POLITECNICO
MILANO 1863

TrackMe

Software Engineering II - Prof. Elisabetta Di Nitto

Requirements Analysis and Specification Document

Michele Gatti, Federica Gianotti, Mathyas Giudici

Document version: 2.0

December 8, 2018

Deliverable:	RASD
Title:	Requirement Analysis and Specification Document
Authors:	Michele Gatti, Federica Gianotti, Mathyas Giudici
Version:	2.0
Date:	December 8, 2018
Download page:	https://github.com/MathyasGiudici/GattiGianottiGiudici
Copyright:	Copyright © 2018, Michele Gatti, Federica Gianotti, Mathyas Giudici – All rights reserved

Contents

Section 1

Introduction

1.1 Purpose

The goal of the Requirement Analysis and Specification Document (RASD) is to give a clear description of the system that is going to be developed, its functional and non-functional requirements, its constraints and its domain. Moreover, it provides information about the relationship between the system taken into account and the external world by providing use cases and scenarios. Finally it gives a more formal specification of the most relevant features of the system to be using the Alloy language. Generally this type of document is mainly addressed to developers, programmers, testers, project managers and system analysts, but it can be useful also for final users. Track Me is a company that wants to develop three different but connected software-based services:

- **Data4Help**: a service that allows third parties to monitor the location and health status of individuals. Through this service third parties can request the access both to the data of some specific individuals, who can accept or refuse sharing their information , and to anonymized data of group of individuals, which will be given only if the number of the members of the group is higher than 1000, according to privacy rules.

- **AutomatedSOS**: a service addressed to elderly people which monitors the health status of the subscribed customers and, when such parameters are below a certain threshold (personalized for every user using the data from Data4Help), sends to the location of the customer an ambulance, guaranteeing a reaction time less than 5 second from the time the parameters are below the threshold.

- **Track4Run**: a service to track athletes participating in a run. It allows organizers to define the path for a run, participants to enrol to a run and spectators to see on a map the position of all runners during the run. This service will exploit the features offered by Data4Help.

1.1.1 Goals

The three applications of the system have in common the following goals:

- **[G.1]**: Allow unregistered user to sign in to access to the application;

- **[G.2]**: Allow registered user to log in and access to the application;

- **[G.3]**: Allow registered user to manage his/her profile;

The description given above can be summarized as a list of goals, specific for each service.

Data4Help:

- **[G.4]**: Allow registered third parties to request data of a single individual;

- **[G.5]**: Allow registered third parties to request data of a group of people;

AutomatedSOS and Track4Run:

- **[G.6]**: Allow data acquisition through smartwatches (or similar);

AutmatedSOS:

- **[G.7]**: Allow monitoring the health status of an individual registered user;

- **[G.8]**: Allow sending location of an individual registered user to an ambulance if his/her parameters are below a certain threshold;

Track4Run:

- [**G.9**]: Allow registered user to become organizers or athletes of a run;

- [**G.10**]: Allow organizers to define the date and the path for a new run;

- [**G.11**]: Allow organizers to delete a run;

- [**G.12**]: Allow registered athletes to enrol in a run;

- [**G.13**]: Allow registered athletes to delete an enrolment of a run;

- [**G.14**]: Allow unregistered user to access as spectator;

- [**G.15**]: Allow registered/unregistered user to see on a map the position of all runners during a run;

1.2 Scope

According to *The World and the Machine* [3] we can divide every system into two parts:

- The **machine**, which is the portion of system to be developed;

- The **world**, which is the portion of the real-world affected by the machine.

As a consequence we can classify phenomena in three different types:

- **World phenomena**: phenomena that the machine cannot observe;

- **Machine phenomena**: phenomena located entirely in the machine;

- **Shared phenomena**: phenomena that can be controlled by the world and observed by the machine or controlled by the machine and observed by the world;

Below we give an analysis of world and shared phenomena:

World phenomena

- A user turns on data connection;

- A user wears his smartwatch during a day;

- The batteries of the smartwatch of a user run out;

- A user turns on the GPS;

- An enrolled runner for a run takes part in it;

- A runner wears his smartwach during a run.

Shared phenomena

- New user registeres to Data4Help service;

- A Data4Help registered user logs into the system;

- A user recives a request to share his data;

- A user accept/decline a request to share his data;

- A third party requires data of a specific user;

- A third party requires data of a group of users;

- A user subscribes to AutomatedSOS service;

- An ambulance is called as a consequence of specific acquired data from the system;

- A Data4Help user access to Track4Run for the first time;

- A Track4Run user organizes a new run;

- A Track4Run user enrols for a run;

- An unregistered user access as a spectator to a run.

