POLITECNICO
MILANO 1863

Software Engineering 2
RASD

Work team:
Paolo Romeo, Andrea Scotti, Francesco Staccone

AY 2018-2019

Table of contents

1. Introduction

1.1. *Purpose*

The purpose of this Requirement Analysis and Specification Document (RASD) is to provide a detailed description of the TrackMe system.
It will explain the main features of the software system, its interfaces, what it will do, the constraints under which it must operate and how the system will react to external stimuli. This document is intended as a contractual basis for both the stakeholders and the developers of the system.

1.2. *Scope*

1.2.1. Description of the system

The TrackMe system is thought to be a Health Data Management and Run-Friendly Mobile App, useful for a quite wide range of users with different features and necessities.
The client base will be mainly composed by common users or runners interested in their health status and third parties, such as companies or associations interested in the users' data.
The system will fulfill their needs providing ad hoc services for each customer segment. For this purpose, the software will be made up of three different modules:

- Data4Help;
- AutomatedSOS;
- Track4Run.

Data4Help, a service that allows the third parties to monitor the location and the health status of the individuals. TrackMe acquires the registered users' data trough external devices connected to their smartphones.
The other two services are built on top of Data4Help and their main purpose will be to provide tangible functionalities to the users.
AutomatedSOS , an optional service that monitors the health status of the users by performing a continuous evaluation of the health parameters' data acquired by Data4Help. It sends an SOS to an entity that can send an ambulance to the location of the individual if it establishes that he/she is in danger.
Track4Run, a service thought for the organization of running events. It allows the third parties to organize races and the users to enrol to them or simply know the position of runners during a run.
The overall system should be able to provide the previously described services by taking into account the world phenomena, such as the health and the position of people, ambulances, places for runs and needs of third parties. Moreover it should observe what is controlled by the world and accordingly react by taking control over the shared phenomena which are described in the product perspective session. The features offered by the abovementioned services comply with strict requirements in the machine world and are intended to satisfy several goals in the application domain. These aspects will be further detailed in the following sections, starting from the next one.

1.2.2. Goals

The goals of the whole application are formalized below:

- G1) Third parties can monitor the position and the health status of the individuals;
- G2) Third parties can access the anonymized data of the groups of individuals;
- G3) The user can accept or refuse the requests concerning the treatment of his/her personal data by the third parties;
- G4) The third party can ask to subscribe to new data and receive them as soon as they are produced;
- G5) The user can stop the subscription of the third party to his/her data at any time;
- G6) The user can be recognized by providing a form of identification;
- G7) The third party can be recognized by providing a form of identification;
- G8) The user can check the position of the runners at any time during a race;
- G9) When the health status of the user is in danger, an SOS is launched and an ambulance is sent to the users current position;
- G10) A user can participate to the available races;
- G11) A third party can organize a race and define the path for the run;
- G12) The user can enable/disable the AutomatedSOS service at any time.
- G13) The user can connect a smartwatch or a similar device to the smartphone.

1.3. *Definitions, acronyms, abbreviations*

1.3.1. Definitions

- Third party: a company, association or, more in general, a public or private entity that uses TrackMe to acquire data of users or to organize running events;
- User: a person who uses TrackMe;
- Individual: equivalent to User;
- Danger: the status that signals that at least one health parameter of the user is out of a safety range;
- TrackMe: the name of the whole system;
- Data4Help: a service offered by TrackMe that permits to acquire the data of the users and to share them with third parties;
- AutomatedSOS: a service offered by TrackMe that raise an SOS if the health of a user is in danger;

- Track4Run: a service offered by TrackMe that permits to define, to join and to watch a run.

1.3.2. Acronyms

- RASD: Requirement Analysis and Specification Document;
- SSN: Social Security Number;
- VAT: Value Added Tax.

1.3.3. Abbreviations

- Gn: n-goal;
- Dn: n-domain assumption;
- Rn: n-functional requirement.

1.4. *Revision history*

- 09/11/2018 Version 1.0
- 10/12/2018 Version 1.1

1.5. *Reference documents*

- Specification Document: Assignments AA 2018-2019.pdf;
- Slides of the lessons;
- IEEE Std 830-1993 - IEEE Guide to Software Requirements Specifications.

1.6. *Document structure*

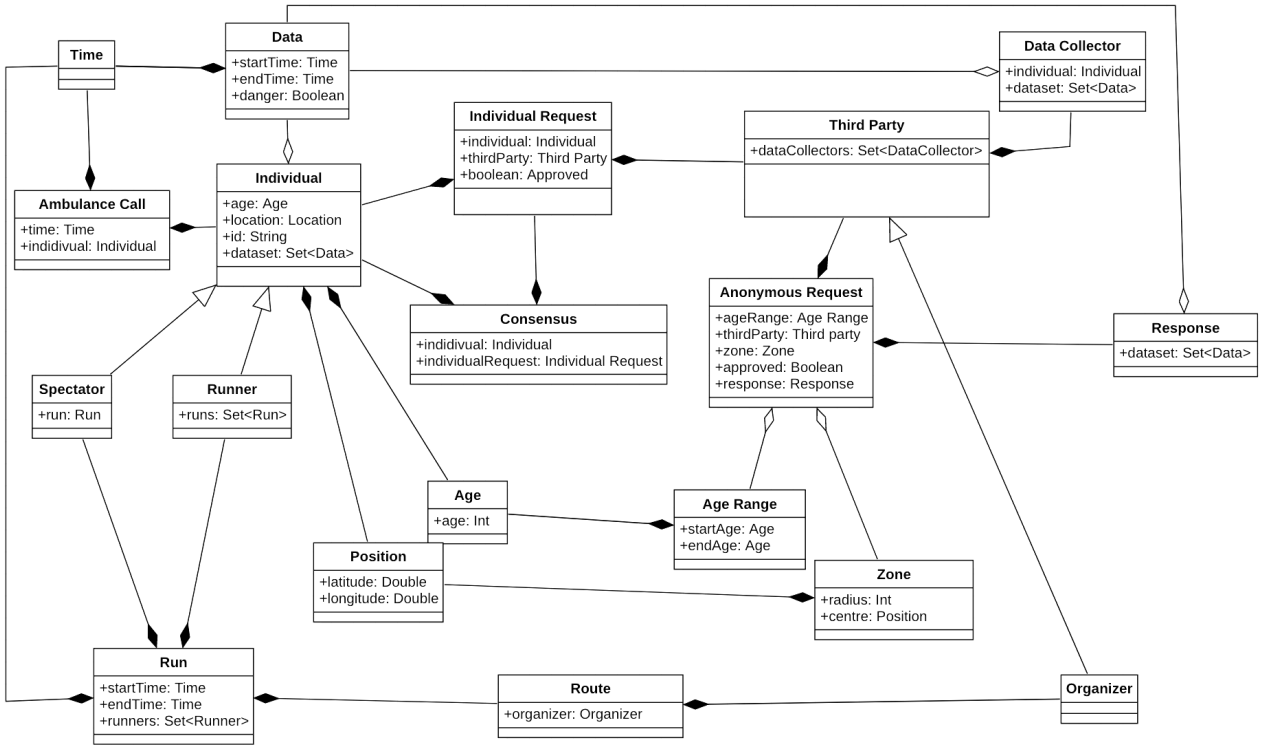
This RASD is composed by six parts:

- The first chapter is an introduction to the problem. You can find a list of the goals and other basic informations in order to better understand the whole system and the other sections of the document;
- The second part is an overall description of the system. You can find details on the shared phenomena and the domain models, a list of the most important offered functions, a brief description of the final users and the boundaries of the system;
- The third part contains the external interface requirements, including: user interfaces, hardware interfaces, software interfaces and communication interfaces. Few scenarios describing specific situations are listed here. Furthermore, the functional requirements are defined by using use case and sequence diagram. The non-functional requirements are defined through performance requirements, design constraints and software system attributes;
- The fourth section includes the alloy model and the discussion of its purpose. There are also six different worlds generated by the model;
- The fifth part presents the number of hours spent to realize this document;
- The sixth part lists all the references.

2. Overall description

2.1. *Product perspective*

The TrackMe system is thought to be a software application developed from scratch. In particular, it is intended to be used as a mobile application by both third parties and individuals. In order to work in an efficient and complete way, it exploits services and functions provided by devices that are actually external to the system itself, like the GPS localization and the health status information coming from the users smartwatch. The functions described in the next section cover the shared phenomena that the TrackMe system should be able to directly detect from the world, such as the receiving of new incoming data streams produced by the external devices or the choices made by the users on the platform. Moreover, they include the actions in the real world that the machine can directly cause, like the notifies appearing on the screen after users' interactions, or the possibility to organize a run and define a track for it.
The following class diagram describes the structure of the TrackMe system, underlining its general conceptual modelling (more details in Alloy section):



2.2. *Product functions*

In the following section, the functions of the system are listed and more precisely specified, with respect to the already mentioned main services:

- Monitor an individual

  This function permits a third party (an association, an hospital, a company, etc) to ask for the data of a single user. More in detail, the third party can select a user by his/her SSN and send him/her a request to be allowed to access his/her data. If the user accepts it, then the third party will receive his/her data. The third party can request to keep monitoring the individual also after the request and it will receive the data as soon as they are ready. The user can decide to deny the permission at any moment.

- Request for anonymous aggregated data

  Third party can access anonymized data of a group of users enrolled in TrackMe. After a third party sends this kind of request by specifying some restrictions about users attributes, like age and residence, the system will collect the data of the target users, anonymize and send them to the third party if the number of users is greater than 1000 in order to guarantee the anonymity. The third party can request to keep receveing new data of the group after the request and it will receive the data as soon as they are ready until the group size will not be lower than 1000.

- SOS assistance

  The system keeps under control the health status of a user by monitoring the values of the health parameters acquired by external devices (smart watches or similar devices). If at least one of the parameters goes under a fixed threshold, the system generates an SOS within 5 seconds starting from the moment of the evaluation of the dangerous parameter. The SOS communicates to the ambulance the position of the user.

- Organize running events

  The system offers to a third party to organize a running event. The third party must specify the timing, the track and the maximum number of participants for the run.

- Enrol to runs or follow running events

  The user can select an available run and enrol to it by sending a simple request. Furthermore the system permits a user to track the position of the runners involved in a run. The user can check the list of available runs and, after selecting one among them, he/she can watch the map of the track filled up by points representing the runners in their actual position.

2.3. *User characteristics*

Actors:

- User: a registered individual who can use the TrackMe services. He/she can login to the system and then use all the functionalities provided by the platform.

- Third party: a registered entity that can use the TrackMe services. It can login to the system and then use all the functionalities provided by the platform.

- System Manager: an employee of TrackMe, in charge of the maintenance of the system. He/she does not need to be registered, since has been inserted in the system directly during the installation process.

