Politecnico di Milano

Dipartimento di Elettronica, Informazione e Bioingegneria

TrackMe

R.A.S.D.

Requirements Analysis & Specifications Document

Giulia Mangiaracina [905106]

Andrea Miotto [920287]

Ilaria Moschetto[915191]

Contents

Chapter 1

Introduction

1.1 Purpose

The purpose of this document is to provide a complete description of the TrackMe System, through the analysis of the problems, the listing and presentation of the goals, constraints, phenomena, domain assumptions and models with the aim of provide to the stakeholders a complete overview of the project. TrackMe provides three main services: Data4Help, AutomatedSOS and Track4run.

[G1] The Registered User can access to the services offered by TrackMe System with a single account.

[G2] The Registered User can be recognized by providing his/her username and password.

Goals of Data4Help:

[G3.1] Allow a User to manage the accesses to his/her personal data.

[G3.2] Allow a User to visualize his/her actual health parameters and position.

[G3.3] Allow a User to visualize his/her past data history.

[G3.4] Allow a Third Party to send an authorization request to a User for the access to his/her data.

- [G3.4.1] Allow a Third Party to request the latest available data of a User.
- [G3.4.2] Allow a Third Party to request a subscription to the data of a Users.

[G3.5] Allow a Third Party to request anonymized data of a set of Users.

- [G3.5.1] Allow a Third Party to request the latest available data of the set of Users.
- [G3.5.2] Allow a Third Party to request a subscription to the data of the set of Users.

[G3.6] Allow a Third Party to visualize the available data through useful statistics.

Goals of AutomatedSOS:

[G4.1] NHS is alerted when the User gets in a critical state.

[G4.2] Allow the User to create a list of contacts to be alerted in case of emergency.

Goals of Track4Run:

[G5.1] Organizers can create a new run.

[G5.2] Allow a User to enroll to a run as participant.

[G5.3] Allow a User to visualize the list of scheduled runs.

[G5.4] Allow a User to visualize the list of live runs.

[G5.5] Allow a Users to visualize on a map the positions of the participants in a live run.

1.2 Scope

The aim of the project is to develop a software-based application for the company TrackMe that provides three different services: Data4Help, AutomatedSOS and Track4Run. Data4Help collects data of individuals through smartwatches and similar devices. Registered users can visualize their data through useful statistics, while third parties can access data, after authorization, in real time to monitor the location and health status of a single individual, or a set of individuals in an anonymized way. AutomatedSOS offers a personalized and non-intrusive SOS service to people. Lying on the data collected by Data4Help, it detects anomalies in the health status of the individual and alterts the NHS, allowing the dispatch of an ambulance as soon as possible. Track4Run is a platform that allows the organization of local runs. Organizators can create a new run, setting a customized path and the related preferences. Users can visualize the planned runs in detail and enroll to them as participants if interested. The service uses Data4Help data to show the position of the participants on a map, allowing all the online users to follow the run live on their devices.

1.3 Definitions, Acronyms, Abbrevations

In this part of the RASD Document there are some definitions, acronyms and abbreviations that will be used among the following chapters.

1.3.1 Definitions

- **User:** When referring to *The User*, we refer to a logged- in user which is not a Third party. The software is usable only by a recognized user.
- **Visitor:** When referring to *The Visitor*, we refer to a user not yet registered to the TrackMe system
- **Registered User:** A User registered to the TrackMe System. To use the System and performs any action, all actors must be registered.
- **Logged-in User:** A registered user who is logged in the TrackMe System
- **Organizator:** Is a logged-in user who uses the Track4Run system, and decides to create a new run
- **The System:** when referring to *The System*, this document refers to the entire system infrastructure.

- **Client:** We refer to a normal logged-in user of the application, not a Third Party
- **Normal user:** is a Client
- **TrackMe:** Is the name of our System, that includes our three main services: Data4Help, AutomatedSOS and Track4Run.
- **App:** We refer to either TrackMe mobile application or the web version.
- **Health data:** With health data we refer to the health data collected by user's device, among those available or requested:heart Beat, blood Pressure, Step counter, etc.
- **Request:** A request is performed by a Third Party and concerns normal users. There are 4 types of requests: One-shot user request (concerning the latest available data of a specific user selected inserting his/her fiscal code), subscription user request (it requires an interval time to specify: the third party will receive the updates how often it has requested), one-shot group request (the third party selects the research preferences, and the requested data that match them are anonymized) and subscription group request (the same of one-shot group request, but it requires to specify the update interval time). In All kind of request the Third Party can specify which data it want to receive, among health data available (heart rate, blood pressure, step counter, blood pressure etc) available. The user requests must be accepted by the user to be performed, while the group requests are performed only if concern a group of individuals greater than 1000,for a security factor.
- **Run:** A run is a sports competition organized by and Organizator through the Track4Run System. We divide them between scheduled run, that are runs that has been planned and will take place in the future, and are in the run list, and live run, that are those that are taking place in this moment, are in the live run list and can be followed live.
- **Path :** Every run has a path, i.e. the route that links the points selected by the run Organizator during the event creation: initial point, the intermediate points and the final point.

1.3.2 Acronyms

- R.A.S.D: Requirements Analysis and Specifications Document
- A.P.I: Application Programming Interface
- N.H.S: National Health Service
- S.M.S: Short Message Service

1.3.3 Abbreviations

These abbreviations will be used both in this document and in the follows documents

- [G k]: The k-th goal
- [D k]: The k-th Domain Assumption
- [R k]: The k-th Functional Requirement

1.4 Document Structure

This document is divided into 5 sections:

- Introduction:
  It includes a general overview on the System, the main goals of the application and all the informations useful to understand better the other sections of the document.
- Overall Description:
  It includes the models of the System: UML, shared phenomena and the state charts; the domain assumptions and constraints, the requirements associated to the goals of the previous section, and a reference to the users of the application.
- Specific Requirements:
  This is the main body of this paper. It contains: the external interface requirements, which includes the mockup of the application, both mobile and web; the functional requirements, with all the scenarios and the use cases of the main functionalities, that aim to describe better how the application will work; the performance requirements; all the design constraints; finally, the software System attributes.
- Formal Analysis using Alloy:
  This section contains the Alloy model of our System, with the code and some runs of the more relevant assertions and predicates, to verify its correctness.
- Effort Spent:
  A brief overview on the effort spent by each component of the group.

1.5 Reference Documents

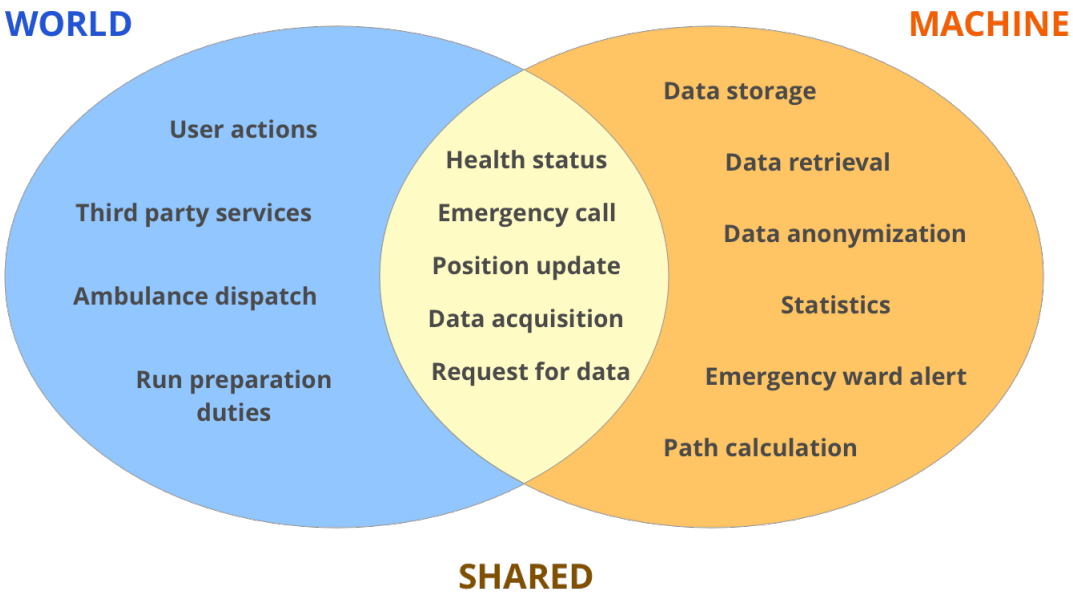
- Specification documents: *Assignments AA 2018-2019.pdf*
- 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering – Life cycle processes –Requirements engineering
- Alloy Language Reference

Chapter 2

Overall Description

2.1 Product Perspective

2.1.1 The world and the Machine

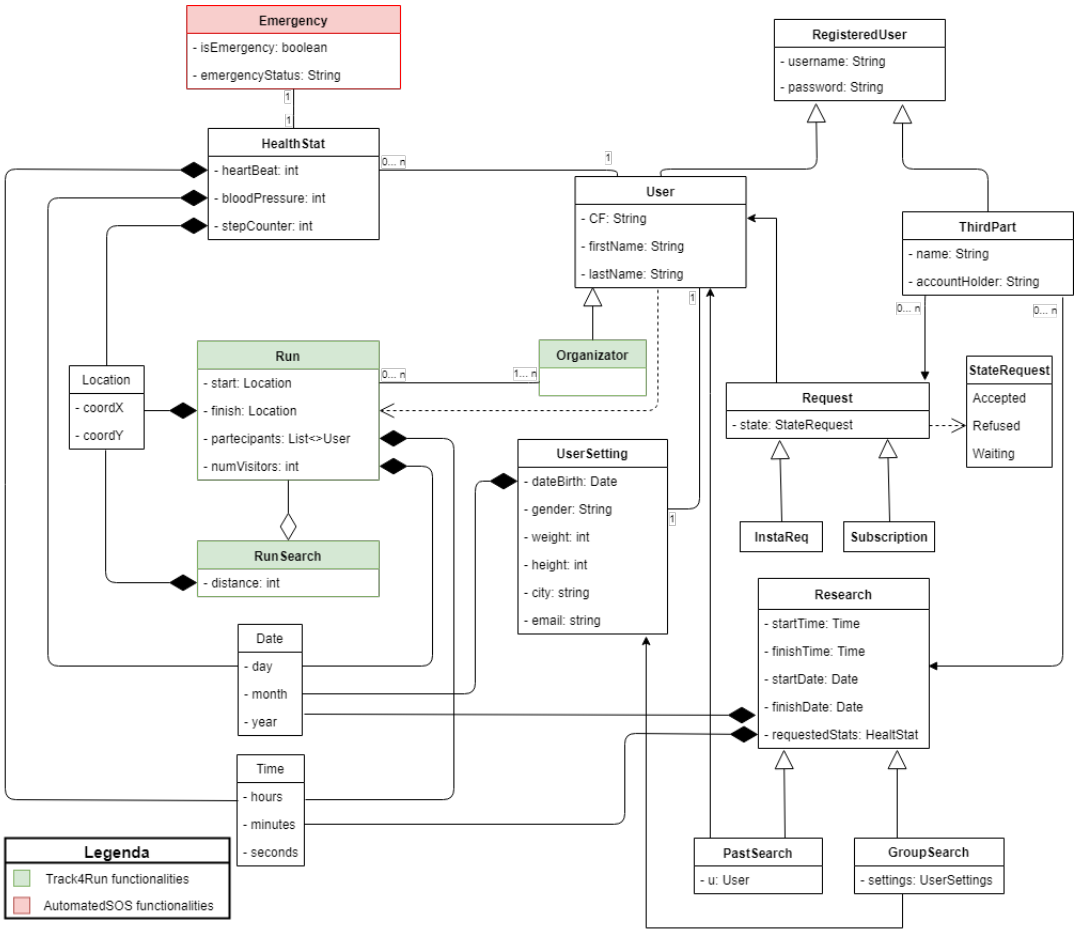


Some clarifications about the diagram:

The User actions are all the behaviors that in the every day life the user performs in the world. The Third Party services are all the services that will arise from the utilization and the study of the collected data. The Run preparation duties are the efforts spent to planning and organizing a run event projected in the real world: e.g. block the circulation of vehicles for the run route that day and mark the run route. The Emergency ward alert represents the background service that the machine performs to check the health status of the users with AutomatedSOS enabled. If it detects an anomalous state, triggers the Emergency procedure.