Machine phenomena

- The machine interfaces with external software and hardware systems;

- The machine manages database queries;

- The machine manages the 3G/4G Internet connection;

- The machine manages the Bluetooth connection;
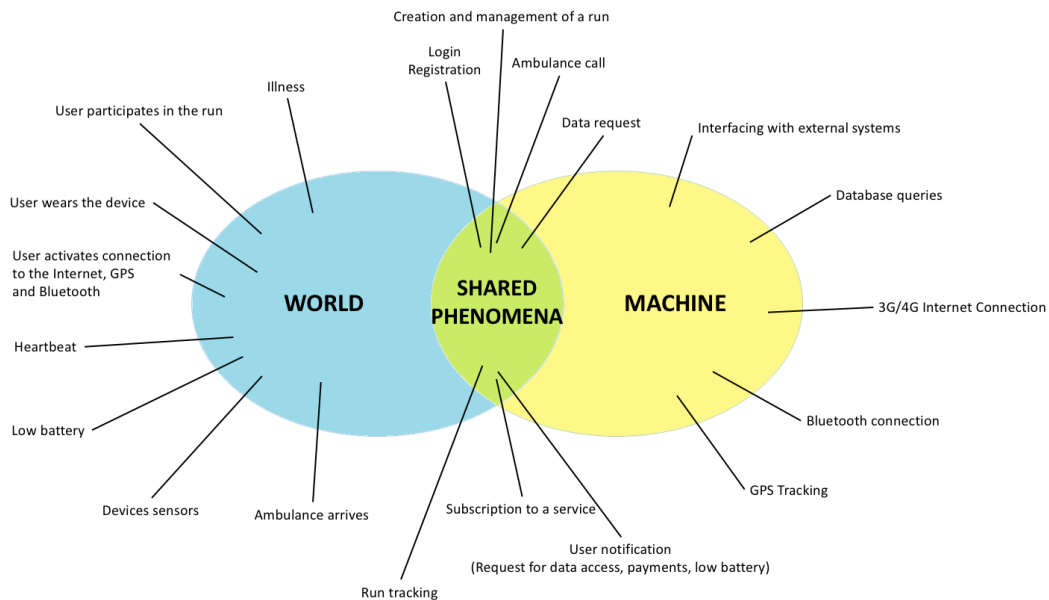
- The machine manages GPS tracking.

Figure 1.1: World and Machine model

1.3   Definitions, Acronyms and Abbreviations

API: Application Programming Interface;

DD: Design Document;

GPS: Global Positioning System;

Organizer: A registered user that organizes a run, defining date and path;

OS: Operating System;

RASD: Requirement Analysis and Specification Document;

Run: An event that is organized by one organizer, at which one or more people can partecipate and that can be followed by one or more spectators;

Runner: A registered user that enrols for a run;

Spectator: Unregistered user that access to Track4Run to follow a run;

SSN: Social Security Number;

System: The software system-to-be, including all of its services;

Third party: Any external organization that wants to access to data acquired by Data4Help;

UML: Unified Modeling Language;

User: Any person, registered or not, who accesses to one of the applications (for Data4Help there is a special user called *Third party*);

VAT: Value Added Tax.

1.4   Reference documents

This document is written according to the assignment for the Software Engineering 2 project [4] of teacher Elisabetta Di Nitto.
Moreover, this document follows ISO/IEC/IEEE 29148:2011 [1] and IEEE 830:1998 [2] standard for software product specificications.

1.5   Overview

This document is structured as follows:

Section 1: Introduction. A general introduction to the goals, the phenomena and the scope of the system-to-be. It aims giving general but exaustive information about what this document is going explain.

Section 2: Overall Description. A general description of the product to be and its requirements. This section provides several information that are detailed explained in Section 3.

Section 3: Specific Requirements. All software requirements are explained using scenarios, use-case diagram and activity diagram. Nonfunctional and functional requirements are also cited.

Section 4: Alloy. This section includes Alloy code that describes the model and checks wether it is consistent or not.

Section 5: Effort Spent. A summary of the worked time by each member of the group.

At the end there are an **Appendix** and a **Bibliography**.

Section 2

Overall Description

2.1 Product Perspective

2.1.1 User Interfaces

Standard Users

Standard users can use two different smartphone applications: *Automated-SOS* and *Track4Run*. Both of them should be very easy to use and should allow the user to connect the smartphone to the smartwatch or to the chosen device. *AutomatedSOS* is mainly used by elderly people so it should have large buttons and large writing and it shouldn't ask to the user to interact a lot with the device. *Track4Run* is mainly used by young people so it should be more interactive. Standard users can also access services provided by TrackMe using a web application. Using it they can manage their accounts in a more comfortable way, verify requests for accessing their data, create new route and follow a race watching players position on the map (in *Track4Run* service).

Special Users

Third parties who want to analyse data collected from *Data4Help* can access the service using a web application. The web application lets special users to insert a request for data. If the request is accepted, it allows the download of the asked data. The system should also offer an online support to help user in using the service.

2.1.2 Hardware Interfaces

- Web applications (both the one for standard users and the one for special users) must be accessible using a computer with characteristic specified in Section 2.4.4.

- Smartphone on which the app will work must provide to the app an Internet connection used to send data to TrackMe servers and must have a GPS antenna built in. The wearable device must also integrate a reasonably precise heartbeat sensor and a pressure sensor.