2.4. *Assumptions, dependencies and constraints*

2.4.1. Text assumptions

- User can choose to disable AutomatedSOS service in preferences;

- A SOS is sent through an SMS which contains the user's data and the coordinates of the position to be reached;

- If the user is not able to connect the device, he/she can contact the TrackMe assistance by calling a specific number;

- When a user signs up to TrackMe he/she accepts also to treatment of his/her data for AutomatedSOS;

- The user can remove the consensus previously given to a third party.

2.4.2. Domain assumptions

- D1) The measurements of the health status parameters of the individuals are supposed to be reliable;
- D2) The position of the individuals is supposed to be reliable;
- D3) The data acquired from the users devices are sent to the mobile application as soon as they are produced;
- D4) The identification data provided by the users are correct;
- D5) The identification data provided by the third parties are correct;
- D6) When an SOS is launched, an ambulance is sent to the position of the user linked to the account that raised the SOS itself;
- D7) The race takes place in an area with internet coverage and in a compliant track;
- D8) The location of a registered user is acquired by his smartphone used by the user himself;
- D9) Data related to the health status of a registered user are acquired by smartwatches or similar devices used by the user himself.

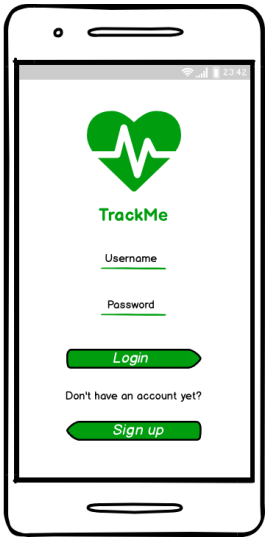
3. Specific requirements

3.1. *External interface requirements*

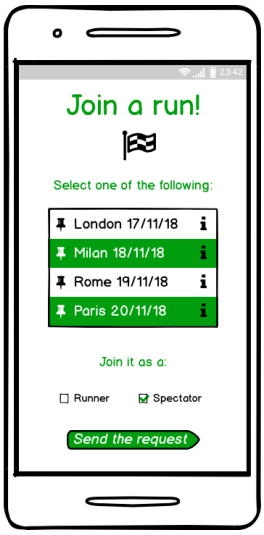
3.1.1. User interfaces

In this section follow the main mockups that represent a basic idea of what the mobile app will look like in the first release:

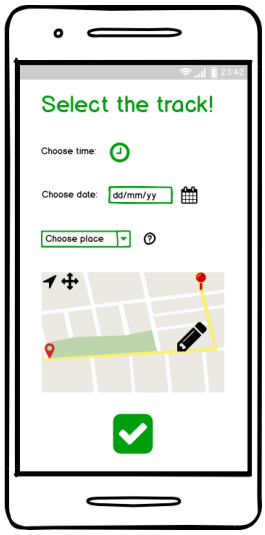
- Login or Sign Up



- (a)Join a run - (b)Select the track of the run

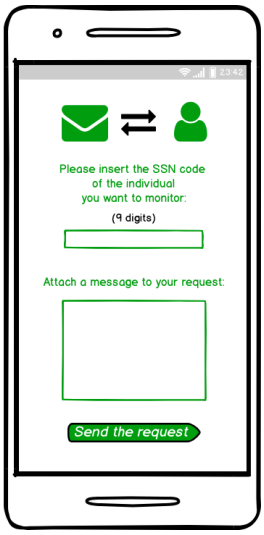


(a) Join a run

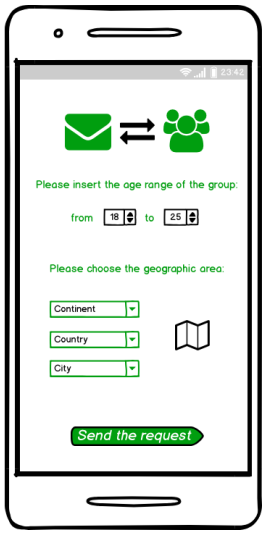


(b) Select the track of the run

- (a)Individual request - (b)Group request



(a) Individual request



(b) Group request

3.1.2. Hardware interfaces

This system is intended to stay with the user during his every day life. In order to guarantee the promised services, its necessary that users install the-front end application on a smartphone (Android or iOS) that can use GPS services to allow localization and supports Bluetooth in order to connect the device worn by the user to retrieve health data. This device can be a smartwatch or any other hardware able to get the interested health parameters. Since the application must run over the internet, the smartphone is required to connect through WIFI or a wireless mobile telecommunications technology.

3.1.3. Software interfaces

The application software doesnt need to lean on other software in order to provide basic services, except for front-end application that must run on a operating system able to connect external devices like smartwatches. Meanwhile the service Track4Run requires to interact with a software that provides a map service which includes these functionalities:

- visualize a map on which a path that cross different points can be shown;
- draw in detail this path on the map;
- provide metrics and statistics about a point moving on the map.

3.1.4. Communication interfaces

The system is essentially split into a front-end and a back-end. They communicate over the internet network. Furthermore a Bluetooth connection between smartphones and external devices is required.

3.2. *Functional Requirements*

3.2.1. Scenarios

3.2.1.1. Join a run as a runner

Mario is a sporty student who hears about a run organized by PoliMi Sport and would like to join it. PoliMi Sport is an association registered to TrackMe as a third party with the aim of guaranteeing the good health status of the enrolled students and proposing them new races every week. The unique way to join the run is to send the request to PoliMi Sport through the TrackMe app, so that Mario decides to download it and sign up. After that, he logs in with his credentials and looks for the run he is interested in. Once found, he asks to join that as a runner. Meanwhile system checks if the number of participants is not exceeded and if Mario has not already joined an overlapping run, then notifies Mario with the result.

3.2.1.2. Join a run as a spectator

Luca is a professor who is fond of running and after a lecture hears his students talking about a run organized by the student union Poli4Run. It is an association registered to TrackMe as a third party with the aim of guaranteeing the good health status of the enrolled students and proposing them new races every week. He is very interested in the race joined by their students and he would like to go and see them competing, so Luca asks them for information and decides to follow their instructions. At first he downloads the TrackMe app and then he signs up. After that, he logs in with his credentials and looks for the run he is interested in. Once found, he asks to join that as a spectator. Meanwhile he system checks if the number of spectators is not exceeded and if Luca has not already joined an overlapping run, then notifies Luca with the result.

3.2.1.3. SOS launched

Antonella is an old woman who had heart disease in the past so she would like to be daily monitored to avoid problems. Her young nephew tells her about the TrackMe app, through which she could fulfil her needs. Delighted by this kind of solution, she decides to buy a smartwatch and download the app. Then she signs up to the TrackMe app and checks to have the AutomatedSOS service correctly activated. One day, after having climbed the stairs, the app retrieves the data from her smartwatch and evaluates them as exceeding the danger thresholds, so that within 5 seconds the SOS is received by the medical facility that sends an ambulance to the position detected by the womans wearable device.

3.2.1.4. Individual request with a positive result

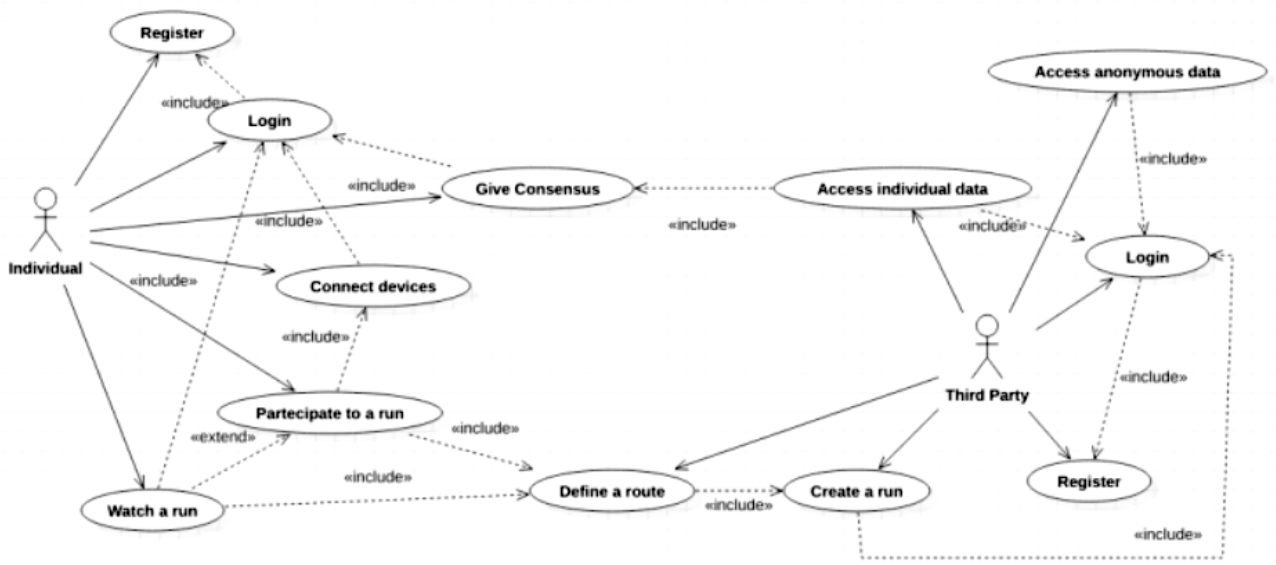
The Green Hospital wants to monitor the health status of one of its patient, Paolo, who recently recovered from an important disease. The hospital knows the mobile application TrackMe and it already has a valid account on the app. The doctor who is taking care of Paolo asks him to download the application, to link a smart device able to measure the health parameter needed by the doctor to monitor his health status and to register to the service. The doctor, using the account of the hospital, sends a request to Paolo (known by his SSN) for the treatment of his data. Paolo accepts the request and the hospital is allowed to receive the data produced by Paolos device.

3.2.1.5. Anonymous request with a negative result

The association WalkMore wants to promote walking among young people who have a too sedentary life. As a first step to reach its goal WalkMore decides to use the new amazing app Data4Help to monitor the distance that a group of target young people runs every day. WalkMore registers to Data4Help and forwards a request to access the anonymized data of people between 18 and 23 living in the municipality of Milan. Unfortunately, due to the recent release of the application, there is not enough data in order to guarantee the privacy of the selected target and the request is frozen until enough data are collected.

3.2.2. Use Case Diagram

Here follows the representation of the users interaction with the system.



3.2.3. Use Cases Descriptions

Name	Sign up
Actor	Individual
Entry Conditions	Application installed on his device.
Event Flows	1. The user opens the application. 2. The user goes to the sign up section. 3. The user gives all the necessary informations. 4. The user chooses an username and a password. 5. The user confirms his operations. 6. The system save informations.
Exit Conditions	The user is successfully registered and now he's able to log in.
Exceptions	1. The user is already registered. 2. The informations are incorrect or missing. The user is notified and invited to try again.