2.1.2 UML: Class Diagram

This is the UML model of the whole system, based on a class diagram:



2.1.3 State charts:

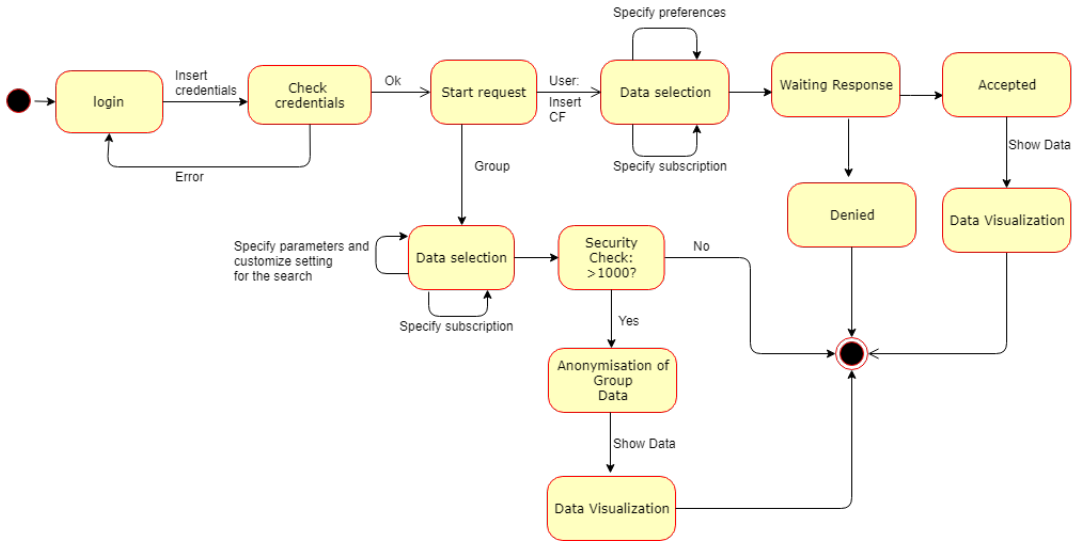


Figure 2.1: State chart of *Data4Help* System

This is a background service that runs in order to keep track of the health status of each user, and notify the NHS in case of Emergency.

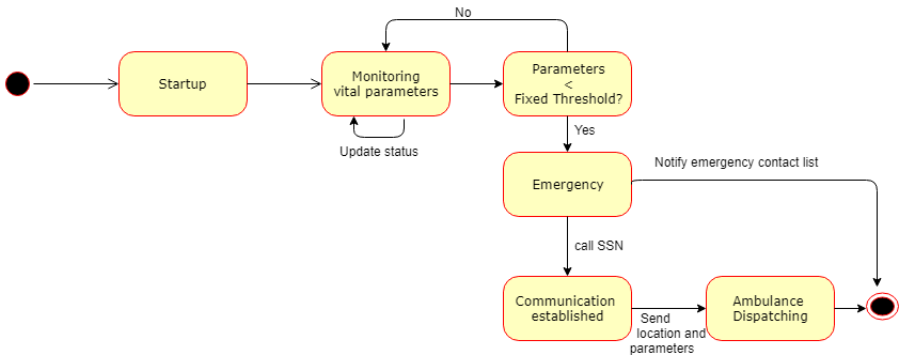


Figure 2.2: State chart of *AutomatedSOS* System

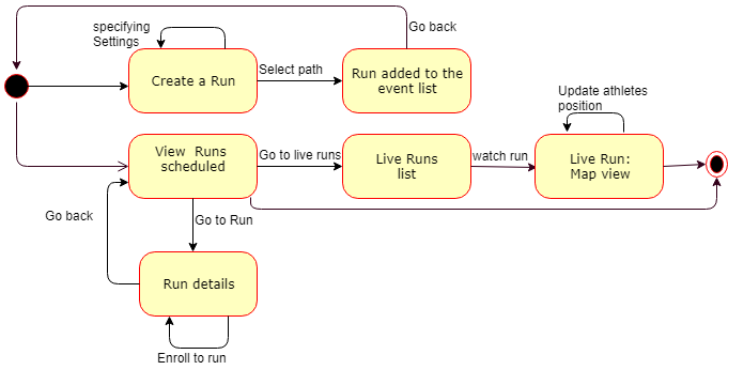


Figure 2.3: State chart of *Track4Run* System

2.2 Product Functions

Here we list the main requirements, concerning each goal.

[G1]The Registered User can access to the services offered by TrackMe System with a single account.

R1)The User accesses to the System only through a mobile application.

R2)The User can access all the three services (Data4Help, AutomatedSOS and Track4Run).

R3)The Third Party can access only the Data4Help service.

R4) The Third Party accesses to the System only through a web application.

R5)The User can decide to keep AutomatedSOS's service active or deactivate it arbitrarily.

[G2]The Registered User can be recognized by providing his/her username and password.

R1)A User have to specify his/her personal data at registration time: first name, last name, fiscal code, age, city, weight and height.

R2)The Third Party have to specify its personal data at registration time: company name, account holder, VAT number, address.

Data4Help:

[G3.1]Allow a User to manage the accesses to his/her personal data.

R1)The User receives in real time requests on his/her data from Third Parties.

R2)The Request can be accepted or denied by the User.

R3)The User can see the specific type of data requested from the Third Party.

[G3.2]Allow a User to visualize his/her actual health parameters and position.

R1)The User must be able to see its position on an interactive map.

R2)The position must be updated in real time.

R3)The System provides the latest health and location data available.

[G3.3]Allow a User to visualize his/her past data history.

R1)The System asks the User what kind of data he/she wants to see.

R2)The data are presented in the order specified by the User.

R2.1)The System provides two grouping options: time and location.

R3)A manual research on its own data history can be performed by the User.

R3.1)The User can customize its search specifying the time or location range.

[G3.4]Allow a Third Party to send an authorization request to a User for the access to the his/her data.

R1)The System asks to insert the fiscal code of the specific User.

R2)The Third Party is asked to specify the type of data to be requested.

R3)The System notify the User about the request from the Third Party.

R4)The System notify the Third Party as soon as the response to their request is available.

[G3.4.1]Allow a Third Party to request the latest available data of a User.

R1)If the request is accepted by the User, the System will provide the Third Party with his/her latest available data.

[G3.4.2]Allow a Third Party to request a subscription to the data of a Users.

R1)The Third Party can define the interval between updates for each specific parameter.

R2)If the request is accepted by the user, the system will provide the third party with periodic updates.

[G3.5]Allow a Third Party to request anonymized data of a set of Users.

R1) The Third Party is asked to specify the type of data to be requested.

R2)The data requested will be chosen among those that match the preferences specified by the Third Party: age range, geographical area, localization, gender, weight, height.

R4)The data will be anonymized to prevent the possibility of a misuse of data.

[G3.5.1]Allow the third party to request the latest data of the set of Users.

R5.1)If the request is allowed, the System will provide the Third party with its latest available data about group.

[G3.5.2] Allow a Third Party to request a subscription to the data of the set of Users.

R5.2)The Third Party can define the interval between updates for each specific parameter.

R5.3)If the request is allowed, the System will provide the Third Party with periodic updates about the group.

[G3.6] Allow a Third Party to visualize the available data through useful statistics.

R1)The System provides the Third Party with different graphical representations of the available data.

AutomatedSOS:

[G4.1] NHS is alerted when the User gets in a critical state.

R1)The System continuously keep track of the vital parameters of the User.

R2) The System communicate to NHS the health status of a User when his/her parameters are below certain thresholds.

R3) The System retrieves from its database the right emergency number for the geographical area based on the User's location.

R4) The System calls the emergency number, communicates the vital parameters and the actual location of the User, and asks the dispatch of an ambulance.

[G4.2]Allow the User to create a list of contacts to be alerted in case of

emergency.

R1) The system sends a message to all the numbers on the specified list in case of emergency.

R2) The user can customize the text message to be sent.

R3) If there is not a customize text message, the system will send a default message.

Track4Run:

[G5.1] Organizers can create a new run.

R1) The System asks the Organizer to select a starting and finishing point for the run.

R2) The Organizer can add intermediate points that will be part of the path.

R3) The System relies on an external service to provide the interaction with a map.

R4) The System calculates the shortest path that includes all the points selected by the Organizer.

R5) The System asks the Organizer to specify the maximum number of participants.

R6) After being created, the run is added to the list of the scheduled runs.

[G5.2] Allow a User to enroll to a run as participant.

R1) The System allow to enroll to a run if there are available entries.

R2)The System sends an email to the User's email address with the information related to the run and the ticket (with the bib number).

R3) The User is notified through email updates if there are changes until the day of the run.

[G5.3]Allow a User to visualize the list of scheduled runs.

R1) The System shows only future scheduled runs.

R2) A manual search can be performed by the User specifying the location range and the minimum/maximum distance.

R3) The System provides the list of scheduled runs ordered by time and User's lo-

cation.

[G5.4] Allow a User to visualize the list of live runs.

R1) The list of live runs shows only the list of runs that are taking place in that moment or that are already started.

R2) The User can decide to follow a live run from the list, directly from their device.

[G5.5] Allow a User to visualize on a map the positions of the participants in a live run.

R1) The positions on the map are updated in real time.

R2) The System shows the number of online visitors that are watching the run.

2.3 User characteristics

The main registered users of Data4Help are third parties interested for various purposes in collecting and analyzing of user's data. Users are people of all kinds, who want to keep track of their data in time and take advantage of the services derivative from the analysis of their own data. The users of AutomatedSOS are people who decide to rely on this service essentially for a security factor, to be assisted in case of emergency.

The user of Track4Run are sports lovers who concerned in planning, partecipating and follow live runs streaming on Internet, directly from their devices everywhere.

2.4 Assumptions, dependencies and constraints

Domain Assumptions:

[D.1] The user owns a smartwatch or similiar device, connected to the user's smartphone.

[D.2] The smartwatch collects the data through its sensors.

[D.3] The device has regular Internet connection.

[D.4] The device has a working GPS system.

[D.5] The GPS position has an accuracy of less than 5 mt.

[D.6] An external map service will be used.

[D.7] The map service is reliable.

[D.8] Only registered users, either as third party or clients, can use the application.

[D.9] At registration time, the user provides correct data: fiscal code, first name, last name, age, city, email, height and weight.

[D.10] At registration time, the third party provides correct data: account holder, VAT number, address and email.

[D.11] The device's sensors send new data at least every 5 seconds.

[D.12] After the start of an emergency, an ambulance will be dispatched at the user's location as soon as possible.

[D.13] For every urgent call, details about the emergency are correctly encoded.

[D.14] The threshold value for detect and emergency state will be provided by a trusted medical authority.

[D.15] The database of emergency world number is updated and reliable.

[D.16] The map service provides only feasible paths accessible by foot.

Constraints:

1. A group request can be executed only if there are at least 1000 people that match the requested characteristics.
2. From the time the parameters are below a fixed threshold, the communication with NHS must be established within 5 seconds.
3. The subscription to a run is closed if the maximum number of participants is reached.