2.1.3 Software interfaces

- Web applications (both the one for standard users and the one for special users) must be compatible with the most popular browsers such as Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari;

- Mobile apps for standard users must be available for both iOS and Android devices and must be compatible with most of the smartwatch and other health devices available on the market regardless of the operating system used by the device (using the API made available to programmers by producers);

- Application backend stores collected data in a relational DBMS;

- Web applications show data by accessing the relational DBMS;

- Web applications for third parties has to interface also with a payments broker in order to receive money from companies who want to get data from the system;

- Web application and Track4Run have to interface also with Maps in order to generate the path for the run and to virtually follow a run;

- *AutomatedSOS* has to interface with ambulance call external service.

2.2   Product Functions

The system is composed by several applications.

AutomatedSOS

AutomatedSOS is designed for elderly people and offers a feature that makes an automatic call for help if it detects a dangerous state of health. To use this app, few user interactions are required. In particular the user can:

- Register to the service;

- Log-in to the service;

- Respond to requests to access to his/her personal data by a third party;

- Manage personal account and send a request to delete all the acquired data;

- Connect an health device such as smartwatch, smart band, heart rate sensors with Bluetooth;

- Pause data monitoring.

The app will autonomously monitor the health status of the user and make an emergency call to the nearest ambulance in case of emergency.

Track4Run

Track4Run is designed to track athletes participating in a run. Using it the user can:

- Register to the service;

- Log-in to the service;

- Respond to requests to access to his/her personal data by a third party;

- Manage his/her personal account;

- Connect an health device such as smartwatch, smart band, heart rate sensors with Bluetooth;

- Pause data monitoring.

The app will autonomously track the health status and the position of the athlete.

Web application for standard users

Using it they can:

- Register to the service;

- Log-in to the service;

- Respond to requests to access to their personal data by a third party;

- Manage personal account;

- Crate a path to be used in *Track4Run*;

- Follow a competition watching the position of the athletes on the map.

Web application for special users

Using it they can:

- Register to the service;

- Log-in to the service;

- Send a request to access to the data of a standard user;

- Send a request to access to the data of a group of people;

- Manage past requests;

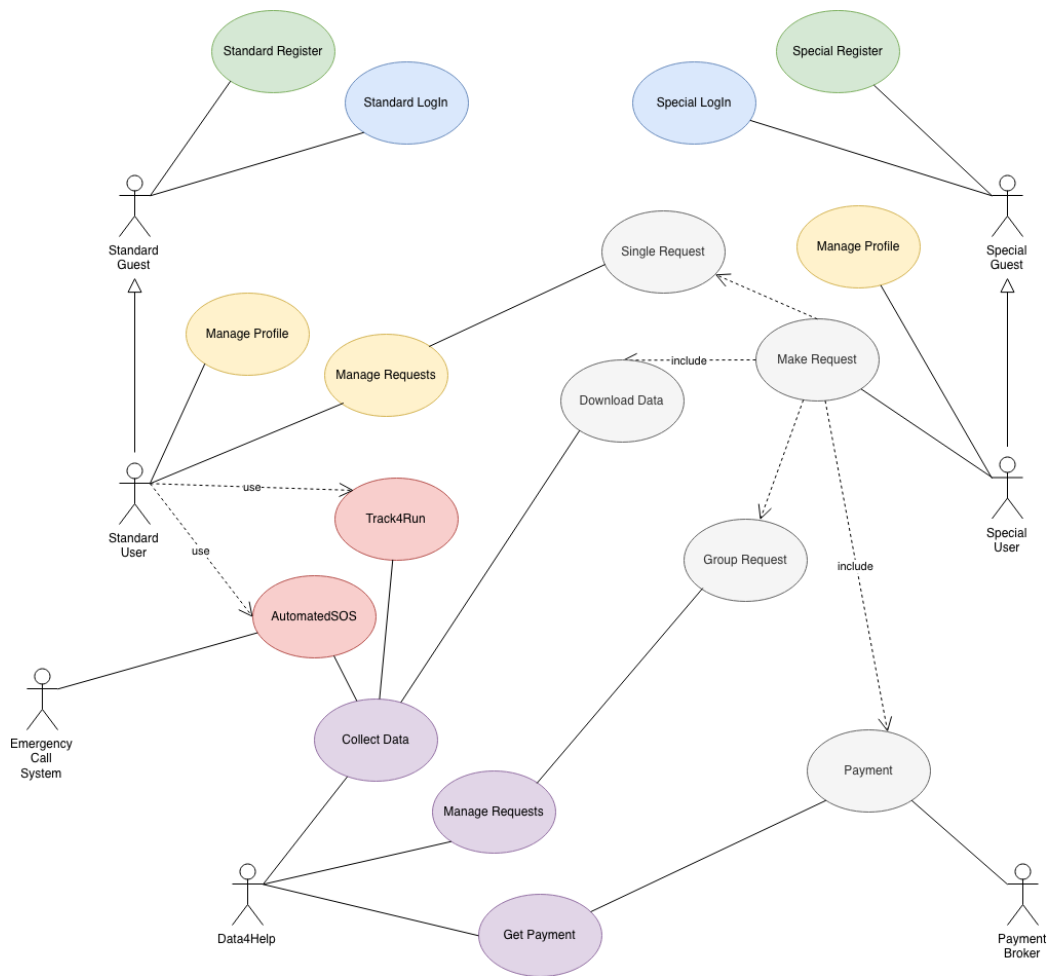- Download data obtained after a request has been accepted and paid.