Name	Login
Actor	Individual
Entry Conditions	The user is previously successfully registered and has the application installed on his device.
Event Flows	1. The user opens the application. 2. The user goes to the login section. 3. The user inserts username and password. 4. The system checks if the credentials are correct.
Exit Conditions	The user is successfully logged in and he can access all the services.
Exceptions	The credentials are incorrect. The user is notified and invited to try again.

Name	Connect a device
Actor	Individual
Entry Conditions	User already logged in.
Event Flows	1. The user goes to the preference section. 2. The user chooses to connect a device and then selects the right one.
Exit Conditions	The device is successfully connected to the application.
Exceptions	The system is not able to find or connect the device. The user is notified and invited to try again by reading carefully the connection instructions or to contact the help assistance of TrackMe.

Name	Watch a run
Actor	Individual
Entry Conditions	User already logged in.
Event Flows	1. The user goes to the run section. 2. The user chooses the run he wants to watch. 3. The system screens a map with the route and the participants. 4. The user can choose to watch statistics. 5. If the user chooses to watch statistics the system provides him details about the run.
Exit Conditions	The user is able to follow a run.
Exceptions	There are no current runs. The user is notified with the list of incoming runs.

Name	Partecipate to a run
Actor	Individual
Entry Conditions	User already logged in and he has already given the consensus to the third party.
Event Flows	1. The user goes to the run section. 2. The system lists all available incoming runs which don't overlap with other user's runs. 3. The user selects an incoming run. 4. The system save the choice of the user in his calendar events.
Exit Conditions	The user is a official participant of the run and he will be notified some days before the run.
Exceptions	There are no available runs. The user is asked if he wants to be notified as soon as a new run is available.

Name	Give consensus
Actor	Individual
Entry Conditions	User already logged in and a third party has requested access to user data.
Event Flows	1. The user goes to the preference section. 2. The user chooses to give or not the consensus to the third party. 3. The system saves user's choice and sends it to the third party.
Exit Conditions	The third party can access user data if and only if the user gave the consensus.

Name	Create a run
Actor	Third Party
Entry Conditions	Third party logged in.
Event Flows	1. The third party goes to the organize a run section. 2. The system show to third party all the incoming events 3. The third party chooses to create a run. 4. The third party inserts all the relevant information except the route, it can be define later on. 5. The system saves the event and update the calendar of incoming events of all the users.
Exit Conditions	The third party is ready to define a route and all the users are informed about the run.
Exceptions	If the user hasn't given all the informations, he's asked to provide the missing ones.

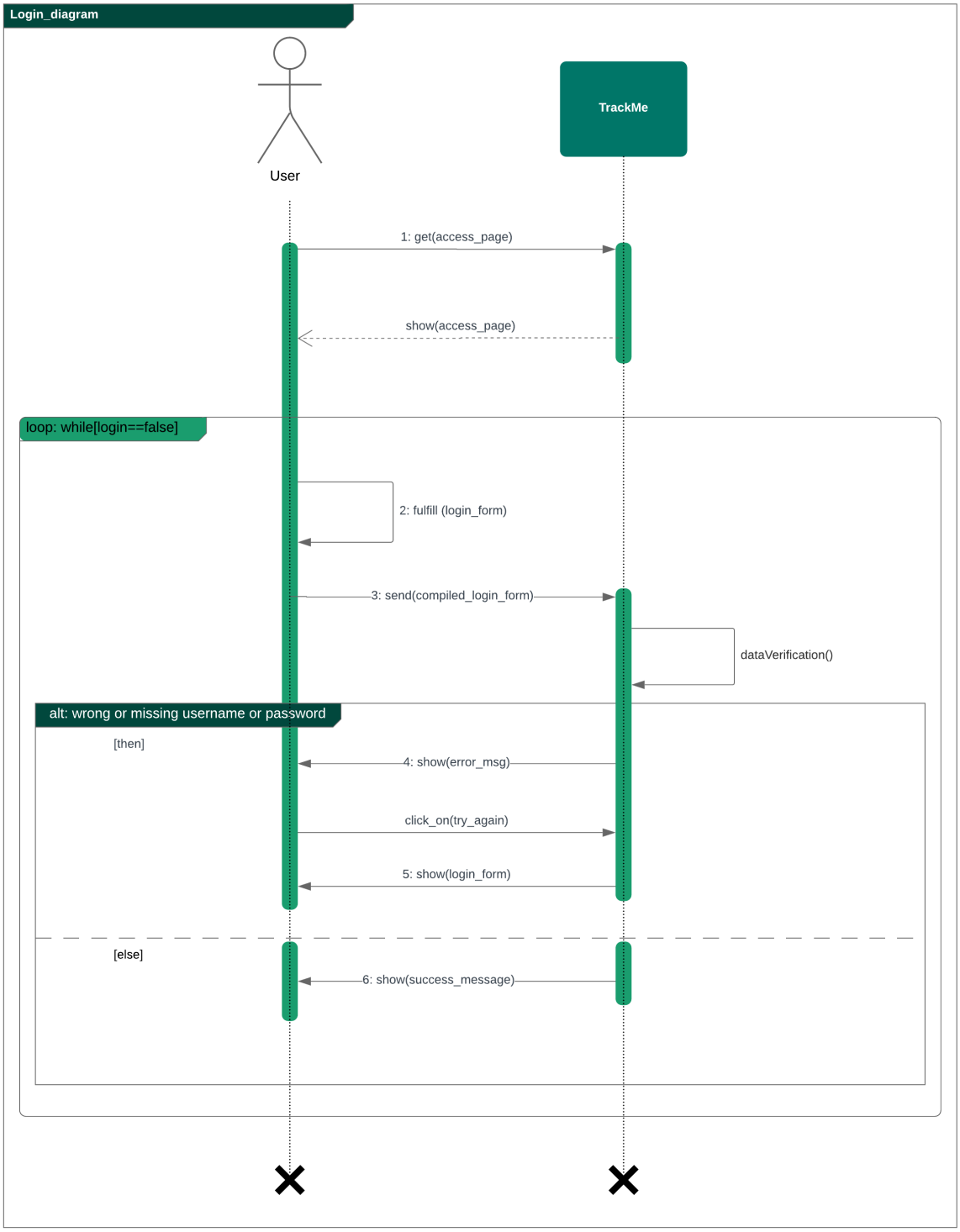
Name	Access individual data
Actor	Third Party
Entry Conditions	Third party logged in.
Event Flows	1. The third party goes to the individual request section. 2. The third party inserts the social security number of a person in order to request access of his/her data. 3. The system forwards the request to the individual.
Exit Conditions	The user is notified about the request of the third party.
Exceptions	1. The third party has already requested to access the data of the individual. In this case the third party is informed about the status of the individual's response. 2. The social security number is not correct. The third party is notified and asked to try again.

Name	Access anonymous data
Actor	Third Party
Entry Conditions	Third party logged in.
Event Flows	1. The third party goes to the group request section. 2. The third party inserts the values of attributes of the the group of which is interested in. 3. The system checks if the group is big enough to guarantee anonymity. 4. The system will send the data to third party as soon as they are ready.
Exit Conditions	The third party has got access to the data of the interested group.
Exceptions	If the group is not big enough, the third party is notified and asked to try with other values of attributes.

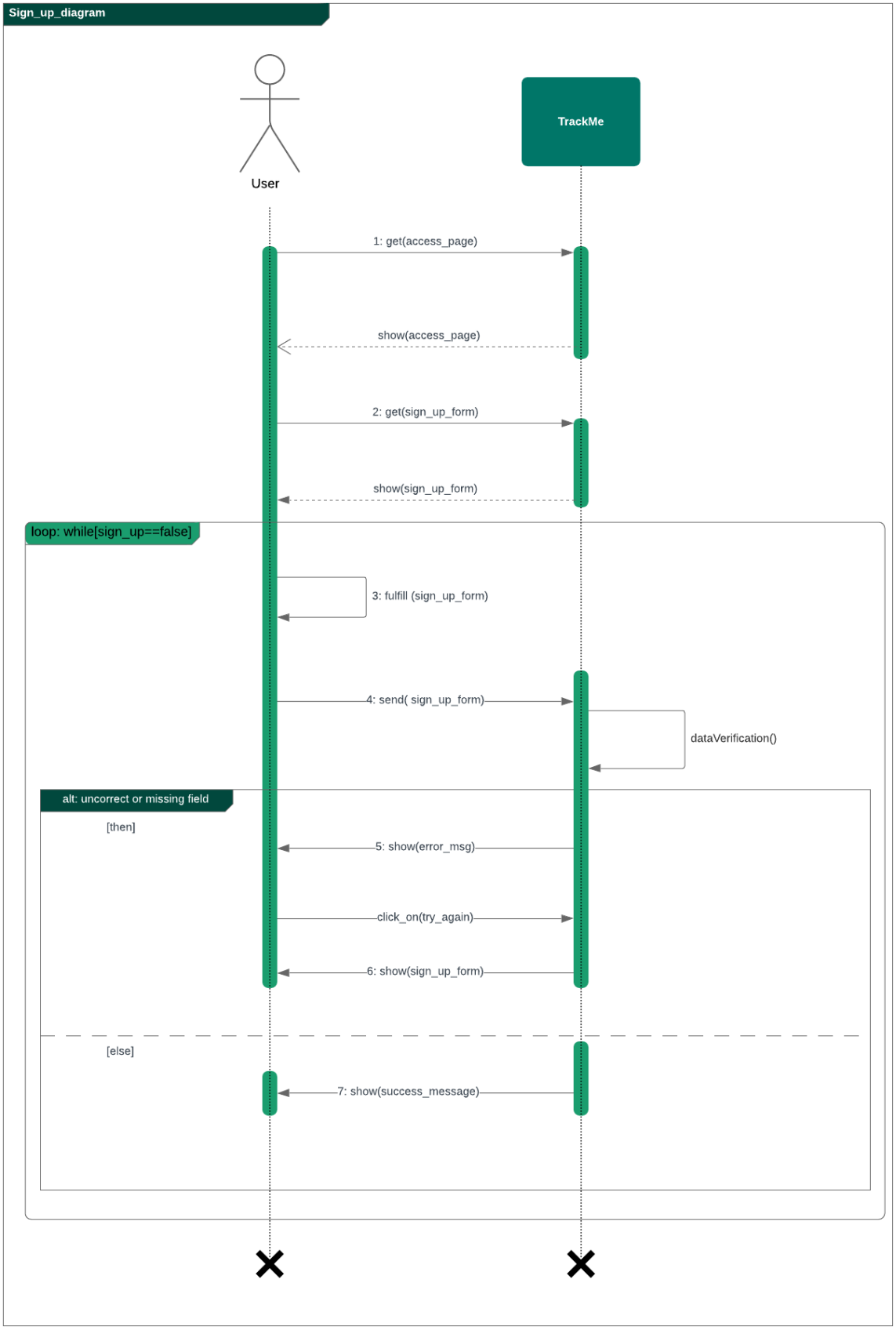
3.2.4. Diagrams

In this section follow the interaction diagrams that detail how the main TrackMe operations are carried out by the software system:

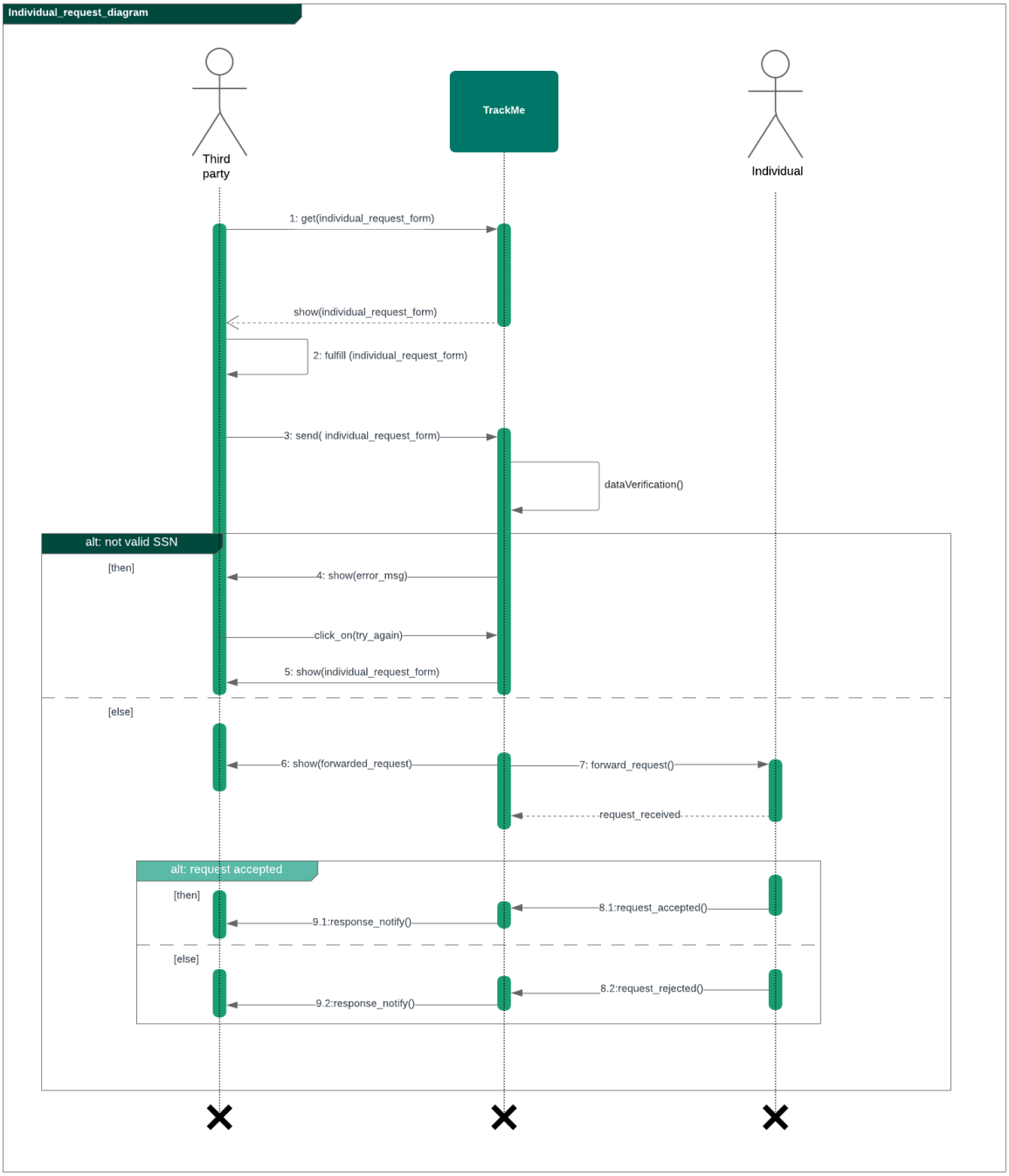
3.2.4.1. Login - *Sequence diagram*



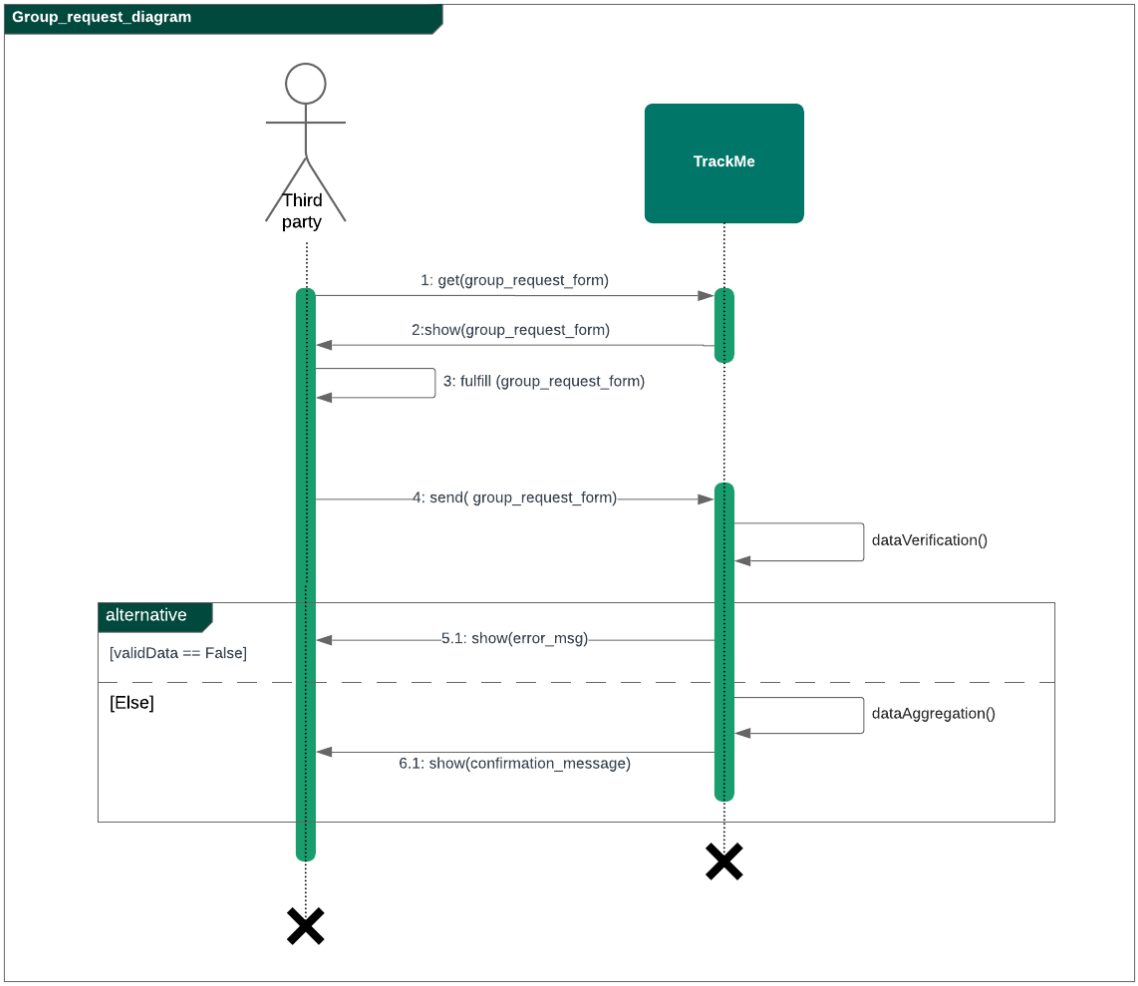
3.2.4.2. Sign up - *Sequence diagram*



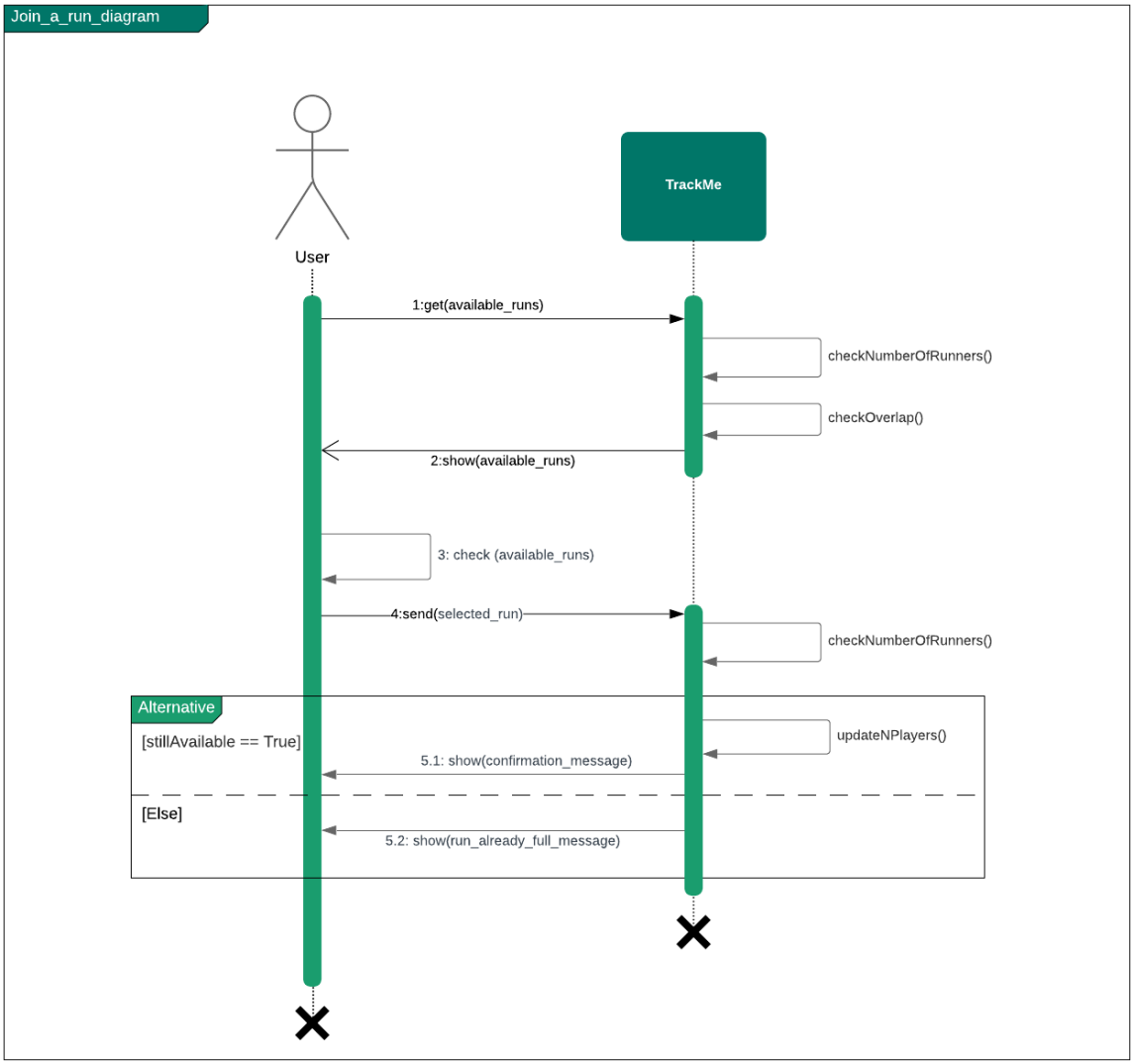
3.2.4.3. Individual request - *Sequence diagram*