Chapter 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

Third Party's Web Interface

Figure 3.1: Log-in on the web application of *TrackMe* System

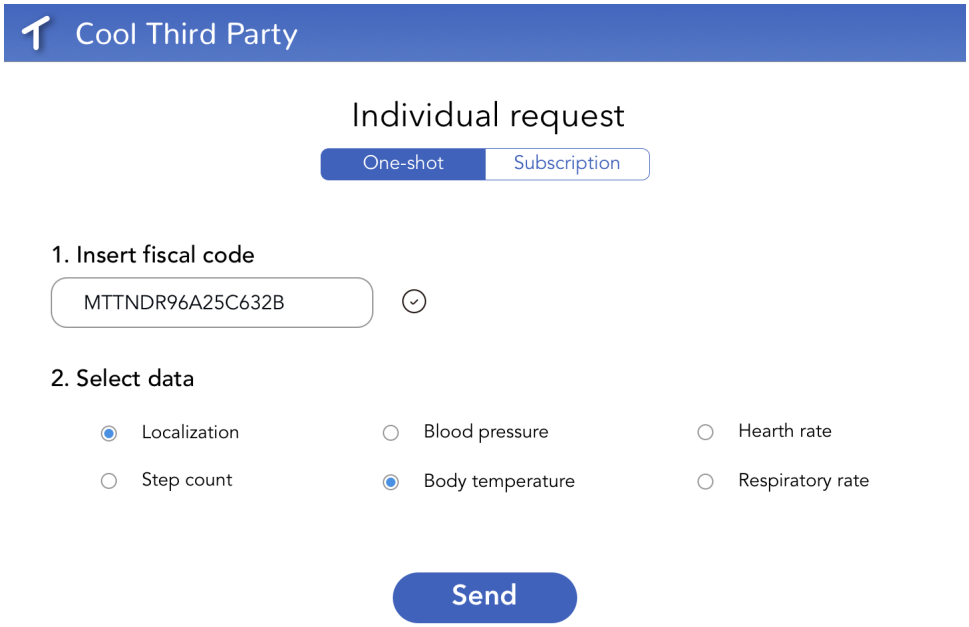


Figure 3.2: One-shot user request

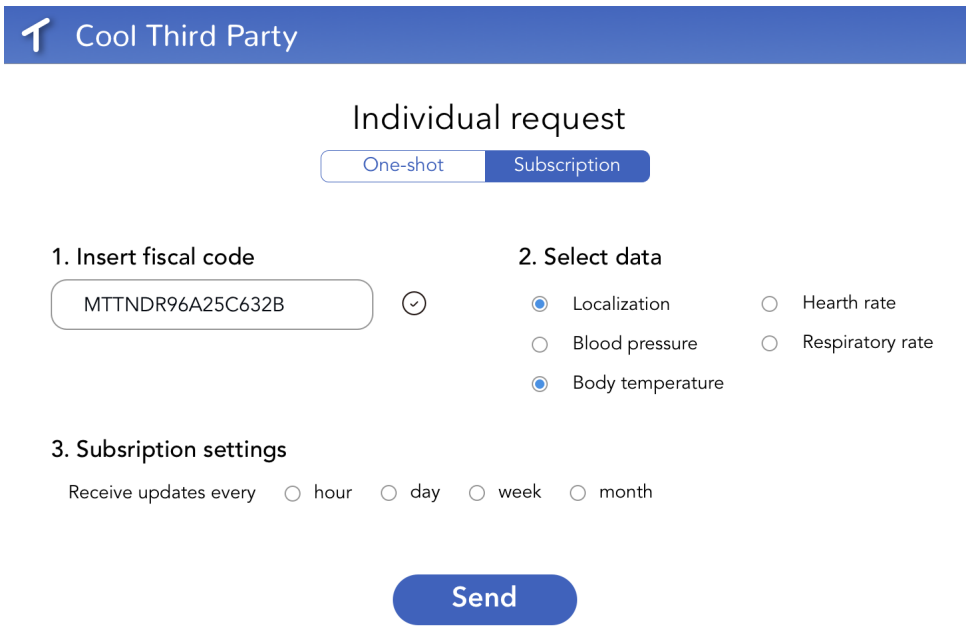


Figure 3.3: Subscription user request

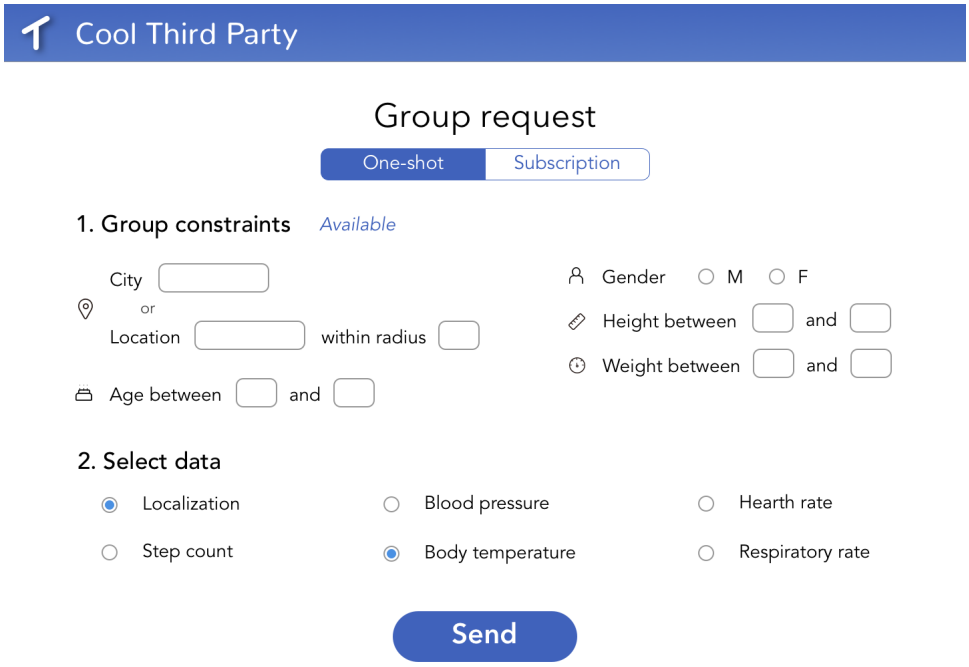


Figure 3.4: One-shot group request

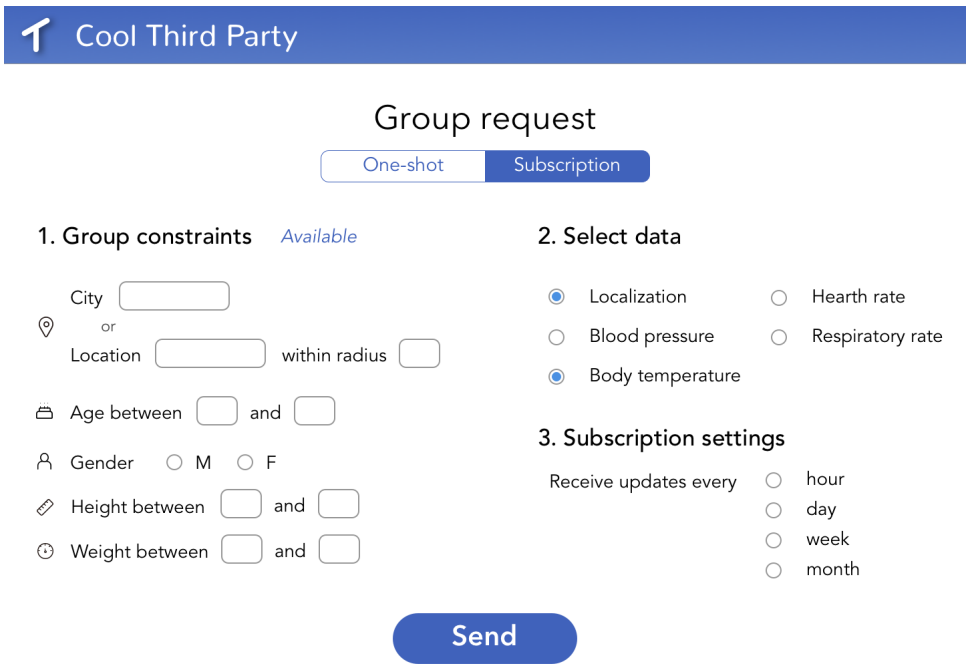


Figure 3.5: Subscription group request

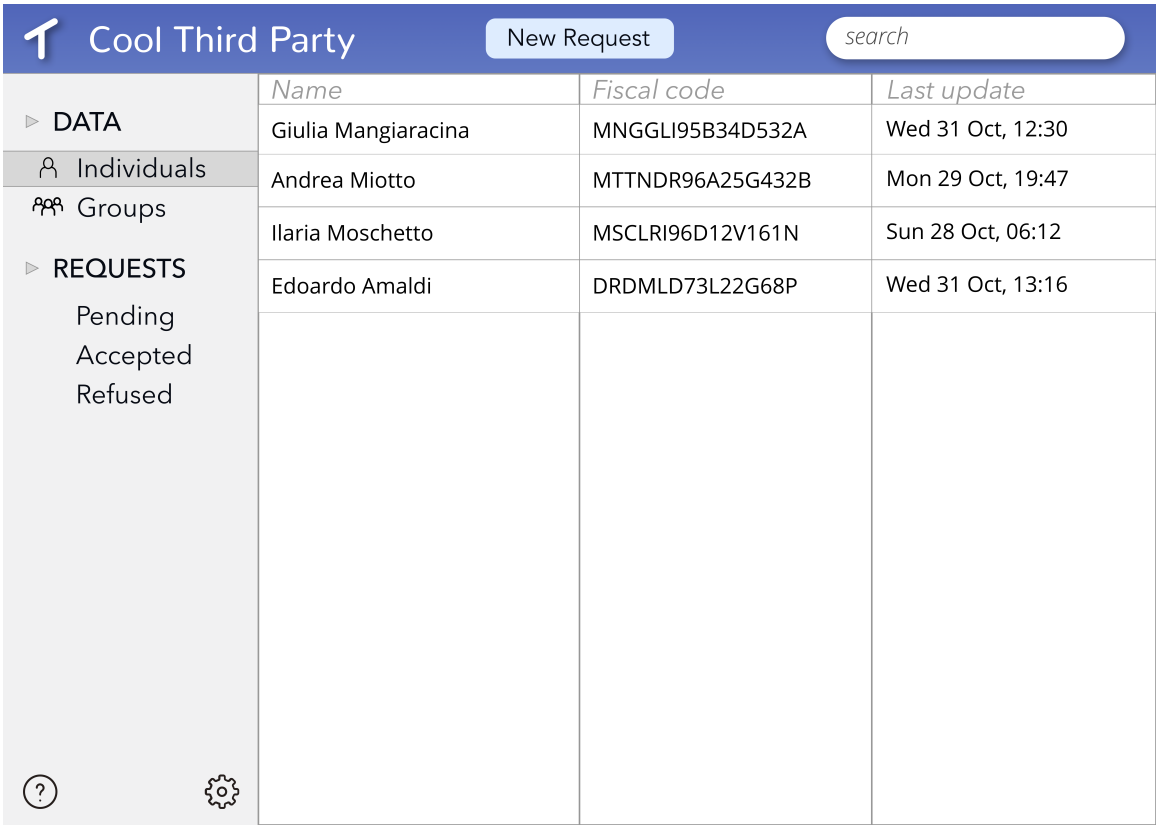


Figure 3.6: Performed Requests

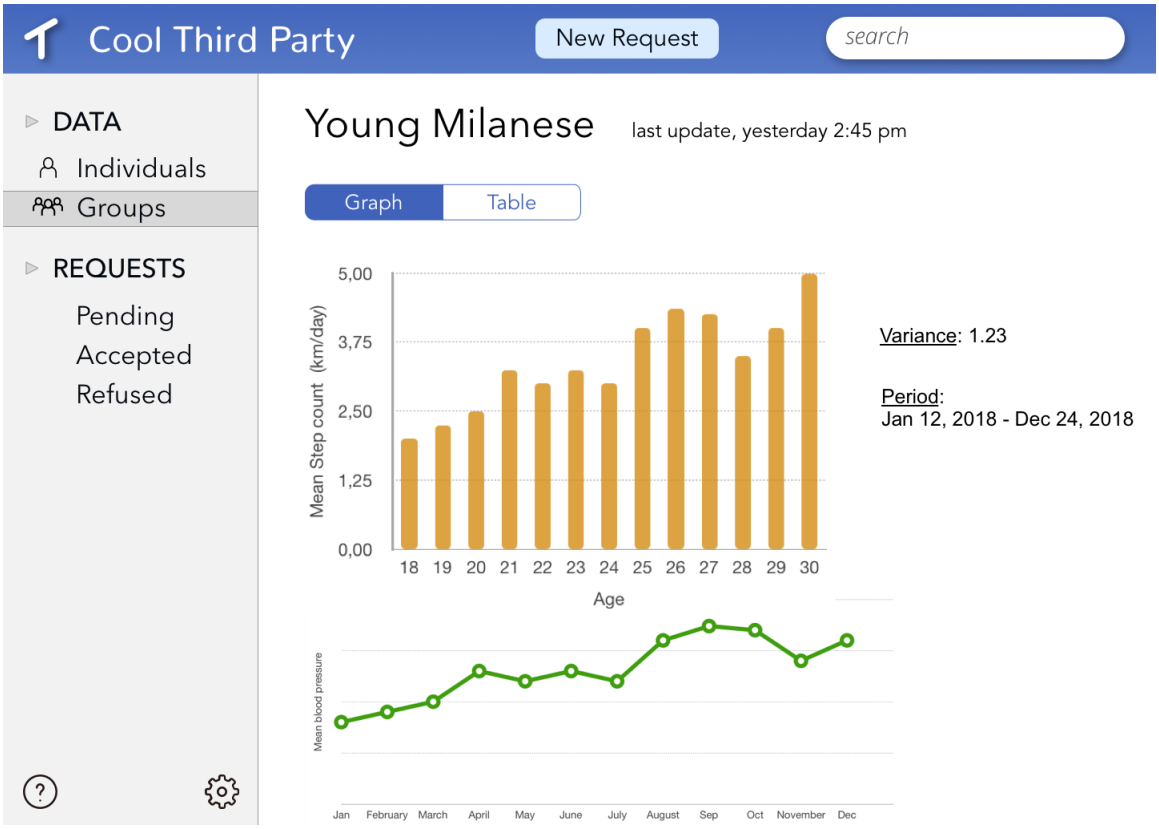
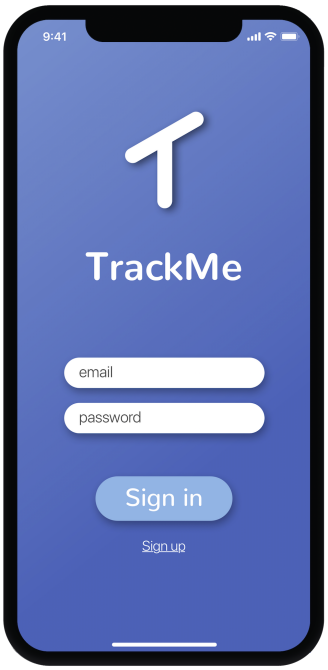
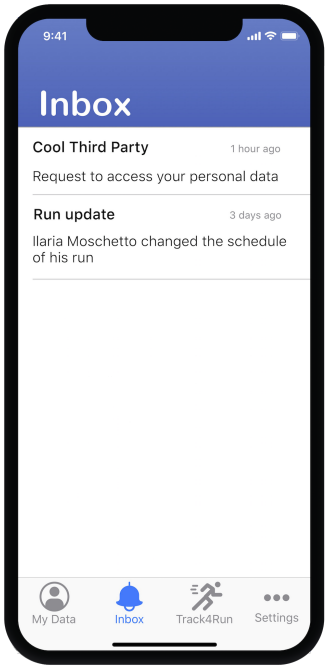


Figure 3.7: Results of a group request

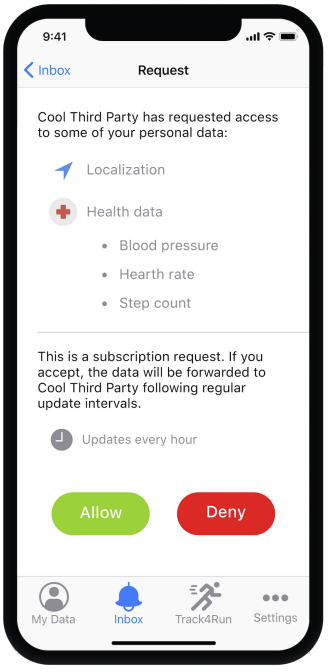
User's Mobile Interface



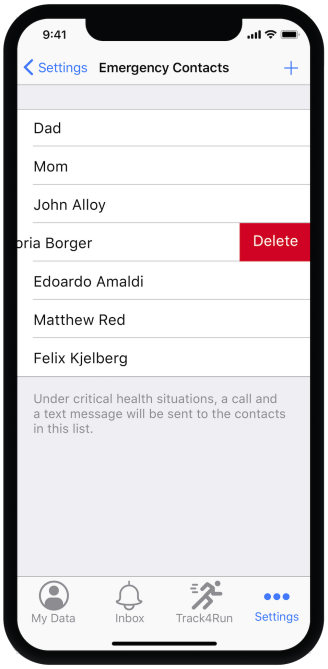
Log-in



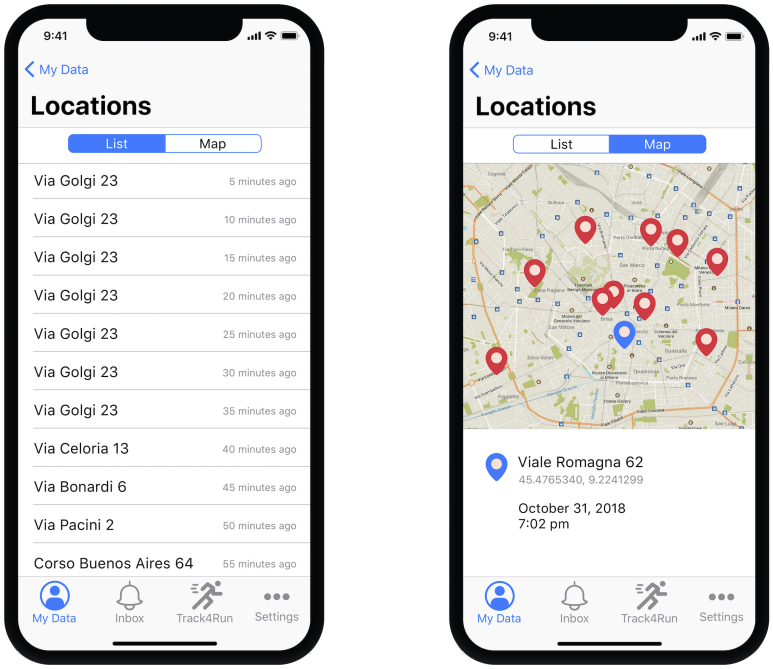
Inbox view



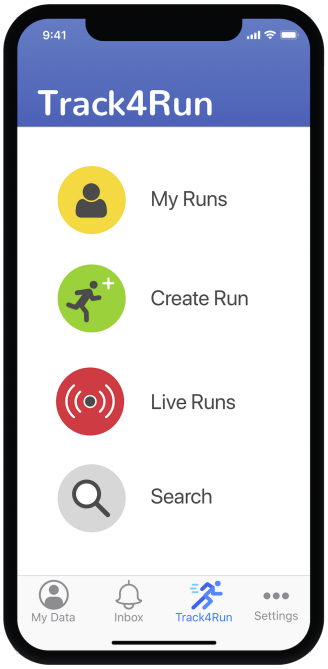
Reply to a Request



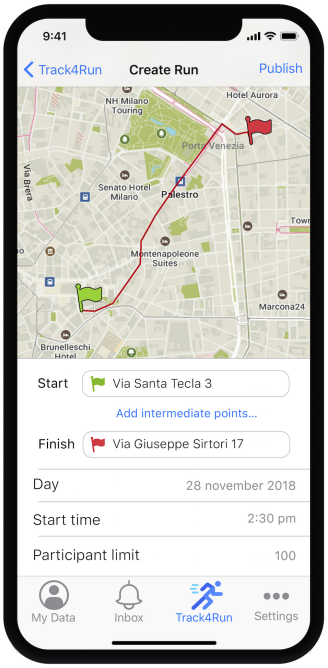
Emergency contact list of AutomatedSOS System



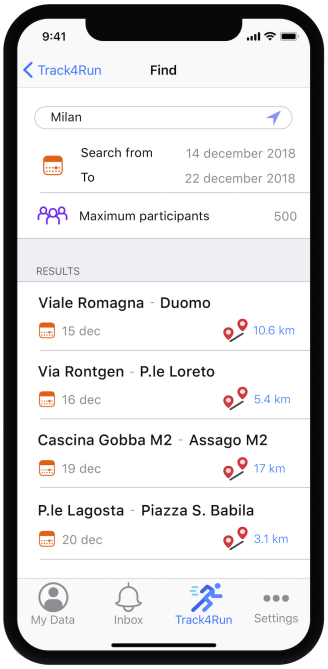
User's location History: List and Map visualization