Figure 2.1: Use Case Diagram

2.3 User Characteristics

Those applications have different targets.

AutomatedSOS

This mobile app is thought for elderly people. It is not necessary that the user is a "tech addicted" because a familiar can setup the system for him/her and than it will work autonomously.

Track4Run

This mobile app is thought for athletes. It is most dedicated to young people who use frequently tech products.

Third party WebApp

This application is thought for companies who want to analyse data collected by the app. They could be statistics or pharmaceutical companies, hospitals, etc.

2.4   Constraints

2.4.1   Anonymous data collection

Companies who want to analyse data from a group of people without asking the permission to every single person must make a request for anonymized data of a group of at least 1000 people.

2.4.2   Privacy

Before allowing a company to access to user's data, it is necessary to get a formal permission by the user.

2.4.3   Regulatory Policies

When a new user registers to the service he must accept the privacy policy in order to use the application. He must be informed about personal and sensible data collection that is carried out by the applications (his/her position and his/her health parameters). All the collected data must be kept safe and must not be accessible by unauthorized person. Also, third parties who access the service must guarantee the security of the data. The whole process must comply with the GDPR regulations for the protection of users' personal data.

2.4.4 Hardware limitations

In order to use the service, user's hardware should comply to these minimum requirements:

Mobile application

- Smartphone:

    - iOS or Android operative system;
    - UMTS/4G Internet connection with a minimum speed of 1Mb/s;
    - Bluetooth antenna;
    - GPS antenna;
    - 300 Mb available memory;
    - Dual-core processor;
    - 1 Gb RAM.

- Smartwatch / other health device:

    - Bluetooth antenna;
    - Heartbeat sensor;
    - Pressure sensor.

Web application

- Computer:

    - Internet connection with a minimum speed of 1Mb/s;
    - Browser application;
    - 720p monitor resolution.

2.4.5 Parallel operation

The system must guarantee the operation of the system in case of simultaneous use of the mobile app by at least 100,000 users and simultaneous use of the web app by at least 10,000 users. Consequently, the DBMS must be able to process a large number of simultaneous transactions.

2.4.6 Reliability requirement

The system reliability, seen as the probability that components and performances will meet the requirement during a specified period of time, must be at least 95%.

2.4.7 Availability requirement

The system should be available 24/7 in order to guarantee the services and manage emergency situations.

2.4.8 Criticality of the system

AutomatedSOS

An error in the system could result in an unnecessary call for an ambulance or in a failure to call an ambulance.

Track4Run

An error in the system could cause a non-optimal use of the service.

2.5 Assumptions and Dependencies

From now on the following assumptions are given for guaranteed:

1. Users of the app have a phone with an iOS or Android operating system;

2. Users of the app have a phone with working GPS module with an uncertain of ± 1 meter;

3. All users enjoy a stable Internet connection;

4. Users accept or refuse a request for access to their data within 24 hours from when it has been recievd. After this period the request is considered rejected;

5. Each user is identified with his/her/its e-mail;

6. Once a request is accepted by a user, the system provides the applicant with the required data within 24 hours;

7. The user autonomously recharges the smartphone and the smartwatch when its battery is low;

8. A user cannot participate in two races at the same time;

9. Every request for access to user's personal data from a third parties must be explicitly accepted by the user;

10. When a third party accesses to the personal data of a user it is responsible for any unauthorized disclosure of data.

Behavioural during devices recharge time:

1. If the smartphone is charging in a fixed position near the user, the wearable device remains connected via Bluetooth to the smartphone and the *AutomatedSOS* service is not interrupted;

2. If the smartphone is charging in a fixed position, the *Track4Run* service is interrupted;

3. If the smartphone is charging using a transportable battery, all the services keep working;

4. If the wearable device is charging, the service *AutomatedSOS* is interrupted;

5. If the wearable device is charging, the service *Track4Run* keep working (available only data about position);

2.6   Future Extensions

The addition of new features will be evaluated in the future. The possible proposals are:

1. The creation of a new application with the purpose of collecting user data, without offering services such as *AutomatedSOS* and *Track4Run*. Users will be paid to use the application consistently. This operation considerably increases the number of users and also collects data from those subjects that are currently excluded (people who are not elderly and do not practice sports);

2. The possibility for the user to share their sanitary facilities for free with their doctor.

Section 3

Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

The user interfaces of *AutomatedSOS* and *Track4Run* must be intuitive and user-friendly in order to permit an easy interaction with all the services offered by the systems. The UI must be developed according to the three-click rule.
Moreover both the application and the web site must support multiple languages.
The Standard User and Special Users experiences are explained in Section 2.1.1.

Figure 3.1: *Data4Help*, *AutomatedSOS* and *Track4Run* logo

3.1.2 Hardware Interfaces

The hardware interfaces of the system are huge explained in Section 2.1.2.

3.1.3 Software Interfaces

The software interfaces of the system are huge explained in Section 2.1.3.

3.1.4 Communication Interfaces

The connection between clients and server and also the connection between server and payment handler must be done with the HTTPS protocol.
In order to manage and visualize *Run*, the system must be connected with Google API to use GoogleMaps services.