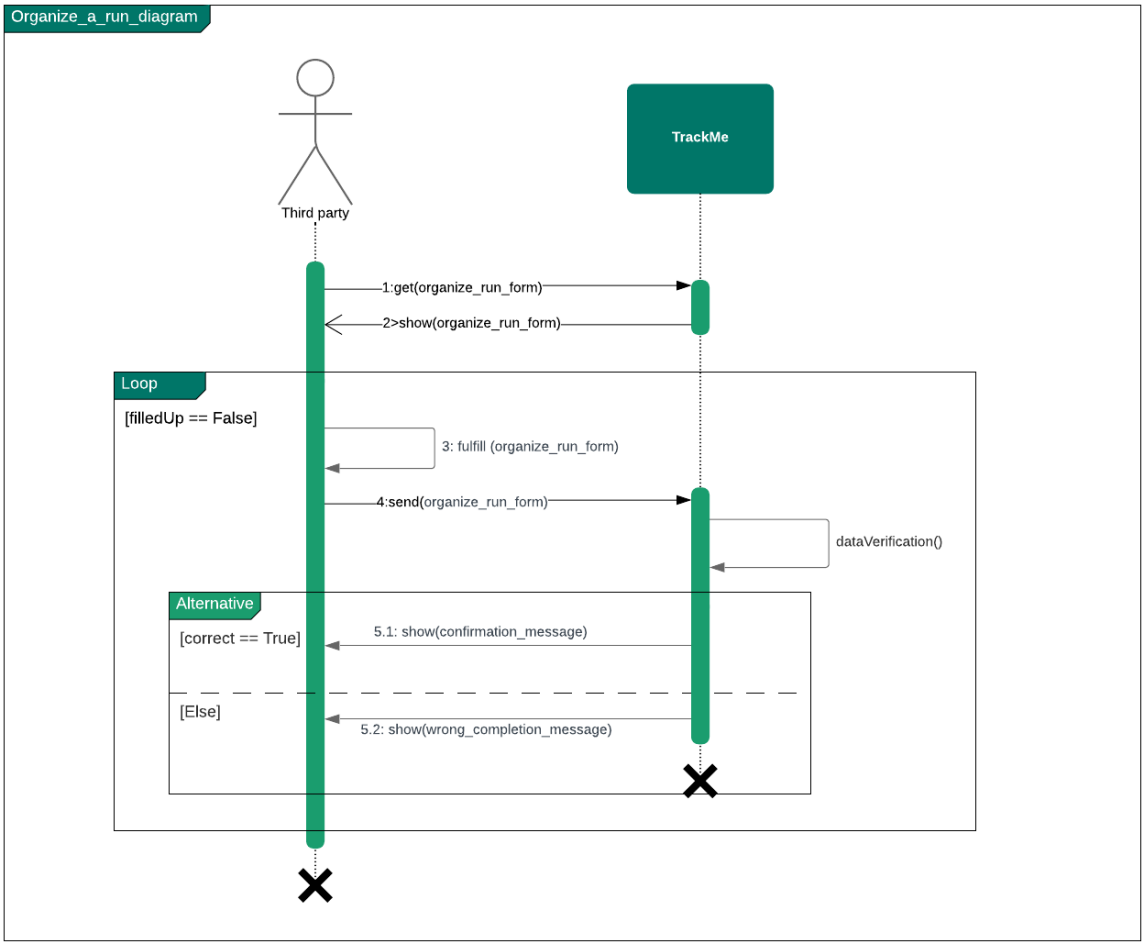
3.2.4.4. Group request - *Sequence diagram*



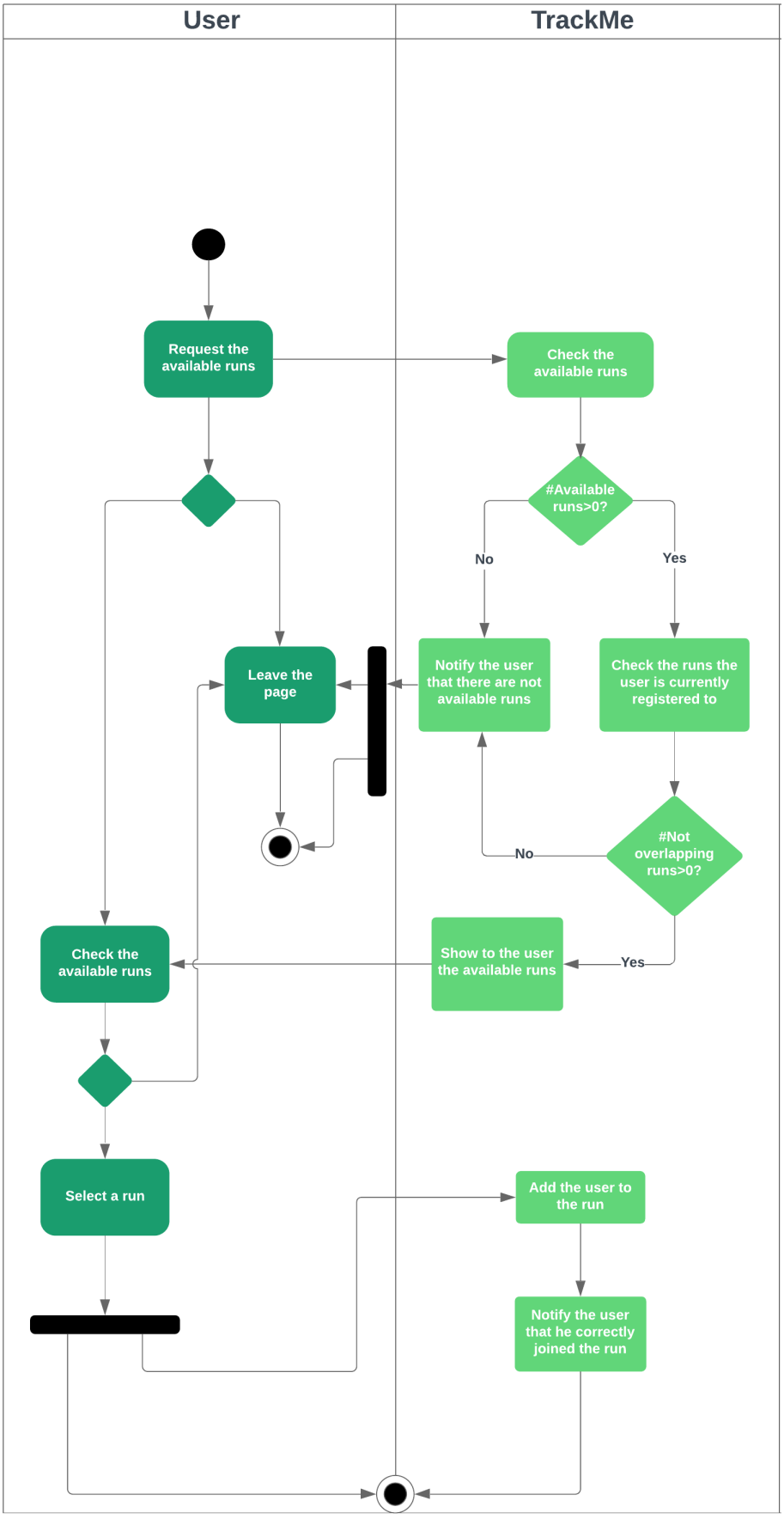
3.2.4.5. Join a run - *Sequence diagram*



3.2.4.6. Organize a run - *Sequence diagram*



3.2.4.7. Join a run - *Activity diagram*



3.2.5. Global requirements

Here are listed the requirements which are needed to satisfy the majority of the goals:

- Third parties and individuals must have registered and logged in to use the services;

- The app is able to acquire data and to share data with the servers also in background in users' smartphones;

- There must be a fully working connection between the application and the smartwatches or similar devices.

3.2.6. Mapping on requirements

- G1) Third parties can monitor the position and the health status of the individuals;
  - D1) The measurements of the health status parameters of the individuals are supposed to be reliable;
  - D2) The position of the individuals is supposed to be reliable;
  - D8) The location of a registered user is acquired by his smartphone used by the user himself;
  - D9) Data related to the health status of a registered user are acquired by smartwatches or similar devices used by the user himself;
  - R1) The users must have given the consensus to the treatment of their information to the third party;
  - R2) The system must be able to provide to the third party the location and the health status of individuals;
  - R3) The system must be able to retrieve data from the smartwatches and similar devices;
  - R27) The system must be able to store data retrieved from registered users;
- G2) Third parties can access the anonymized data of the groups of individuals;
  - D1) The measurements of the health status parameters of the individuals are supposed to be reliable;
  - D8) The location of a registered user is acquired by his smartphone used by the user himself;
  - D9) Data related to the health status of a registered user are acquired by smartwatches or similar devices used by the user himself;
  - R3) The system must be able to retrieve data from the smartwatches and similar devices;
  - R4) The groups must be composed at least by 1000 individuals;
  - R5) The system must be able to provide to the third party the health status of individuals in an anonymous way;
  - R6) The system must be able to aggregate the data of the individuals, as requested by the third party;
  - R27) The system must be able to store data retrieved from registered users.

