByExtAmbulanceProvider {
  all old: Elderly | (
    some hStatus: HealthStatus | hStatus.userID = old.IDnumber and
                                (hStatus.bloodPressure > old.threshold.bloodPressure or
                                 hStatus.GSR < old.threshold.GSR or
                                 hStatus.heartRate < old.threshold.heartRate) and
    (all hStatus2: HealthStatus | hStatus2 != hStatus implies hStatus2.timestamp <
hStatus.timestamp)
  ) implies some amb: ExtAmbulanceProvider | old in amb.peopleToRescue
}

-- every Athlete is enrolled only to runs that have not begun yet
-- the constant '3' represents the value "now"
fact AthletesAreEnrolledOnlyInFutureRuns {
  all ath: Athlete | (all enRun: ath.enrolledRuns | enRun.startTime > 3)
}

pred showThirdPartyWithUserData {
  some tp: ThirdParty | #tp.userData > 1
  and some permission: IndividualReqPermission | permission.allowed = False
  and no Run and no Athlete and no Elderly
  and #ThirdParty = 1
}

pred showThirdPartyWithGroupData {
  some tp: ThirdParty | #tp.groupData > 0
  and some rHandler: RequestHandler | #rHandler.singlePermissions = 0
  and all gData: GroupData | (some tp: ThirdParty | gData in tp.groupData)
  and no Run and no Athlete and no Elderly
  and #ThirdParty = 1
}

pred showAutomatedSOS {
  some ambulance: ExtAmbulanceProvider | #ambulance.peopleToRescue > 0
  and no Athlete and no Run
  and no ThirdParty and no GroupData
  and one User
}

pred showAthleteEnrolled {
  some ath: Athlete | #ath.enrolledRuns > 0
  and some sRun: Run | sRun.startTime < 3
  and no ThirdParty and no Elderly and no GroupData and no IndividualReqPermission
  and #Athlete > 1
}