3.2 Functional Requirements

3.2.1 Individual Sign In

Purpose

Anyone who wants to subscribe to one or both services offered by *Data4Help* must go through the registration process, which can be carried out either through *AutomatedSos* and *Track4Run* apps or through the web site. The process requires exactly the same steps regardless the platform through which it is carried out:

1. The new user is required to fill in all the fields in which he/she is asked for his/her name, his/her surname, his/her date of birth, his/her city of birth, his/her city of residence, his/her address, his/her occupation, and a valid e-mail address;

2. The user must accept the conditions regarding his/her privacy, in particular about the collection of his/her data by *Data4Help* and the sharing of them in anonymous way with third parties.

After that the system will check the correctness of the inserted data, in particular it will check that the user isn't already registerd and that the inserted e-mail isn't already used by someone else. If the result of this control is positive the registration is authorized and the user will receive a confermation e-mail to the specified e-mail address with the password he/she has to use to access to all *Data4Help* services.

Scenario 1

 Sara would like to register her grandmother to *AutomatedSos* to not worry about her helth status when they are not together. She opens the browser on her personal computer and search for *Data4Help* web site, then she clicks on the "*Sign In*" button, which is located in the main page. She passes through the steps of the registration process, inserting her grandmother data and accepting the required conditions. Finally, if the inserted data are accepted by the system, she receives the confirmation e-mail.

Scenario 2

Marco wuold like to organize an amateur run with his friends and remembers that someone told him something about a new application called *Track4Run* so he decides to try it. He downloads the app on his smartphone and turns it on. The first page that is shown to him contains the "*Sign In*" button and the "*Log In*" one, he presses on the first one and starts his registration

process. He doesn't use his personal e-mail address, but an e-mail address he has in common with his brother that they usually use to make purchase online. Unexpectedly he is informed by the system that the insert e-mail is already registered in the database and so he has to change it and this time he inserts his personal e-mail address. This time the registration is succesfull and he receives the confermation e-mail.

Use Case
The *Individual Sign In* use case is analyzed in Table 3.1.

Activity Diagram
The *Individual Sign In* activity diagram is shown in Figure 3.2.

Mockup
The *Individual Sign In* mocukp is shown in Figure 3.3.

Functional requirements

1. The system must not accept an e-mail address that is already used by an already registered user;

2. The system must not authorize the registration untill all the fields are filled up;

3. The system must not authorize the registration untill the required conditions aren't accepted;

4. The system must send the confirmation e-mail to the inserted e-mail address with the password when "*Submit*" button is clicked only if all the inserted data are acceptable and the required conditions has been accepted;