- G3) The user can accept or refuse the requests concerning the treatment of his/her personal data by the third parties;
  - R7) The system must be able to forward the requests from the third party to the user;
  - R8) The system must save the preference of the user;
  - R9) The third party is not allowed to access the users data until he/she accepts the request.
- G4) The third party can ask to subscribe to new data and receive them as soon as they are produced;
  - D3) The data acquired from the users devices are sent to the mobile application as soon as they are produced;
  - D9) Data related to the health status of a registered user are acquired by smartwatches or similar devices used by the user himself;
  - R11) The system is optimized to send the data received from the mobile application to the third parties as soon as possible.
- G5) The user can stop the subscription of the third party to his/her data at any time;
  - R28) The user must have an active subscription to stop it;
  - R29) The system must be able to allow the user to unsubscribe to the third party and to stop the transmission of his/her data.
- G6) The user can be recognized by providing a form of identification;
  - D4) The identification data provided by the users are correct;
  - R12) The users must provide their personal data to the application during the registration process, SSN (or fiscal code) included;
  - R13) The user can register to the application by selecting a username and a password;
  - R14) The system must allow the user to log in to the application by providing the combination of a username and a password that matches an account;;
  - R15) Two different users cannot have the same username.
  - R31) The system must allow the user to change his/her personal info.
  - R32) The system must allow the user to change his/her password.
- G7) The third party can be recognized by providing a form of identification;
  - D5) The identification data provided by the third parties are correct;
  - R16) The system must allow the third party to register to the application, by specifying its VAT registration number and a password;
  - R17) The system must allow the third party to log in to the application by providing the combination of a VAT registration number and a password that match an account;

  - R33) The system must allow the Third party to change its password.

- G8) The user can check the position of the runners at any time during a race;
  - D2) The position of the individuals is supposed to be reliable;
  - D8) The location of a registered user is acquired by his smartphone used by the user himself;
  - R19) The system must be able to retrieve the position of all the runners;
  - R20) The system must be able to provide the position of all the runners in the track in real time.

- G9) When the health status of the user is in danger, an SOS is launched and an ambulance is sent to the users current position;
  - D6) When an SOS is launched, an ambulance is sent to the position of the user linked to the account that raised the SOS itself;
  - D8) The location of a registered user is acquired by his smartphone used by the user himself;
  - D9) Data related to the health status of a registered user are acquired by smartwatches or similar devices used by the user himself;
  - R18) When the health status values go below the threshold, the system must send an SOS within 5 seconds;
  - R25) The AutomatedSOS service must be enabled.

- G10) A user can participate to the available races.
  - R21) The system must allow the user to check the list of available races at any time.
  - R22) The system must allow the user to join an available race only before its starting time.
  - R23) The user cannot join two different overlapping races.
  - R30) The system must avoid the registration of users after having reached of the maximum number of participants.

- G11) A third party can organize a race and define the path for the run;
  - D7) The race takes place in an area with internet coverage and in a compliant track;
  - R24) The system must allow the third party to organize a race by defining its track and its time.

- G12) The user can enable/disable the AutomatedSOS service at any time.
  - R26) The system must allow the user to enable/disable the AutomatedSOS service at any time.

- G13) The user can connect a smartwatch or a similar device to the smartphone.
  - R34) The system must allow the user to connect a smartwatch or a similar device to its smartphone.

3.3. *Performance requirements*

The system has to be able to respond to a possibly great number of simultaneous requests and it must continuously acquire data from all the users' smartphones. The idea is to implement a system able to manage thousands of users. It also must be flexible in order to face fast increases in the number of users.
Third parties will use the data provided by us as part of their business processes, users' health can be preserved by our app and a large amount of events can be organized trough our application at the same time, so TrackMe have to guarantee quick, reactive and correct responses.

3.4. *Design constraints*

3.4.1. Standards compliance

- The app requests the users' permission to manage their health status data and their location to support core functionalities.
- The app supports portrait orientation.
- The app preserves app state when leaving the foreground and prevents accidental data loss due to back-navigation.
- Users' private data are stored in the app's internal storage (this is very useful for connection lost).

3.4.2. Hardware limitations

The main HW limitations regard the smartphone used by the customer. It has to provide:
- iOS or Android compatibility
- 2G/3G/4G connection
- GPS

3.5. *Software System Attributes*

3.5.1. Reliability

The system is not fault tolerant, so it will be reliable according to its ability to avoid process crashes and to scale in case of numerous accesses.

3.5.2. Availability

The system is not fault tolerant, so in case of failure it will not be available until the administrator will be able to reboot the system. So for this system availability is strictly correlated to maintainability.

3.5.3. Maintainability

The code should be written in a way that facilitates the testability and the implementation of new functionalities.

3.5.4. Scalability

The right choice of a PaaS allows the system to increase or decrease the level of allocated virtual resources at application or database on demand.

3.5.5. Portability

The system front-end will be a web app, runnable on the majority of browsers but also natively on Android and IOS.

3.5.6. Safety

The communication between the client and server is protected through a protocol such as HTTPS. The principal motivation is authentication of the accessed website and protection of the privacy and integrity of the exchanged data while in transit. It protects against man-in-the-middle attacks. The bidirectional encryption of communications between a client and server protects against eavesdropping and tampering of the communication. In practice, this provides a reasonable assurance that one is communicating without interference by attackers with the website that one intended to communicate with, as opposed to an impostor.

4. Formal analysis using Alloy

The alloy model has been thought to be coherent with the class diagram. In order to generate worlds not much complex, we decided to split the code in three parts, one for each service, by replicating the common signatures and facts. We used small integers to represent numbers in order to allow Alloy software to generate instances.

- AutomatedSOS:
  Individual are characterized by batches of data. If one of them contains a value which overcomes a predetermined threshold, an ambulance call associated to that individual should exists. Here we check if the ambulance call is made fast by considering that the sum among the maximum length in time of a batch of data and the time needed to process this batch is lower than the required value.
- Data4Help:
  Third parties can make individual requests and anonymous requests. If a request is accepted, in the first case, individual data should appear in a data collector owned by the third party, meanwhile in the second case data should compare in a response associated to the request (we don't formalized this last aspect).
- Track4Run:
  For this service it's only formalized that two runners cannot choose two runs which are overlapping and that a run has got a maximum number of partecipants.

```
-- Let's assume that data of individuals, time, latitude and longitude are integers.
open util/integer as integer
open util/boolean as boolean

one sig Threshold{
	threshold: one Int
}
{threshold = 4}

one sig ReactionTime{
	time: one Int
}
{time = 3}

sig Time{
	time: one Int
}
{time >= 0 and time <= 5}

-- Atomic piece of data
sig Value{
	value: one Int
}
{value >=0 and value <=5}

-- Batch of data of some individual during a fixed range of time
sig Data{
	startTime: one Time,
	endTime: one Time,
	-- Array of values
	values: seq Value,
	-- True if and only if exists a value in values that is higher than the threshold
	danger: one Bool
}
-- Maximum batch slot duration is 2 unit of time
{startTime.time < endTime.time and #values > 0 and #values <= 5 and sub[endTime.time, startTime.time] <= 2 and (danger = True iff
 some i : values.inds | values[i].value > Threshold.threshold)}

sig Individual{
	--age: one Age,
	--position: one Position,
	data: set Data
}
{ #data > 0 and #data <= 3}
```

```
fact noDataWithoutInvididual{
	no d : Data | (all ind : Individual | d not in ind.data)
}

fact dataHasOnlyOneIndividual{
	no d : Data | some disj ind1, ind2 : Individual | d in ind1.data and d in ind2.data
}

-- Batches of data refer to a specific and unique range of time
fact noOverlappingSequenceOfDataForOneIndividual{
	all ind : Individual | no disj d1, d2 : Data | d1 in ind.data and d2 in ind.data and d1.startTime.time <= d2.startTime.time and
 d1.endTime.time >= d2.startTime.time
}

--AUTOMATEDSOS

sig AmbulanceCall{
	individual: one Individual,
	time: one Time
}

-- We consider only the case in which just an ambulance is sent for individual
fact atMostOnlyOneAmbulance{
	no disj c1, c2 : AmbulanceCall | c1.individual = c2.individual
}

-- The ambulance is called if and only if there is a value which is higher than the threshold. We assume 1 unit of time is needed
 to process the data
fact callAmbulanceIfAndOnlyIfNeeded{
	(all c : AmbulanceCall | some d : Data | d in c.individual.data and d.danger = True and d.startTime.time < c.time.time) and
 (all ind : Individual | all d : Data | d in ind.data and d.danger = True implies some c : AmbulanceCall | c.individual = ind an
d c.time.time > d.endTime.time and sub[c.time.time, d.endTime.time] <= 1 )
}

-- Check if the call of the Ambulance is done within the reaction time (this is assured by the boundaries on the duration of the batch
 slots and on the time to process the data)
assert fastAmbulanceCall{
	--all t : Time | t.time = 0
	all ind : Individual | all d : Data | d in ind.data and d.danger = True implies some c : AmbulanceCall | c.individual = ind and
 sub[c.time.time, d.startTime.time] <= ReactionTime.time

}

pred show{}
--run show for 3 but 2 Individual, 1 AmbulanceCall
check fastAmbulanceCall for 1 but 6 Data, 10 Time, 2 Individual, 4 Value
```

```
-- Let's assume that data of individuals, time, latitude and longitude are integers.
open util/integer as integer
open util/boolean as boolean

one sig Threshold{
	threshold: one Int
}
{threshold = 4}

one sig GroupSize{
	size: one Int
}
{size = 2}

sig Time{
	time: one Int
}
{time >= 0 and time <= 5}

-- Current position of the individual
sig Position{
	lat: one Int, -- y
	lon: one Int -- x
}
{ lat >= 0 and lat <= 5 and lon >= 0 and lon <= 5}

sig Age{
	age: one Int
}
{age >= 0 and age <= 5}

-- Atomic piece of data
sig Value{
	value: one Int
}
{value >=0 and value <=5}

-- Batch of data of some individual during a fixed range of time
sig Data{
	startTime: one Time,
	endTime: one Time,
	-- Array of values
	values: seq Value,
	-- True if and only if exists a value in values that is higher than the threshold
	danger: one Bool
}
-- Maximum batch slot duration is 2 unit of time
{startTime.time < endTime.time and #values > 0 and #values <= 5 and sub[endTime.time, startTime.time] <= 2 and
(danger = True iff some i : values.inds | values[i].value > Threshold.threshold)}

sig Individual{
	age: one Age,
	position: one Position,
	data: set Data
}
{ #data >= 0 and #data <= 3}

fact noDataWithoutInvididual{
	no d : Data | (all ind : Individual | d not in ind.data)
}

fact dataHasOnlyOneIndividual{
	no d : Data | some disj ind1, ind2 : Individual | d in ind1.data and d in ind2.data
}

-- Batches of data refer to a specific and unique range of time
fact noOverlappingSequenceOfDataForOneIndividual{
	all ind : Individual | no disj d1, d2 : Data | d1 in ind.data and d2 in ind.data and d1.startTime.time <= d2.startTime.time
and d1.endTime.time >= d2.startTime.time
}

-------------------------------------------------------------------------------------------------------------------------------------------------------
```