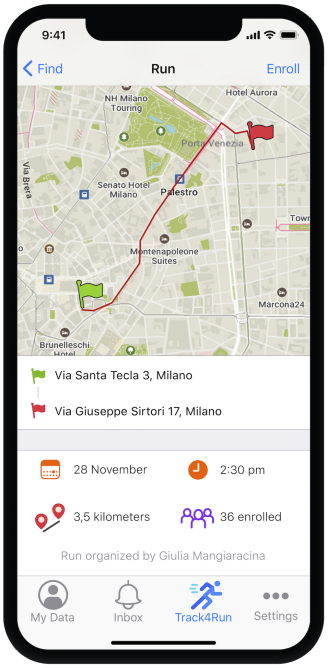
Track4Run Service



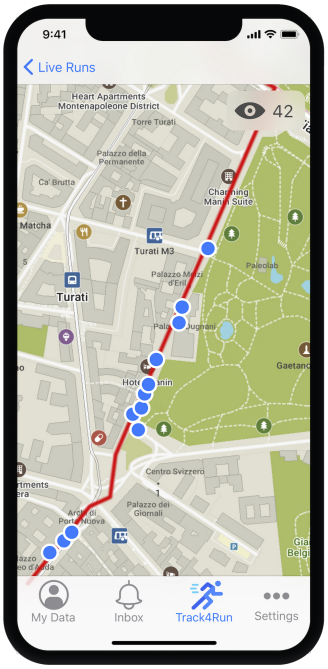
Create a Run



Search a Run



Visualize Run detail



Follow a Run live

3.1.2 Hardware Interfaces

The system requires users to own a smartwatch (or similiar device) and a smartphone provided with an internet connection and GPS. The system needs one or more servers dedicated.

3.1.3 Software Interfaces

The system will need:

- A client database to store data in the User's device, so that in case of lack of connection the data are not lost.
- A server database that will store the User data history and allow to retrieve the data through queries.
- A mobile application that will be dowloaded on the User's devices (both on smartwatch and smartphone).
- A web application usable from a modern browser that provides access to TrackME services for the Third Party.

3.1.4 Communication Interfaces

The communication between the client application and the server will be provided by a custom-built API. The connection will follow the HTTPS protocol. The various User's smartdevices can communicate with each other through Bluetooth connection.