5. The system must let the **Individual user to be** leave the registration process at anytime.

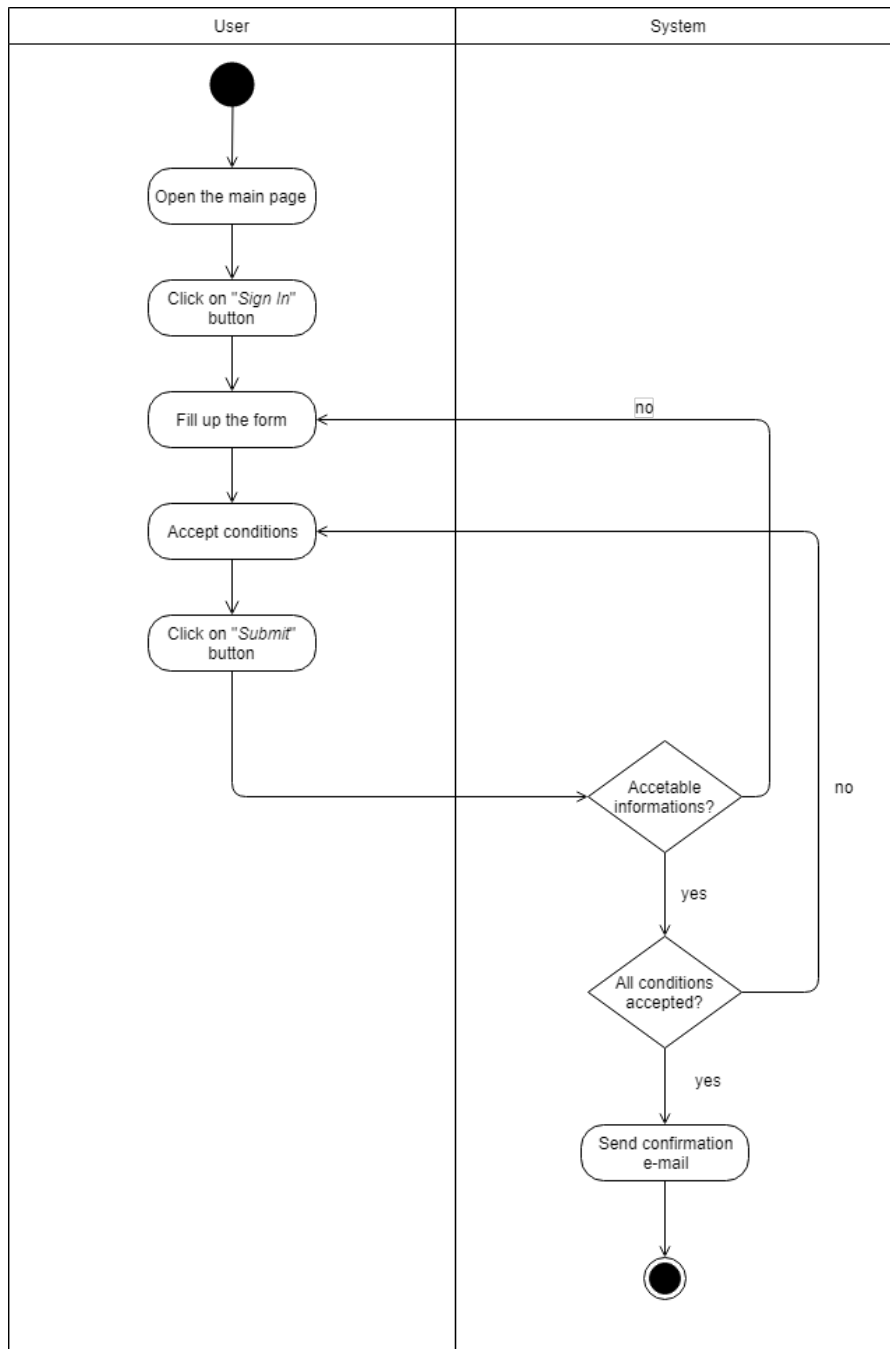| Actor	**Individual user to be**
Goal	**[G.1]**
Input Condition	A person wants to subscribe to one of *Data4Help* services
Event Flow	1. The **Individual user to be** opens the main page of *Data4Help* web site from his/her personal computer or of *AutomatedSos* or *Track4Run* apps from his smartphone; 2. The **Individual user to be** clicks on the "*Sign in*" button; 3. The system shows the form the **Individual user to be** has to fill up; 4. The **Individual user to be** fills up the form with his/her name, his/her surname, his/her date of birth, his/her city of birth, his/her city of residence, his/her address, his/her occupation, and a valid e-mail address; 5. The **Individual user to be** accepts the required conditions; 6. The **Individual user to be** clicks on the "*Submit*" button; 7. The system checks wheter the inserted information are acceptable or not; 8. The **Individual user to be** receives a confirmation e-mail containing the password he/she has to use to access to all *Data4Help* services.
Output Condition	The system tells the **Individual user to be** that his/her registration is completed
Exceptions	• If functional requirements 1 or 2 are not satisfied the process goes back to step 4; • If functional requirement 3 is not satisfied the process goes back to step 5; • If the **Individual user to be** decides to leave the registration process this one is aborted.

Table 3.1: *Individual Sign In* use case

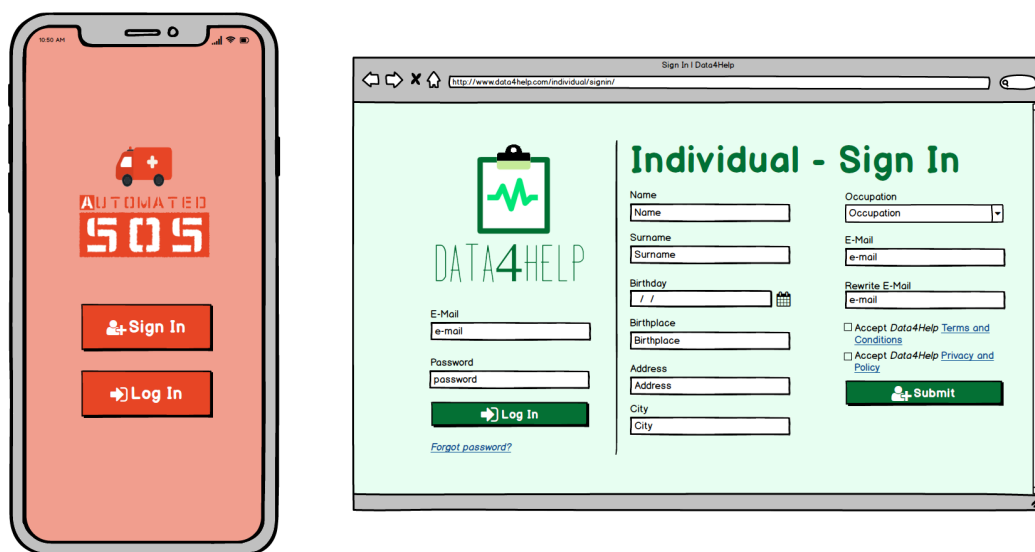Figure 3.2: *Individual Sign In* activity diagram

Figure 3.3: *Individual Sign In* mockup

3.2.2 Third Party Sign In

Purpose

Any third party who wants to subscribe to *Data4Help* must go through the registration process, which can be carried out through *Data4Help* web site. The process requires several mandatory steps:

1. The third party which aim to become a new member is required to fill in all the fields in which it is asked for information about the company itself like: its business name, its VAT number, its legal address, its billing address, its corporate e-mail address and the sector in which it operates;

2. The third party must also provide the data of its legal representative, in particular his/her name, his/her surname, his/her office address, his/her phone number, his/her e-mail address and his/her SSN;

3. The third party must accept different conditions:

   - It must assume responsibility in case of unauthorized disclosure of user data;

   - It must accept Milan as the place of jurisdiction in the case of a legal dispute.