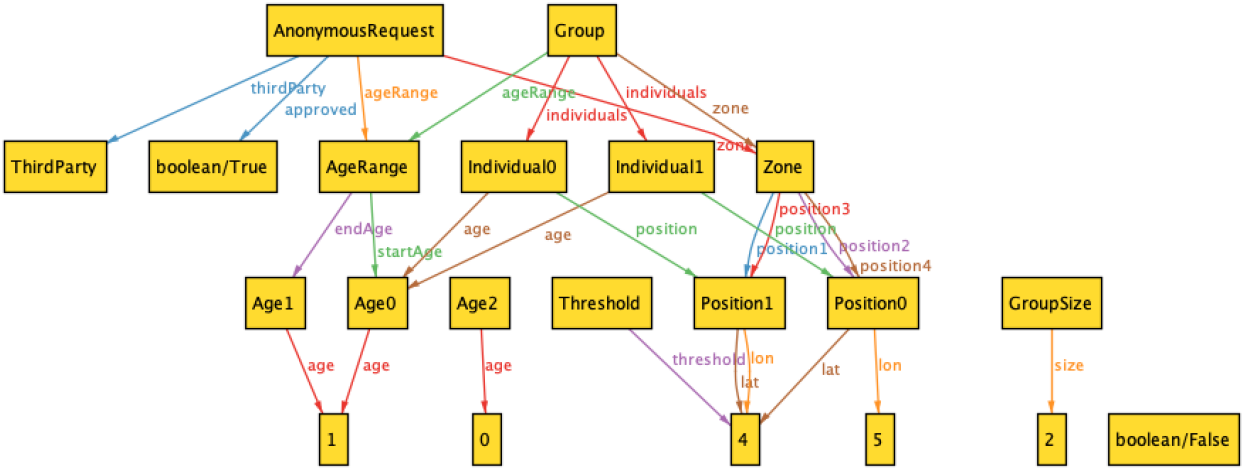
```
--DATA4HELP

sig ThirdParty{
	dataCollectors: set DataCollector
}
-- One DataCollector for individual
{no disj d1, d2 : dataCollectors | d1 in dataCollectors and d2 in dataCollectors and d1.individual = d2.individual}

sig DataCollector{
	individual: one Individual,
	data: set Data
}
{data = individual.data and no disj t1, t2: ThirdParty | this in t1.dataCollectors and this in t2.dataCollectors}
-- DataCollector can be or empty or filled with all the data owned by individual, we assume in this case that the passage of each
Data is istantaneous

sig IndividualRequest{
	thirdParty: one ThirdParty,
	individual: one Individual,
	approved: one Bool
}

sig AnonymousRequest{
	thirdParty: one ThirdParty,
	ageRange: lone AgeRange,
	zone: lone Zone,
	approved: one Bool,
	--response: lone AnonymousResponse
}
-- There is no response only in the case the request is not approved and it's approved when the anonymous group is enough big.
{/*(response = none iff approved = False) and */(approved = True iff some g: Group | g.ageRange = ageRange and g.zone = zone
and #g.individuals >= GroupSize.size)}

-- For semplicity we decided to not check if this response contains the correct data
/*sig AnonymousResponse{
	data: set Data
}*/

sig AgeRange{
	startAge: one Age,
	endAge: one Age
}

sig Group{
	ageRange: lone AgeRange,
	zone: lone Zone,
	individuals: set Individual
}
-- All individual inside individuals have got the correct charateristics to be inside this group and there is no individual with these
charateristics who is not inside the group
{(all ind: Individual | ind in individuals implies checkPositionInsideZone[zone, ind.position] and checkAgeInsideRange[ageRange, ind.age])
and (all ind: Individual | checkAgeInsideRange[ageRange, ind.age] and  checkPositionInsideZone[zone, ind.position] implies ind in individuals)}

-- 12
-- 34
sig Zone{
	position1: one Position,
	position2: one Position,
	position3: one Position,
	position4: one Position,
}
{ position1.lat = position2.lat and position3.lat = position4.lat and position1.lon = position3.lon and position2.lon = position4.lon}

pred checkPositionInsideZone[z: Zone, p: Position]{
	not(p.lat > z.position2.lat or p.lat < z.position3.lat or p.lon < z.position3.lon or p.lon > z.position2.lon)
}

pred checkAgeInsideRange[r: AgeRange, a: Age]{
	not (a.age < r.startAge.age or a.age > r.endAge.age)
}

fact noDataCollectorWithoutThirdParty{
	no d : DataCollector | all t : ThirdParty | d not in t.dataCollectors
}

--The third party can access the data only if the request has been approved by the system
fact accessDataIfAndOnlyIfConsensusIsGiven{
	all r : IndividualRequest | r.approved = False iff no d : DataCollector | d in r.thirdParty.dataCollectors and d.individual = r.individual
}
```

```
pred show1{
	AnonymousRequest.approved = True
}
pred show2{
	AnonymousRequest.approved = False
}
pred show3{
	IndividualRequest.approved = True
}
pred show4{
	IndividualRequest.approved = False
}
run show1 for 3 but 1 AnonymousRequest, 0 IndividualRequest, 2 Individual, 0 DataCollector, 1 ThirdParty, 1 Zone, 1 AgeRange, 1 Group, 0 Value, 0 Data, 0 Time
run show2 for 3 but 1 AnonymousRequest, 0 IndividualRequest, 2 Individual, 0 DataCollector, 1 ThirdParty, 1 Zone, 1 AgeRange, 1 Group, 0 Value, 0 Data, 0 Time
run show3 for 2 but 0 AnonymousRequest, 0 Zone, 0 AgeRange, 1 IndividualRequest, 1 Individual, 1 Age, 1 Position, 1 DataCollector
run show4 for 2 but 0 AnonymousRequest, 0 Zone, 0 AgeRange, 1 IndividualRequest, 1 Individual, 1 Age, 1 Position, 1 DataCollector
```

```
-- Let's assume that data of individuals, time, latitude and longitude are integers.
open util/integer
--open util/boolean as boolean

sig Time{
	time: one Int
}
{time >= 0 and time <= 5}

--TRACK4RUN

sig Run{
	startTime: one Time,
	endTime: one Time,
	runners: set Runner,
	maxSize: one Int
}
{startTime.time < endTime.time and #runners <=  maxSize and maxSize <= 5 and no disj r1, r2 : Runner | r1
in runners and r2 in runners and r1 = r2}

sig Runner{
	runs: set Run
}
{no disj r1, r2 : Run | r1 in runs and r2 in runs and r1 = r2}

fact runnerInRunIfRunnerInRun{
	all race : Run | all r : Runner | r in race.runners implies race in r.runs
}

fact runInRunnerIfRunInRunner{
	all r : Runner | all race : Run | race in r.runs implies r in race.runners
}

-- The runner cannot partecipate to different runs which are hold in the same time
fact noOverlappingRunForRunner{
	all runner : Runner | no disj r1, r2 : Run | r1 in runner.runs and r2 in runner.runs and r1.startTime.time
<= r2.startTime.time and r1.endTime.time >= r2.startTime.time
}

pred show[r1: Runner, r2: Runner, race1: Run, race2: Run]{
	r1 != r2 and race1 != race2
	addRunnerToRun[r1, race1]
	addRunnerToRun[r2, race1]
	addRunnerToRun[r1, race2]
}



pred addRunnerToRun[r: Runner, race: Run]{
	r in race.runners
}

run show for 4 but 2 Run, 2 Runner
```

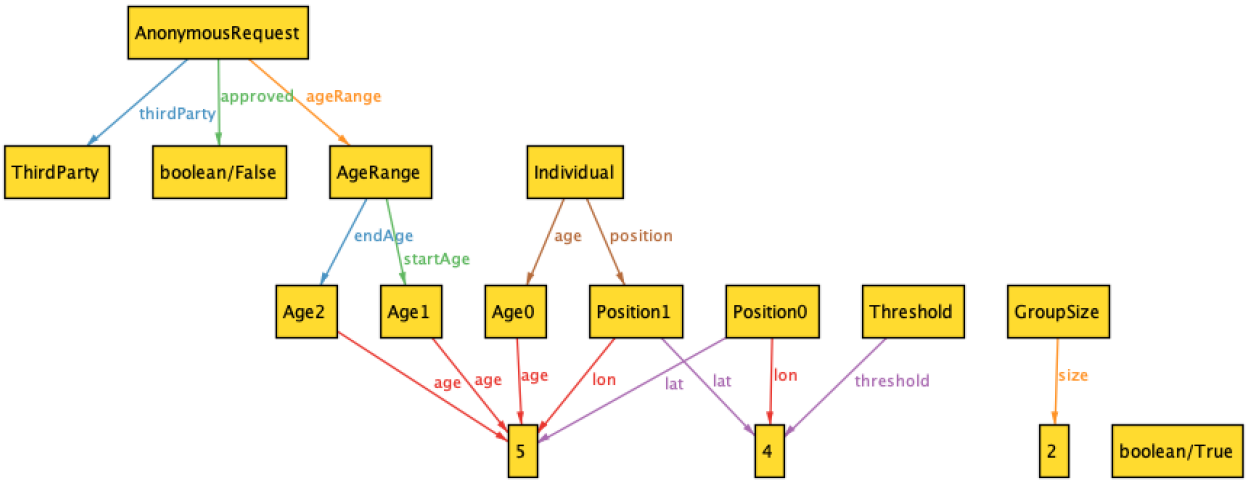
- Anonymous request accepted

```
Executing "Run show1 for 3 but 1 AnonymousRequest, 0 IndividualRequest,
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  5960 vars. 228 primary vars. 17077 clauses. 407ms.
  Instance found. Predicate is consistent. 206ms.
```



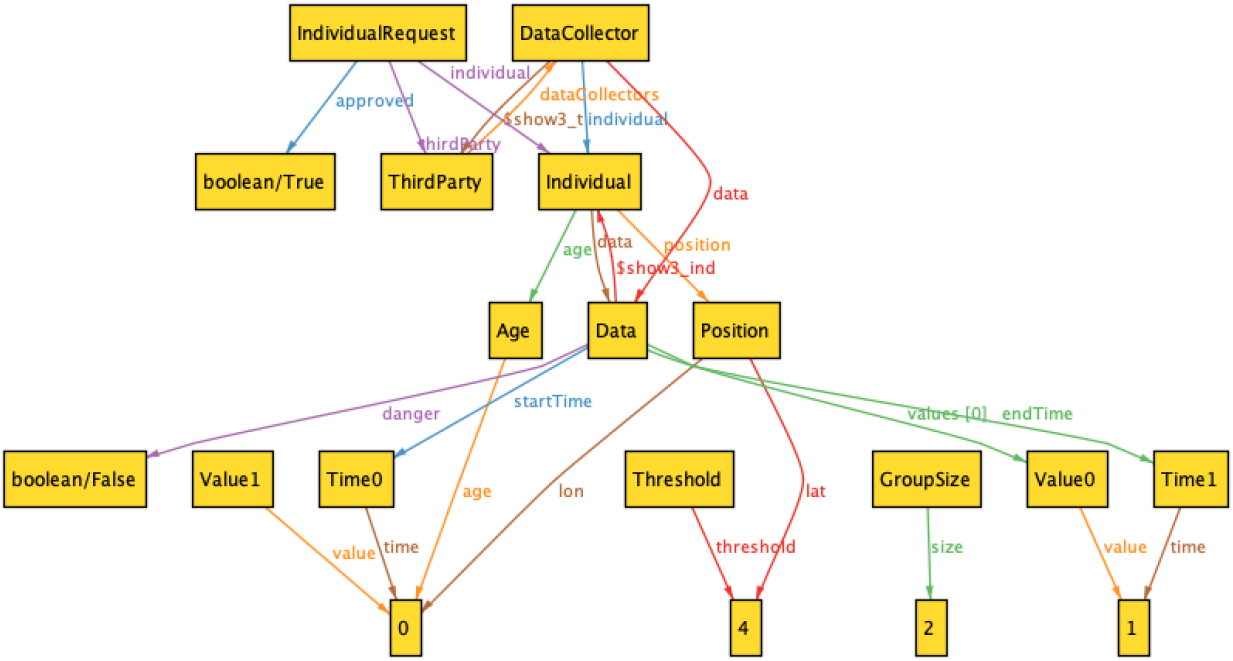
- Anonymous request refused

```
Executing "Run show2 for 3 but 1 AnonymousRequest, 0 IndividualRequest,
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  5960 vars. 228 primary vars. 17077 clauses. 182ms.
  Instance found. Predicate is consistent. 56ms.
```