3.2 Functional Requirements: Use Cases and Scenarios

3.2.1 Scenarios:

Scenario A
Tom is a middle age man who wants to keep an healthy lifestyle. He wants to go to work on foot, to walk at least 10000 steps for day. Through the TrackMe System, he can visualize his actual and past health data, and check if he reaches his daily goal.

Scenario B
Happy Company is a medical association registered as a Third Party to the TrackMe System. It exploits the Data4Help System to offer a new service to their patients which already use TrackMe. The patients that accept the company's subscription request about health data are monitored daily, and the collected data are stored in their personal medical records. During the examinations, those data are consulted by the doctors to complete the patient switchboard.

Scenario C
Data Corporation is a company interested in huge amount of data to make statistical analysis on different sets of individuals, based on different parameters e.g. age range, gender, city. It is registered to the TrackMe System as a Third party and uses the System as an useful database to query for its purposes, performing periodically anonymous researches on groups and studying the data that receives.

Scenario D
John is an elderly cardiopath man who is walking in the street. He's going to buy the newspaper like every morning. John has received a smartwatch as Christmas's gift from his son. His son has already managed to download the TrackMe application, register his father to TrackMe service and enable AutomatedSos System. Moreover he has added himself as emergency contact, in order to be alerted in case of emergency. Suddenly John feels bad and falls to the ground. The sensors of his smartwatch detects an anomalous health status, probably an heart attack, and AutomatedSOS starts the Emergency procedure: calls the emergency number of the NHS, communicates the location and the health status of John and asks for an ambulance. His son is notified. The ambulance arrives as soon as possible and the doctors take care of him.

Scenario E
The mayor of an italian city wants to organize a city run. She decides to manage the event with Track4Run. The interested citizens can sign up to the run through the enroll function. She selects the day and the time of the event, a path of 10 km that includes all the main places of the city, and sets as maximum number of participants 2000. The entries are soon sold out and the run is a successful event. The Mayor then decides to organize the event every year.

Scenario F
Steve loves sports. He is far from home for a business trip, but he knows that on that day a city run will take place, organized by the mayor of his city through Track4Run System. He downloads the app and, at the starting time, selects the run from the live run list and follows it comfortably from his device.

3.2.2 Registration and Login System

Every User who wants to use the application must be registered to the System. After that, the User/Third Party has to log in into the System in order to be recognized and use the mobile/web application.

Actors	Visitor Registered User Logged-In User
Goals	[G1] [G2]
Enter Condition	There is no enter condition for this Use Case
Events Flow	1. The *Visitor* accesses to the web site or application log in page. 2. The *Visitor* inserts all the mandatory informations (username that will identify the user and password, personal data, email). 3. The System registers the new User and sends back a confirmation email to the provided email address. 4. The *Visitor* becomes a *Registered user* inserting a unique username and the password in the log in page. 5. The *Registered User* now can access the *TrackMe* services through the log in page. 6. After the insertion of the identifier and password, the *Registered User* becomes a *Logged-in User*.
Exit Condition	The User/Third Party is registered in the *TrackMe* System, and his account is added to they system. Now the User/Third Party is able to use all the functionalities provided by the system.
Exception	1. The *Visitor* cannot register himself because is already registered. 2. The *Registered User* is not able to sign in the System because the login data are wrong or he did not confirmed the confirmation email. If one of these problems occur, the system both on the web site and on the application shows a message error to the User/Third Party, which is invited to re-insert their credentials or confirm the registration email.

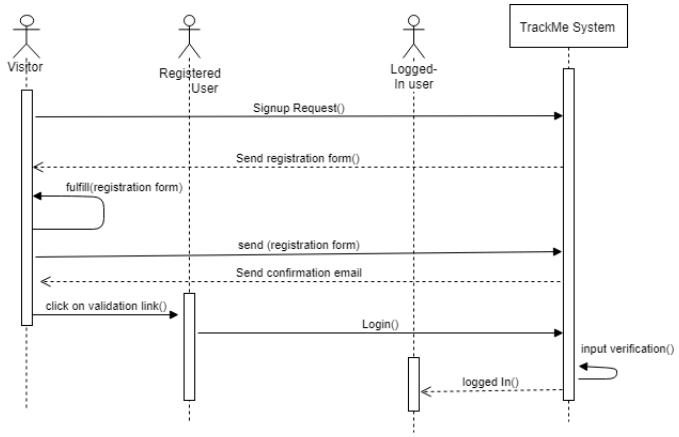


Figure 3.8: Sequence diagram for the registration and login process

3.2.3 Request of selected data of a specific user with subscription and data visualization

The Third Party can exploit the Data4Help System, sending a request for the selected data (health data and location) of a specific User, either the last available data, or in subscription mode. In the latter case it inserts the interval time, i.e. how often wants to receive the updates of these data. The User can manage his requests, and decide to accept or deny them. If the request is authorized, the Third Party receives the requested data from the System, according to the chosen modalities.

Actors	Third Party User
Goals	[G3.1] [G3.4] [G3.4.1] [G3.4.2] [G3.6]
Enter Condition	The Third Party is already logged in.
Events Flow	1. The *Third Party* selects the Individuals section. 2. The *Third party* presses the "New Request" button. 3. The *Third party* inserts the specific fiscal code of the target User. 4. The *Third party* specifies the data that want to receive. 5. The *Third Party* can select the subscription mode. If it is chosen, the *Third Party* should also specify the interval time for receiving updates. 6. The System sends an authorization request to the target User with all the related informations. 7. The *User* receives the request in his "Inbox" section and can decide to accept or deny it. 8. If the request is accepted, the System retrieves the requested data. (If the *Third Party* has performed a subscription request, the System sends the updates to the *Third Party* how often it has requested). 9. The requested data are shown to the *Third Party*.
Exit Condition	The Third Party can examine the requested data.
Exception	1. The target *User* denies the request: the negative response is communicated to the *Third Party*. 2. The selected fiscal code does not correspond to a registered *User*:the System asks to insert a validate one.

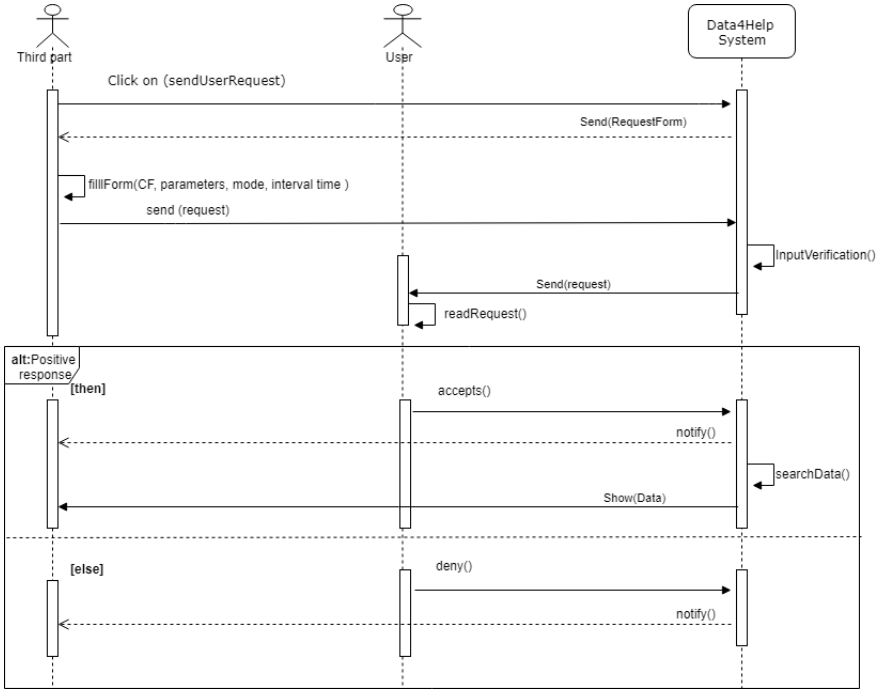


Figure 3.9: Sequence diagram for requesting data of a specific User

3.2.4 Group Research

The Third Party can make a request for anonymized data specifing the constraints of the group, the data that wants to retrieve and optionally the subscription mode with the interval time of updates. The anonymized group data are shown to the Third Party only if the privacy constraints are respected (i.e. there are at least 1000 users that match the specified constraints).

Actors	Third Party
Goals	[G3.5] [G3.5.1] [G3.5.2] [G3.6]
Enter Condition	The Third Party is already logged in.
Events Flow	1. The *Third Party* selects the Group section. 2. The *Third Party* selects the "New Request" button. 3. The *Third Party* specifies the group constraints (such as city or location, the age interval, gender, height and weight, that can be chosen optionally). 4. The *Third Party* selects the data that wants to retrieve and, if it has chosen the subscription mode, should also specify the interval time for the updates. 5. Finally, the *Third Party* sends the request, selecting the "Send" button. 6. If the privacy constraints are respected, the System sends the requested anonymous data to the *Third Party*.
Exit Condition	The Third Party can visualize the anonymized data relative to the given specifications.
Exception	1. The *Third Party* specified an inexistent city or location. 2. The *Third Party* specified an impossible age/height/weight range. 3. In subscription mode, the *Third Party* hasn't specified an update interval time. 4. The *Third Party* hasn't specified the data that wants to retrieve. If one of these problems occur, the System shows a error message to the Third Party, which is invited to re-insert the data or to specify the missing data.

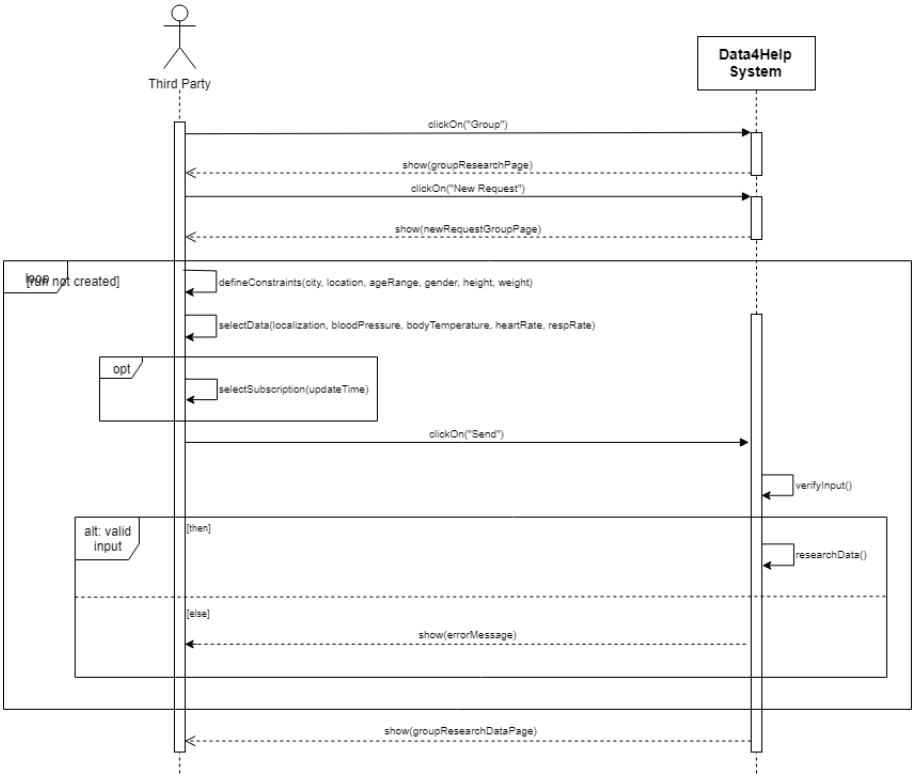


Figure 3.10: Sequence diagram for anonymized group research performed by a Third Party

3.2.5 An User visualizes his History data through a customized research

The User can access to his/her past data stored in the system, specifying the interval time of interest or the location in which the data were collected. If the User does not insert any preference, are shown all the available data, and the User can reads them performing a manual research. The data can be visualized ordered by time or by location (e.g. the health data collected when the User were in the city of Milan).