After that the system will check the correctness of the inserted data, in particular it will check that the third party isn't already registered. If the result of this control is positive the registration is authorized and the third party will receive a confirmation e-mail to the specified e-mail address with the password it has to use to access to *Data4Help* services. From now we will refer to the third party that wants to become a new member as "**Special user to Be**" to distinguish it from an Individual user.

Scenario 1

PharmaAnalisi SPA wants to acquire data of a group of young people in order to do an analysis about the kind of life they conduct. It opens the browser and search for *Data4Help* web site, then it clicks on the "*Third Party Sign In*" button, which is located in the main page. It passes through the steps of the registration process, inserting all the required data but forgetting to accept one of the conditions. As a consequence the system won't permit it to conclude the registration process, so it checks again and figures out what was missing, it accepts the condition and submits its registration. Finally, it receives the confimation e-mail.

Use Case
The *Third Party Sign In* use case is analyzed in Table 3.2.

Activity Diagram
The *Third Party Sign In* activity diagram is shown in Figure 3.4.

Mockup
The *Third Party Sign In* mocukp is shown in Figure 3.5.

Functional requirements

1. The system must not accept an e-mail address that is already used by an already registered third party;

2. The system must not accept a business name that is already used by an already registered third party;

3. The system must not accept a VAT number that is already used by an already registered third party;

4. The system must not authorize the registration untill all the fields are filled up;

5. The system must not authorize the registration untill the required conditions aren't accepted;

6. The system must send the confirmation e-mail to the inserted e-mail address with the password when "*Submit*" button is clicked only if all the inserted data are acceptable and the required conditions has been accepted;