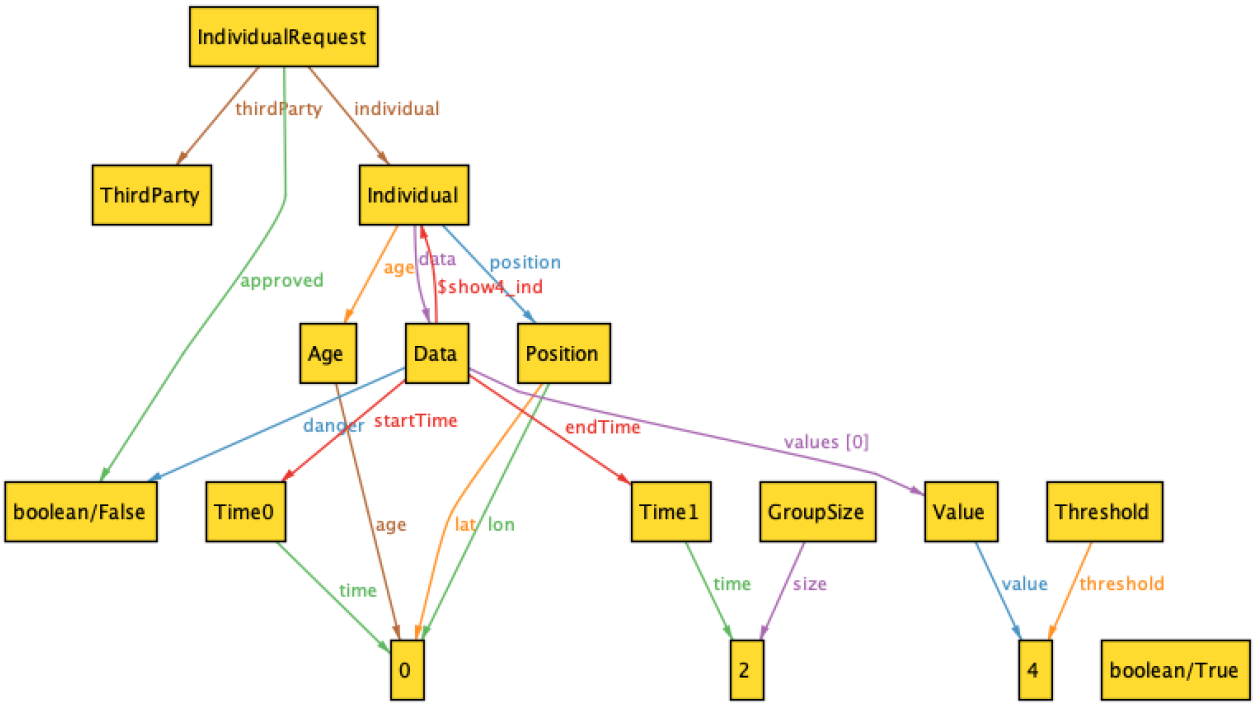
- **Individual request accepted**

```
Executing "Run show3 for 2 but 0 AnonymousRequest, 0 Zone, 0 AgeRange,
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4345 vars. 199 primary vars. 12035 clauses. 357ms.
  Instance found. Predicate is consistent. 163ms.
```



- **Individual request refused**

```
Executing "Run show4 for 2 but 0 AnonymousRequest, 0 Zone, 0 AgeRange,
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  4342 vars. 199 primary vars. 12028 clauses. 372ms.
  Instance found. Predicate is consistent. 239ms.
```

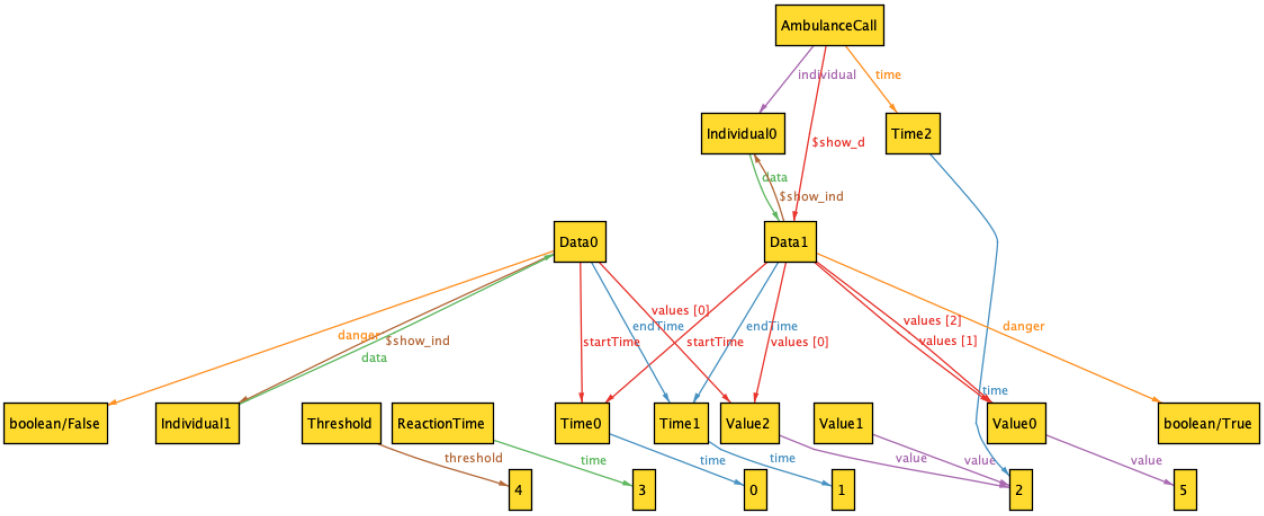


- Check assertion on AutomatedSOS

```
Executing "Check fastAmbulanceCall for 1 but 6 Data, 10 Time, 2 Individual,
  Solver=sat4j Bitwidth=4 MaxSeq=1 SkolemDepth=1 Symmetry=20
  14445 vars. 485 primary vars. 38577 clauses. 1013ms.
  No counterexample found. Assertion may be valid. 340ms.
```

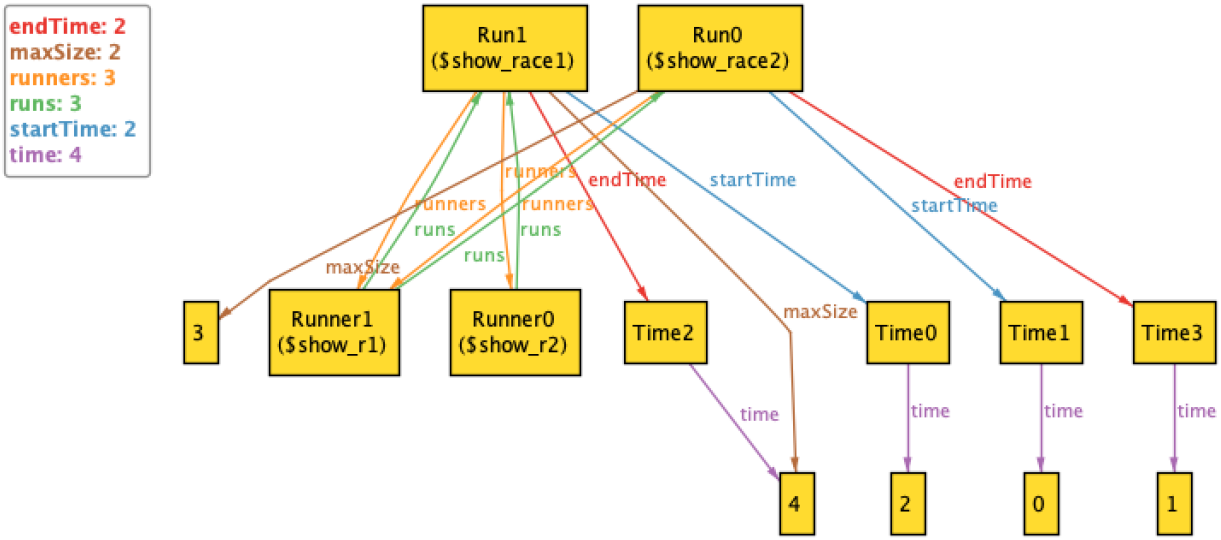
- Ambulance call

```
Executing "Run show for 3 but 2 Individual, 1 AmbulanceCall"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  6825 vars. 211 primary vars. 18870 clauses. 475ms.
  Instance found. Predicate is consistent. 499ms.
```



- **Track4Run**

```
Executing "Run show for 4 but 2 Run, 2 Runner"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  2742 vars. 136 primary vars. 7109 clauses. 350ms.
  Instance found. Predicate is consistent. 206ms.
```



5. Effort spent

- **Paolo Romeo**

Description of the task	**Hours spent**
Definition of goals, requirements and assumptions	7
Product perspective and functions	4
Specific requirements	4
Formal analysis using Alloy	3
Other related activities	7

- **Andrea Scotti**

Description of the task	**Hours spent**
Definition of goals, requirements and assumptions	6
Product perspective and functions	4
Specific requirements	3
Formal analysis using Alloy	8
Other related activities	7

- **Francesco Staccone**

Description of the task	**Hours spent**
Definition of goals, requirements and assumptions	5
Product perspective and functions	3
Specific requirements	7
Formal analysis using Alloy	4
Other related activities	8

6. References

- Specification Document: Assignments AA 2018-2019.pdf;
- Slides of the lessons;
- IEEE Std 830-1993 - IEEE Guide to Software Requirements Specifications.