Actors	User
Goals	[G3.3]
Enter Condition	The User should be logged in.
Events Flow	1. The *User* presses the "History" button. 2. The System asks to specify what kind of parameters he wants to see. 3. The *User* specifies the parameters of the data that wants to visualize. 4. The *User* can specify the time or the location range. 5. The System retrieves the informations requested from the database. 6. The System shows the data to the user. 7. The *User* selects the visualization order(time or location order). 8. In case of localization data, the *User* can view them more in detail on an interactive map pressing on the map button.
Exit Condition	The User can examine the requested data.
Exception	1. The requested data do not exist in the database. In this case the System shows an error message and invites the User to insert consistent preferences.

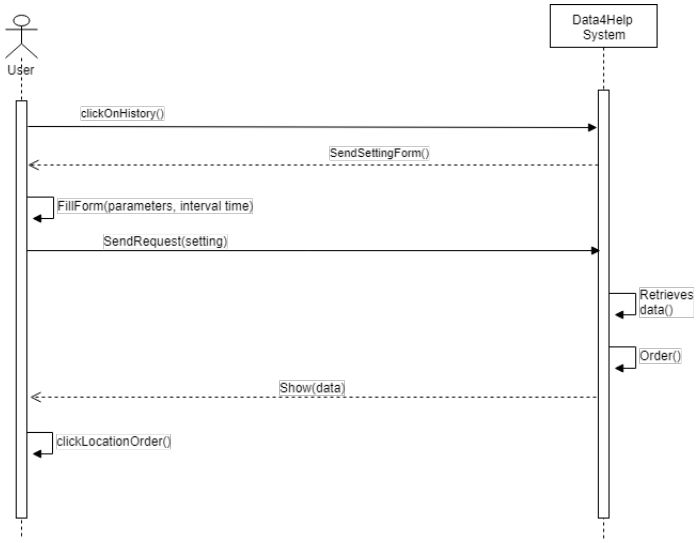


Figure 3.11: In this use case the User performs a history search on his data specifying an interval time and visualizes them in location order

3.2.6 Emergency call

The system detects an emergency, calls the NHS and send a text message to the contacts that are present in the emergency list.

Actors	NHS
Goals	[G4.1]
Enter Condition	The latest health data retrieved from the smart device exceed the critical parameters.
Events Flow	1. The System creates a basic diagnosis based on the anomalous parameters. A text-to-speech algorithm generates the audio file from the diagnosis and the localization of the User. 2. The System calls the *NHS* emergency number and request the dispatching of an ambulance. 3. The System sends a text message (SMS) to the contacts from the emergency list.
Exit Condition	A call to the *NHS* is performed by the System.
Exception	The cellular credit is insufficient to deliver the text messages.

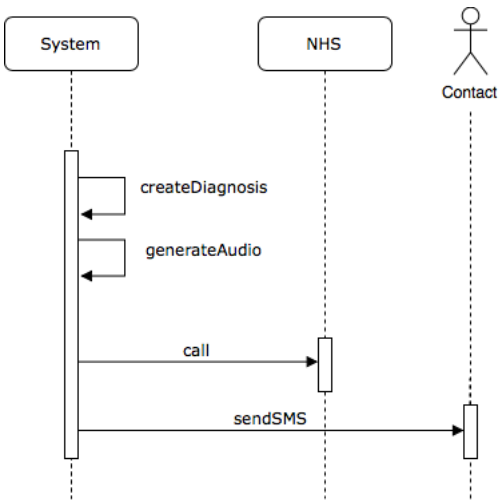


Figure 3.12: Sequence diagram for the occurrence of an emergency status

3.2.7 Create a new run

Every Organizator can create a new run, specifing the start and finish locations, date, time and the maximum number of enrollments.

Actors	Organizator
Goals	[G5.1]
Enter Condition	The Organizator is already logged in.
Events Flow	1. The *Organizator* accesses to the "Create Run" page of the application. 2. The *Organizator* inserts all the required informations to define the run (start and finish locations, date, time and maximum number of partecipants). 3. The *Organizator* defines the list of intermediate locations that will be part of the run's path. 4. The System adds the new run into the list of scheduled runs.
Exit Conditions	The new run is created and successfully added into the list of scheduled runs.
Exceptions	1. The *Organizator* specified a start or finish location that does not exist. 2. The *Organizator* specified intermediate points not reachable on foot. 3. The *Organizator* specified a negative maximum number of partecipants. 4. The *Organizator* specified a date or time that is already passed or the current day. 5. The *Organizator* specified a run with all the same informations of another run. If one of these problems occur, the system shows an error message to the user, which is invited to re-insert the required informations for creating a new run.

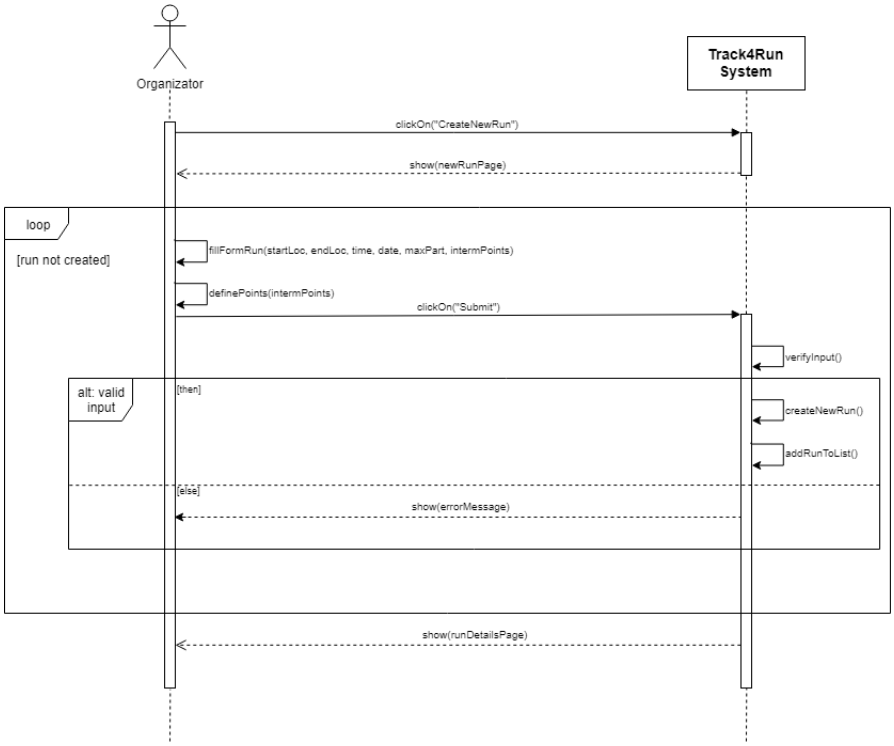


Figure 3.13: Sequence diagram for creating a new run

3.2.8 Enroll to a scheduled run

After visualizing the list of scheduled runs, the User can choose one of them and, if there are available entries, can enroll to the chosen run.

Actors	User
Goals	[G5.2]
Enter Condition	The *User* has already visualize the list of scheduled runs.
Events Flow	1. The *User* chooses a run from the list of scheduled runs. 2. The System shows the details relative to the chosen run. 3. The *User* enrolls to the chosen run selecting the "Enroll" button. 4. If there are available entries, the System adds the *User* to the list of partecipants of the run. 5. The System sends an email with all the informations about the run, the ticket's code and the bib number to the User's email address. 6. The System notifies the *User* through email updates in case of scheduling changes before the day of the run.
Exit Conditions	The User is enrolled to the run and added to the list of partecipants of the chosen run.
Exceptions	1. The *User* tried to enroll to a run that has no more available entries. If this problem occurs, the system shows an error message to the user, which is invited to choose another run.

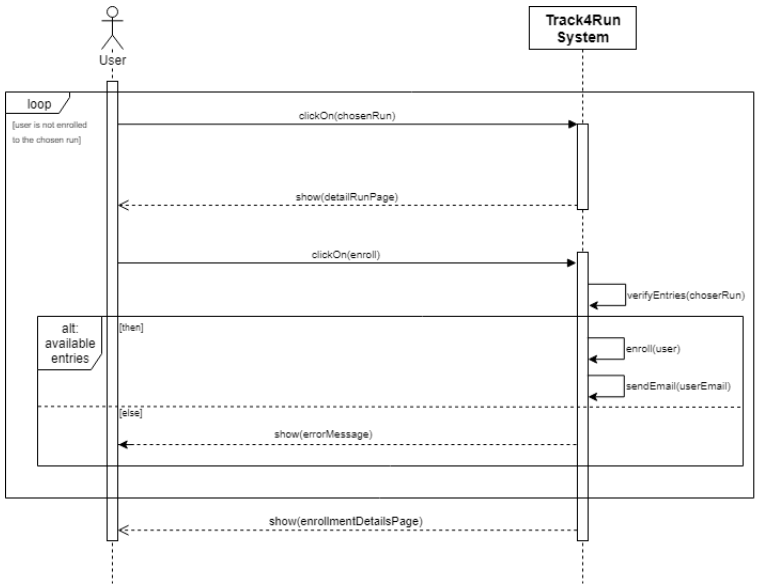


Figure 3.14: Sequence diagram for the enrollment of a user to a scheduled run

3.2.9 Visualize the list of scheduled runs

Into the page of *Track4Run* a User can perform a manual search specifying the characteristics (location range and minimum/maximum distance) of the scheduled runs that wants to visualize. Then the System will show the list of scheduled runs that meet the User's specifications.

Actors	User
Goals	[G5.3]
Enter Condition	The *User* is already logged in.
Events Flow	1. The *User* selects the "Search" button on the Track4Run page. 2. The *User* specifies the characteristics of the run that wants to visualize(if not specified the System will show all the scheduled runs). 3. The System shows the list of scheduled runs that meet the required features.
Exit Conditions	The list of scheduled runs, that correspond to the user's choice, is shown.
Exceptions	There are no exceptions.

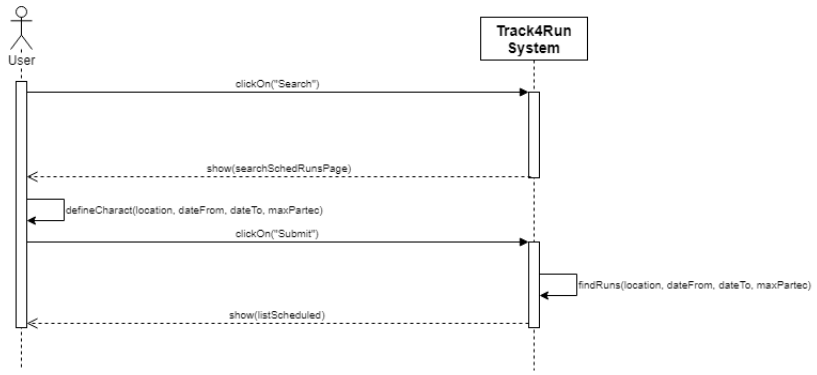


Figure 3.15: Sequence diagram of the search and visualization of the list of scheduled runs

3.2.10 Visualize the list of live runs

Into the page of *Track4Run* a User can visualize the list of live runs that can be followed online.

Actors	User
Goals	[G5.4]
Enter Condition	The *User* is already logged in.
Events Flow	1. The *User* selects the "Live Runs" button into the Track4Run page to visualize the list of live runs. 2. The System shows the list of all the available live runs.
Exit Conditions	The list of live runs is shown.
Exceptions	There are no exceptions.