run showThirdPartyWithUserData for 3
run showThirdPartyWithGroupData for 3 but 5 Data, 5 User
run showAutomatedSOS for 3
run showAthleteEnrolled for 3
```

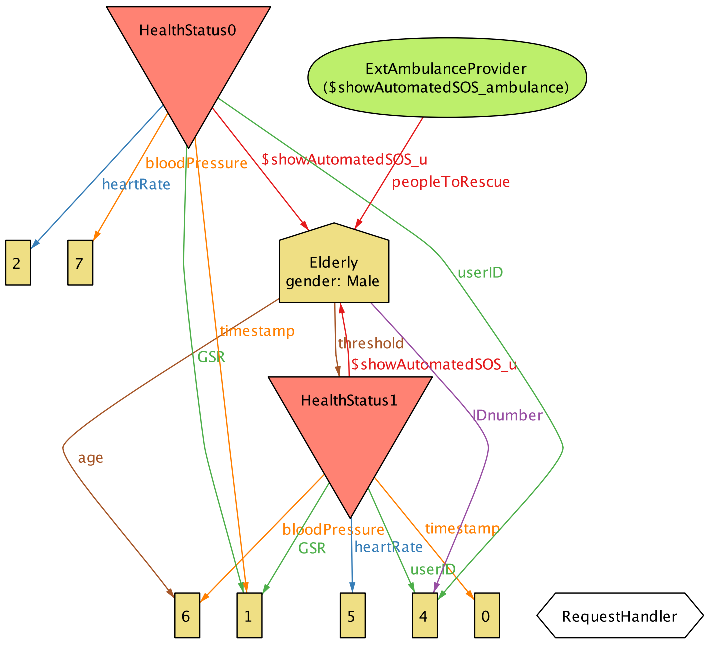
4.3 Alloy generated worlds



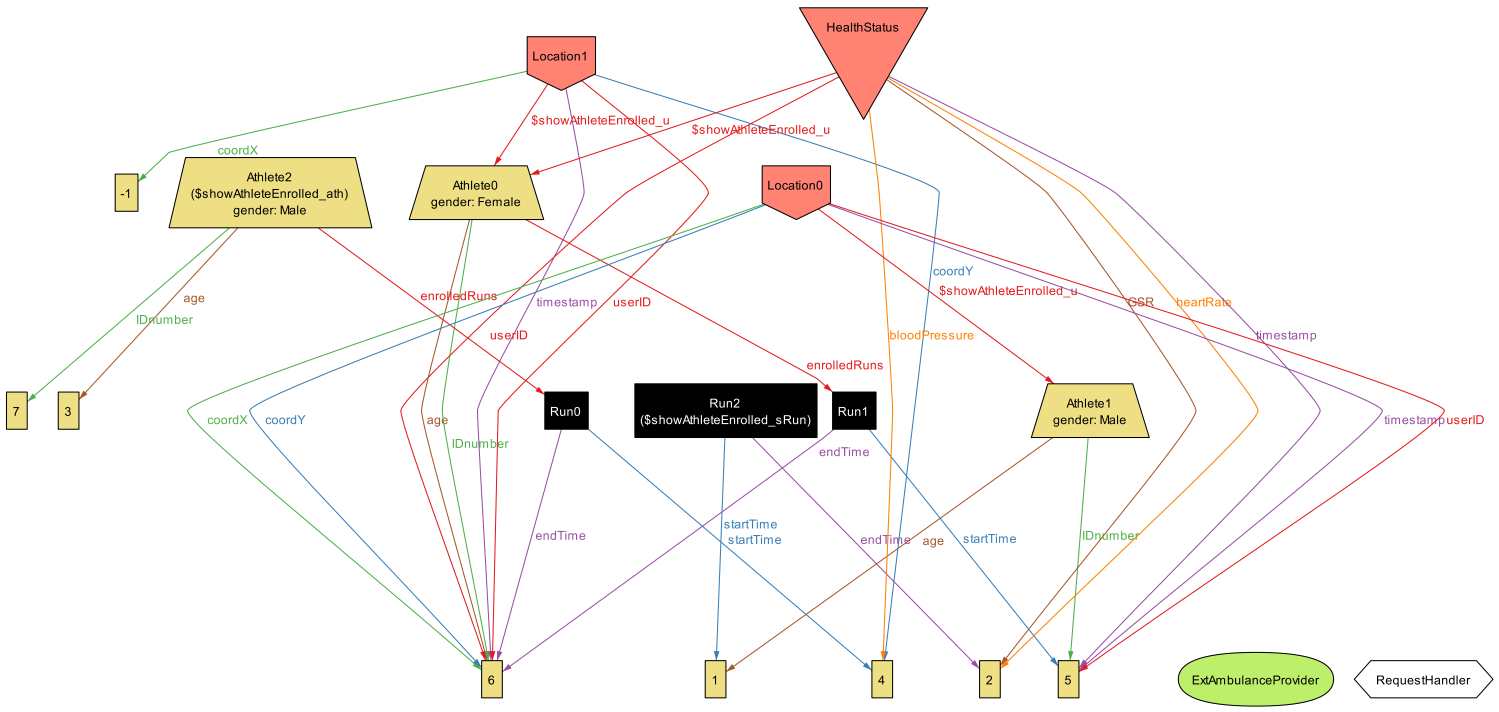
ThirdPartyWithUserData: Here we can see the ThirdParty has only access to User0's and User2's data, associated to IndividualReqPermission1 and IndividualReqPermission0 respectively, and it possesses User2's data. The ThirdParty does not have Location1, since it belongs to User1 and ThirdParty has not permission to access User1's data.



ThirdPartyWithGroupData: GroupData holds 4 instances of Data (which is greater than 3, so the data is anonymizable), and every Data in GroupData belongs to different Users.



AutomatedSOS: *HealthStatus0 represents the last HealthStatus instance of Elderly, and HealthStatus1 is the Elderly's threshold. We can see that HealthStatus0 heartRate is equal to 2, which is lower than threshold heartRate, which is 5, so the Elderly belongs to peopleToRescue set of ExtAmbulanceProvider.*



AthleteEnrolled: *Athlete2 and Athlete0 are enrolled to Run0 and Run1 respectively, whose startTime is 4 and 5. Since the value '3' represents the time value 'now', Run0 and Run1 start in the future. Instead Run2 began in the past, and Athletes cannot enroll to it.*

5 Effort Spent

5.1 Piccinotti Diego

Description of the task	**Hours**
Purpose, Scope, Definition	5.5
Product Perspective	8
Product Functions	4
User Characteristics	1
Domain Assumptions	3
Functional Requirements	12
Non-functional Requirements	0.5
Formal Analysis Using Alloy	1.5

5.2 Pietroni Umberto

Description of the task	**Hours**
Purpose, Scope, Definition	6
Product Perspective	9
Product Functions	2.5
User Characteristics	1
Domain Assumptions	3
Functional Requirements	15
Non-functional Requirements	0.5
Formal Analysis Using Alloy	2

5.3 Rossi Loris

Description of the task	**Hours**
Purpose, Scope, Definition	7
Product Perspective	6
Product Functions	2.5
User Characteristics	2
Domain Assumptions	3.5
Functional Requirements	7
Non-functional Requirements	0.5
Formal Analysis Using Alloy	11.5