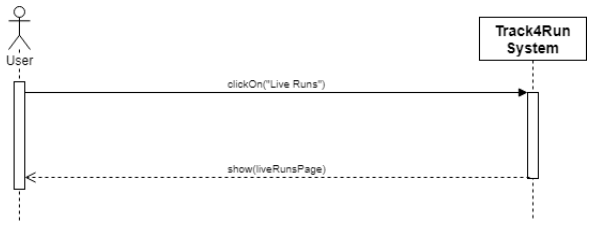


Figure 3.16: Sequence diagram for the visualization of the list of live runs

3.2.11 View a live run

A User can view a run live through his/her device, visualizing the map in which the run is being performed and the relative locations of each partecipant.

Actors	User
Goals	[G5.5]
Enter Condition	The User has already visualize the list of live runs.
Events Flow	1. The *User* selects the run that wants to view. 2. The System shows the details about the run (location, date, time, number of partecipants, online spectators). 3. The *User* can select the 'View Live' button to visualize the map of the run and the live location of each partecipant. 4. The System shows the map with the partecipants locations and increments the number of online spectators.
Exit Conditions	The User visualizes the run and the number of online spectators in incremented.
Exceptions	There are no exceptions.

Figure 3.17: Sequence diagram for the view of live run

3.2.12 Use Case Diagrams:

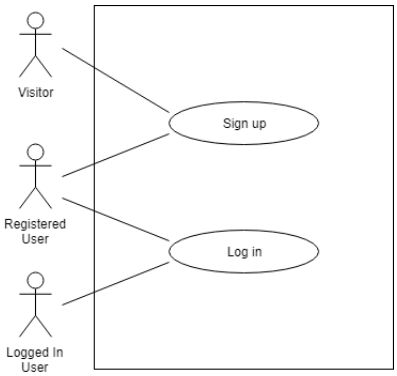


Figure 3.18: Visitor's Use case of *TrackMe* System

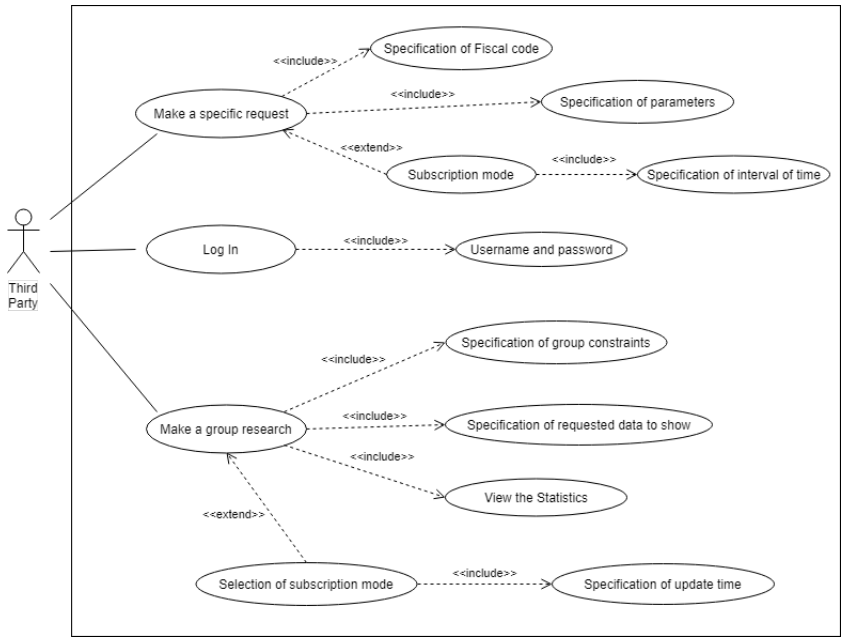


Figure 3.19: Third Party's Use case of *TrackMe* System

Figure 3.20: User's Use case of *TrackMe* System

3.3 Performance Requirements

The application should be able to serve a big number of users simultaneously for the collection and research of data with the fastest response possible. It should be flexible to respond to different demands of workload. The application should provides a seamless user interaction, with the minimum amount of lag and stuttering.

3.4 Design Constraints

- The mobile application will be available for Android (from 6.0) and IOS systems (from 10.3.3). The web application can be browsed through any modern browser.
- Connectivity:
  - 2G, 3G, 4G.
  -GPS services.

3.5 Software System Attributes

3.5.1 Reliability

TrackMe guarantee a service that is continuous in time. AutomatedSOS must guarantee a 24h per day and 7 days per week service, since this is an emergency service and should monitor the health status of each user continuosly to detect any occurence of an emergency.

3.5.2 Availability

Our system will provide an availability of 99.75%, with a downtime of about 8.73 hours/year.

3.5.3 Security

Through the https protocol we guarantee a secure communication between client and server. User's data are encrypted to protect data against unauthorized accesses. The use of salting will be supported.

3.5.4 Maintainability

The system will be modular and extensible to make it more mantainable and stable in case of following update or the introduction of new software applications that can be integrated with the TrackMe system.

3.5.5 Portability

The mobile app will support the two main mobile operating system (Android and IOS). The web app will be available on any modern browser.

Chapter 4

Formal Analysis using Alloy

Here we present a possible formal model for the TrackMe system. The model is built using the Alloy specifications.

We modelled mainly the Data4Help and Track4Run services, with just a small part dedicated to AutomatedSOS. In the intent not to overwhelm the model with numerous signatures and attributes, some features of the system have been left out and we focused on the request mechanisms, the run organization and the anonymity of data in group requests. In particular, we modelled an anonymous identifier that permits the identification of a user in a group, without accessing his personal information.

```
open util/integer
open util/boolean
open util/time

sig Username{}
sig CF {}
sig VAT {}

abstract sig RegisteredUser {
    username: one Username,
}

sig User extends RegisteredUser {
    cf: one CF,
    data: some Record,
    automated_sos: one Bool
}

sig ThirdParty extends RegisteredUser {
    vat: one VAT,
    // For simplicity we consider only positive requests
    positive_requests: set Request
}

sig HealthStatus {
    heartBeat: lone Int,
    bloodPressure: lone Int,
    bodyTemperature: lone Int,
    stepCounter: lone Int
}

sig Position {
    latitude: one Int,
    longitude: one Int
}

sig Record {
    timestamp: one Time,
    location: one Position,
    health: one HealthStatus,
    anon_id: one Int
}
```

```
sig Description {}
sig EmergencyRecord extends Record {
    user: one User,
    description: one Description
} {
    this in user.data
}

abstract sig Status {}
// only one of these can be true
one sig Accepted extends Status{}
one sig Pending extends Status{}
one sig Refused extends Status{}

abstract sig Request {
    sender: one ThirdParty,
}

sig UserRequest extends Request {
    receiver: one User,
    status: one Status,
    obtained_data: lone Record
} {
    // Data is present only if the request is accepted by the user
    (status = Pending or status = Refused) implies no obtained_data
}

sig GroupRequest extends Request{
    restrictions: Restrictions,
    allowed: one Bool,
    obtained_data: set Record
} {
    // Data are present only if the request is allowed by the system
    allowed.isFalse implies no obtained_data
    allowed.isTrue implies some obtained_data
}

sig SubscriptionUserRequest extends UserRequest {}
sig SubscriptionGroupRequest extends GroupRequest {}
```

```
// Restrictions applied to group requests
sig Restrictions {
    location: lone Position,
    radius: lone Int,
    age_max: lone Int,
    age_min: lone Int
}

sig Run {
    organizer: one User,
    path: one Path,

    start_time: one Time,
    end_time: one Time,

    max_participants: one Int,
    enrolled_users: set User,
    num_spectators: one Int,

    started: one Bool,
    ended: one Bool
}

sig Path {
    starting_point: one Position,
    intermediate_points: set Position,
    ending_point: one Position
}
```

```
// DATA4HELP

// Registration data for the system are unique
fact registrationDataUniqueness {
    // Unique username
    no disjoint u1, u2: RegisteredUser | u1.username = u2.username
    // Unique fiscal code
    no disjoint u1,u2: User | u1.cf = u2.cf
    // Unique VAT
    no disjoint t1,t2: ThirdParty | t1.vat = t2.vat
}

// Request status can have only one of these values: Accepted, Pending or Refused
fact requestConsistency {
    all s: Status | (s = Accepted && s != Pending && s != Refused) ||
            (s != Accepted && s = Pending && s != Refused) ||
            (s != Accepted && s != Pending && s = Refused)
}

// If a request has no sender, it also has no receiver
fact noSenderNoReceiver {
    all r: Request | no r.sender implies no r.receiver
}

fact thirdPartyRequests{
    // Accepted/allowed requests are related to the third party that sent them
    all t: ThirdParty, r: Request | r in t.positive_requests implies r.sender = t

    // All the request related to a third party have been accepted/allowed
    all t: ThirdParty, r: UserRequest | r in t.positive_requests implies r.status = Accepted
    all t: ThirdParty, r: GroupRequest | r in t.positive_requests implies r.allowed.isTrue
}

// If accepted by the user, a user request contains one record of the user who accepted it
fact obtainedDataAfterAcceptance {
    all r: UserRequest | (r.status = Accepted implies one r.obtained_data)
                    and r.obtained_data in r.receiver.data
}
```

```
// No unlinked instances
fact noUnlinked {
    all h: HealthStatus | some r: Record | h = r.health
    all p: Position | some r: Restrictions, rec: Record | p = r.location or p = rec.location
    all r: Restrictions | some g: GroupRequest | r = g.restrictions
    all p: Path | some r: Run | p = r.path
    all d: Description | some e: EmergencyRecord | d = e.description
}

// If a record is not already associated to a user, it can not be available for requests
fact noUserNoRequest{
    all r: Record, ureq: UserRequest, greq: GroupRequest |
        (r not in User.data) implies (r not in ureq.obtained_data and r not in greq.obtained_data)
}


fact uniqueAnonID {
    // Records associated with different users have different anonymous ID
    all u1, u2: User, r1, r2: Record | ((r1 in u1.data) and (r2 in u2.data) and u1 != u2)
                                        implies r1.anon_id != r2.anon_id
    // Records associated with the same user have the same anonymous ID
    all u: User, r1, r2: Record | ((r1 in u.data) and (r2 in u.data)) implies r1.anon_id = r2.anon_id
}

// Records associated with the same user have different timestamps
fact uniqueTimestampUser {
    all u: User, r1, r2: Record | ((r1 in u.data) and (r2 in u.data) and (r1 != r2))
                                    implies r1.timestamp != r2.timestamp
}

// Considering meaningful integer values
fact possibleValues{
    all h: HealthStatus | h.heartBeat > 0 and h.bloodPressure > 0 and h.bodyTemperature > 0
                            and h.stepCounter >= 0
    all p: Position | p.latitude >= -90 and p.latitude <= 90 and p.longitude >= -180
                            and p.longitude <= 180
    all rec: Record | rec.anon_id >= 0
    all res: Restrictions | res.radius > 0 and res.radius < 100 //km
                and res.age_max >= 18 and res.age_max < 110 and res.age_min >= 18 and res.age_min < 110
    all r: Run | r.max_participants > 0 and r.max_participants < 100
                and  #r.enrolled_users > 0 and #r.enrolled_users < 100
                and  r.num_spectators > 0 and r.num_spectators < 100
}
```

```
// TRACK4RUN

// The end of a run must happen after its start
fact TimeConstraints{
    all r: Run | gt[r.end_time, r.start_time]
}

// Organizer and enrolled user must be different
fact OrganizerNotEnrolled {
    all r: Run | r.organizer not in r.enrolled_users
}

// The number of users enrolled for the same run must not exceed the maximum number
// of participants set by the organizator
fact MaxParticipants {
    all r: Run | (#r.enrolled_users >= 0 and #r.enrolled_users <= r.max_participants)
}

// A run can not be finished if it is not started yet
fact noFinishedIfNotSarted {
    all r: Run | (r.ended.isTrue implies r.started.isTrue)
                    and (r.started.isFalse implies r.ended.isFalse)
}

// A run has no spectators if it hasn't started yet or if it has already ended.
fact numSpectators {
    all r: Run | (r.started.isFalse or r.ended.isTrue) implies r.num_spectators = 0
}

// In order for a run to start, there must be at least one user enrolled
fact minPartecipants {
    all r: Run | (r.started.isTrue implies #r.enrolled_users >= 1)
}

// Intermediate points are different from the starting and ending points
fact points {
    all pa: Path | (pa.starting_point not in pa.intermediate_points and
            pa.ending_point not in pa.intermediate_points)
}
```

```
// AUTOMATED SOS

// An emergency record must contain at least one critical parameter
fact emergencyIfNeeded {
    all e: EmergencyRecord | e.health.heartBeat < 45 or e.health.heartBeat > 100 //bpm
                        or e.health.bloodPressure < 50 or e.health.bloodPressure > 160 //mmHg
                        or e.health.bodyTemperature < 33 or e.health.bodyTemperature > 40 //°C
}

// A health status can not be related with both a record and an emergency record
fact healthStatusConsistency {
    all e: EmergencyRecord, r: Record - EmergencyRecord | e.health != r.health
}

// Users with disabled AutomatedSOS setting do not have associated emergency records
fact noEmergencyIfSOSDisabled {
    all u: User | u.automated_sos.isFalse implies u.data - EmergencyRecord = u.data
}
```

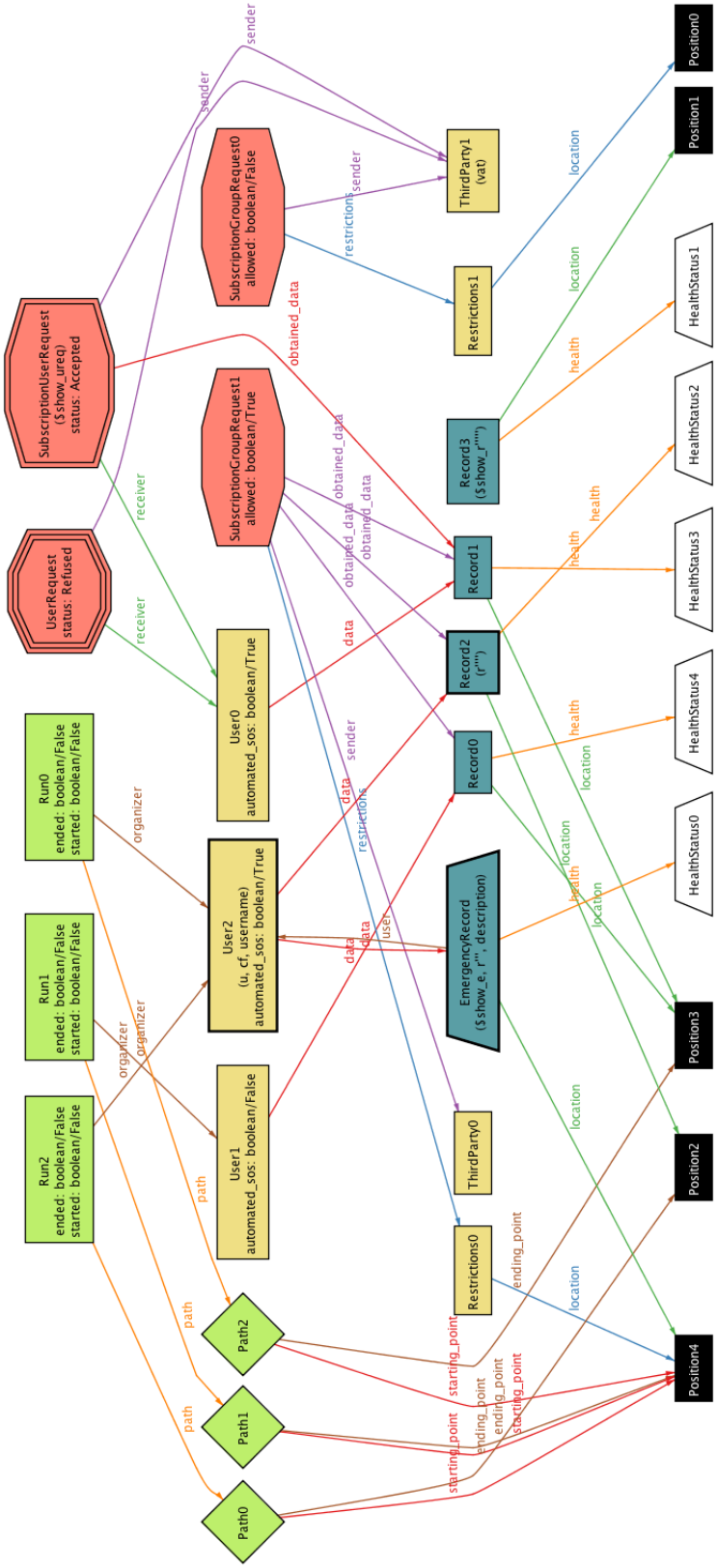
```
// PREDICATES

pred addRecord[r, r': Record, u, u': User] {
    // preconditions
    r' not in User.data
    // postconditions
    #u.data > 0 implies (r in u.data and r'.anon_id = r.anon_id)
    u'.data = u.data + r
}

pred acceptRequest[r, r': UserRequest] {
    // preconditions
    r.status = Pending
    r in SubscriptionUserRequest implies r' in SubscriptionUserRequest
    r in UserRequest - SubscriptionUserRequest implies r' in UserRequest - SubscriptionUserRequest
    // postconditions
    r'.status = Accepted
    r' in r'.sender.positive_requests
}

pred refuseRequest[r, r': UserRequest] {
    // preconditions
    r.status = Pending
    r in SubscriptionUserRequest implies r' in SubscriptionUserRequest
    r in UserRequest - SubscriptionUserRequest implies r' in UserRequest - SubscriptionUserRequest
    // postconditions
    r'.status = Refused
    r' not in r'.sender.positive_requests
}

pred enrollToRun[r, r': Run, u: User]{
    // preconditions
    r.started = False
    #r.enrolled_users < r.max_participants // entries available
    not u in r.enrolled_users
    // postconditions
    r'.started = r.started and r'.ended = r.ended
    r'.enrolled_users = r.enrolled_users + u
    u in r'.enrolled_users
    #r'.enrolled_users <= r.max_participants
    all u': User| u' in r.enrolled_users implies u' in r'.enrolled_users
}
```

```
pred viewLiveRun[r, r': Run]{
    // precondition
    r.started.isTrue and r.ended.isFalse
    // postcondition
    r'.started = r.started and r'.ended = r.ended
    r'.num_spectators = r.num_spectators + 1
}

pred createRun[r: Run, o: User, p: Path, st, et: Time, max: Int]{
    // postconditions
    r.organizer = o
    r.path = p
    r.start_time = st
    r.end_time = et
    r.max_participants = max
    no r': Run | (r' != r and r'.start_time = r.start_time and r'.end_time = r.end_time
                  and r'.path = r.path and r'.max_participants = r.max_participants)
    r.started = False
    r.ended = False
    no u: User | u in r.enrolled_users
}

pred show{

}


run show for 10 but 9 Int

run addRecord for 6

run acceptRequest for 3

run refuseRequest for 3

run enrollToRun for 3

run viewLiveRun for 3

run createRun for 4
```

```
Executing "Run enrollToRun for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  7010 vars. 939 primary vars. 12998 clauses. 61ms.
  Instance found. Predicate is consistent. 49ms.

Executing "Run viewLiveRun for 3"
  Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
  6854 vars. 936 primary vars. 12468 clauses. 56ms.
  Instance found. Predicate is consistent. 49ms.

Executing "Run createRun for 4"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  11312 vars. 1396 primary vars. 20427 clauses. 81ms.
  Instance found. Predicate is consistent. 50ms.
```

```
Executing "Run addRecord for 6"
   Solver=sat4j Bitwidth=4 MaxSeq=6 SkolemDepth=1 Symmetry=20
   22811 vars. 2424 primary vars. 40984 clauses. 183ms.
   Instance found. Predicate is consistent. 163ms.

Executing "Run acceptRequest for 3"
   Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
   6700 vars. 936 primary vars. 11973 clauses. 57ms.
   Instance found. Predicate is consistent. 53ms.

Executing "Run refuseRequest for 3"
   Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
   6700 vars. 936 primary vars. 11968 clauses. 43ms.
   Instance found. Predicate is consistent. 39ms.
```

```
Executing "Run show for 10 but 9 int"
   Solver=sat4j Bitwidth=9 MaxSeq=10 SkolemDepth=1 Symmetry=20
   1914267 vars. 64770 primary vars. 7115458 clauses. 24886ms.
   Instance found. Predicate is consistent. 149031ms.
```


SubscriptionUserRequest
($show_ureq)
status: Accepted
UserRequest
status: Refused
SubscriptionGroupRequest0
allowed: boolean/False
SubscriptionGroupRequest1
allowed: boolean/True
ThirdParty1
(vat)
Restrictions1
Record3
($show_r"")
Record1
Record2
(r"")
Record0
User0
automated_sos: boolean/True
User2
(u, cf, username)
automated_sos: boolean/True
User1
automated_sos: boolean/False
Run0
ended: boolean/False
started: boolean/False
Run1
ended: boolean/False
started: boolean/False
Run2
ended: boolean/False
started: boolean/False
EmergencyRecord
($show_e, r"", description)
ThirdParty0
Restrictions0
Path0
Path1
Path2
Position0
Position1
Position2
Position3
Position4
HealthStatus0
HealthStatus1
HealthStatus2
HealthStatus3
HealthStatus4
sender
receiver
obtained_data
restriction
data
user
organizer
path
location
health
starting_point
ending_point

Chapter 5

Effort Spent

- Giulia Mangiaracina:
  - [17/10/2018]Team Reunion: 1h
  - [19/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [24/10/2018]: 1h
  - [25/10/2018]Team Reunion: 2h
  - [28/10/2018]Team Reunion: 4h
  - [30/10/2018]: 2h
  - [31/10/2018]Team Reunion: 2h
  - [2/11/2018]Team Reunion: 5h
  - [4/11/2018]: 3h
  - [5/11/2018]Team Reunion: 4h
  - [6/11/2018]: 2h
  - [7/11/2018]Team Reunion: 5h
  - [8/11/2018]Team Reunion: 2h
  - [9/11/2018]: 1h

  Total: 40 h
- Andrea Miotto:
  - [17/10/2018]Team Reunion: 1h
  - [20/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [24/10/2018]: 1h
  - [25/10/2018]Team Reunion: 2h
  - [26/10/2018]: 2h
  - [28/10/2018]Team Reunion: 4h

- [31/10/2018]Team Reunion: 2h
- [2/11/2018]Team Reunion: 5h
- [5/11/2018]Team Reunion: 4h
- [6/11/2018]: 2h
- [7/11/2018]Team Reunion: 5h
- [8/11/2018]Team Reunion: 2h
- [9/11/2018]: 3h
- [10/10/2018]: 1h

Total: 40 h

- Ilaria Moschetto:
  - [17/10/2018]Team Reunion: 1h
  - [19/10/2018]: 1h
  - [22/10/2018]Team Reunion: 2h
  - [23/10/2018]Team Reunion: 3h
  - [25/10/2018]Team Reunion: 2h
  - [26/10/2018]Team Reunion: 2h
  - [27/10/2018]: 1h
  - [28/10/2018]Team Reunion: 4h
  - [30/10/2018]: 2h
  - [31/10/2018]Team Reunion: 2h
  - [1/11/2018]: 1h
  - [2/11/2018]Team Reunion: 5h
  - [4/11/2018]: 2h
  - [5/11/2018]Team Reunion: 4h
  - [7/11/2018]Team Reunion: 5h
  - [8/11/2018]Team Reunion: 2h
  - [9/11/2018]: 1h

  Total: 40 h