7. The system must let the **Special user to be** leave the registration process at anytime.

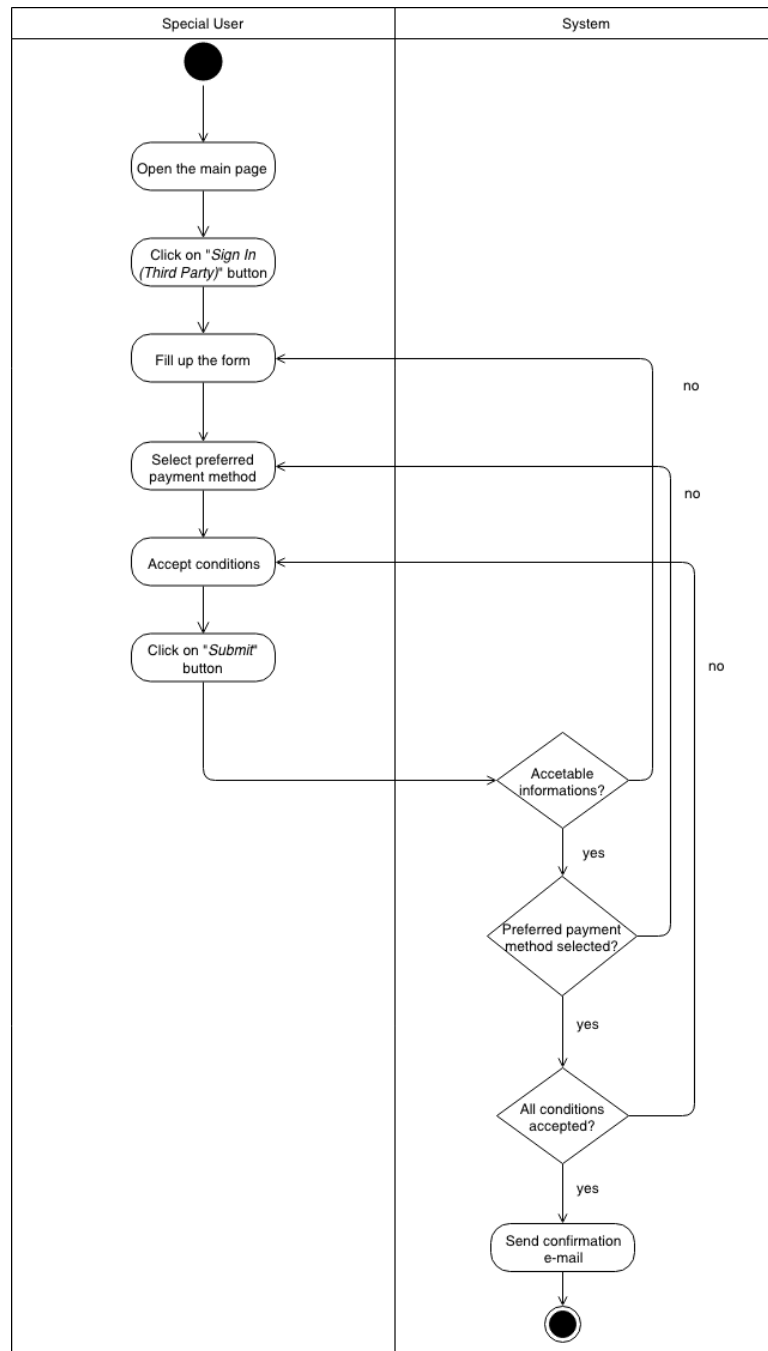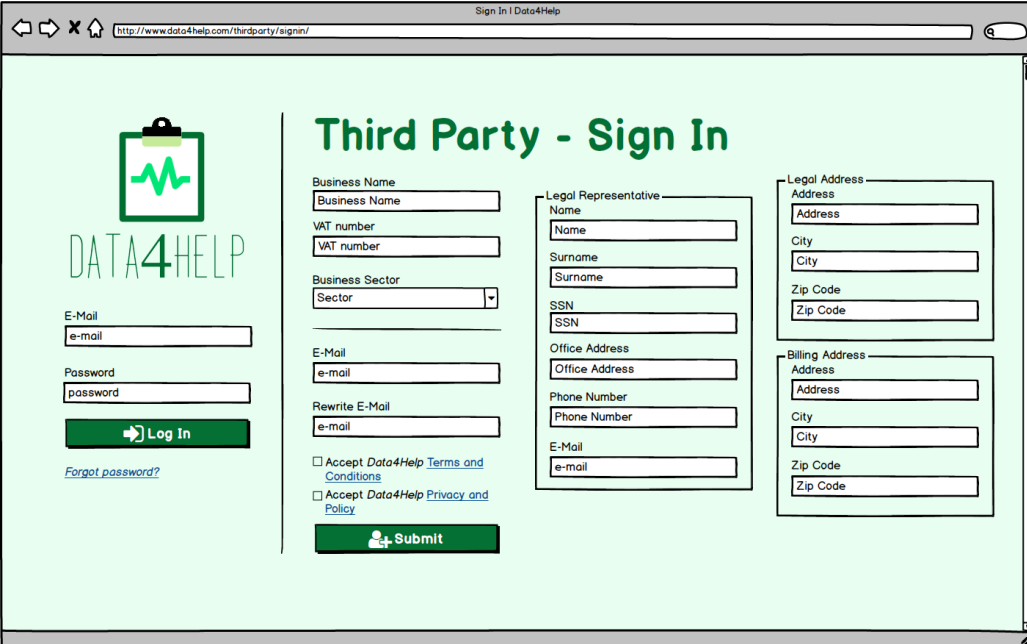|	
Actor	**Special user to be**
Goal	**[G.1]**
Input Condition	A third party wants to subscribe to *Data4Help* services
Event Flow	1. The **Special user to be** opens the main page of *Data4Help* web site. 2. The **Special user to be** clicks on "*Sign in (Third party)*" button; 3. The system shows the form the **Special user to be** has to fill up; 4. The **Special user to be** fills up the form with its business name, its VAT number, its legal address, its billing address, its corporate e-mail address and the sector in which it operates; 5. The **Special user to be** accepts the required conditions; 6. The **Special user to be** clicks on "*Submit*" button; 7. The system checks wheter the inserted information are acceptable or not; 8. The **Special user to be** receives a confirmation e-mail containing the password it has to use to access to *Data4Help* services.
Output Condition	The system tells the **Special user to be** that its registration is completed
Exceptions	• If functional requirements 1,2,3 or 4 are not satisfied the process goes back to step 4; • If functional requirement 5 is not satisfied the process goes back to step 5; • If the **Special user to be** decides to leave the registration process this one is aborted.

Table 3.2: *Third Party Sign In* use case

Figure 3.4: *Third Party Sign In* activity diagram

Figure 3.5: *Third Party Sign In* mockup

3.2.3   Individual Log In

Purpose
The main goal of the login feature is to allow the access to one of the services of *Data4Help* to any registered user. To access to an application the user has to fill out the credential form where e-mail and password are required. Moreover, there is a *Forgot password?* section where a user could recover his/her password via e-mail. An e-mail is sent to the user with the new password that the user could change once logged in.

Moreover, at the first login in one of the application the *Individual user* must associate to the system one device (like smartwatch or similar) to allow the system to trace his/her data. This process is very important in *Automated-SOS* application.

Scenario 1
Francesca loves running. When she has heard about *Track4Run* application she downloaded it immediately. Her friend Clara told her about a charity run for the following weekend, so Francesca opened *Track4Run*, she clicked on the "*Log in*" button. She inserted her e-mail address and password and clicked on the "*Log in*" button. Everything was correct, so she entered in the system and enrolled in the run.

Scenario 2
One year ago Tommaso, Aldo's grandchild, installed *AutomatedSOS* on Aldo's phone. Yesterday Aldo bought a new phone, he downloaded the app but he forgot his password so he couldn't log in the application. He clicked on the "*Forgot password?*" button, he inserted his e-mail address and clicked on the "*Restore my password*" button. He received an e-mail with a new password and he became able to access to the system.

Scenario 3
After that Sara helped her grandmother to register to *Data4Help*, that we esplained in Section 3.2.1, now she also helps her to do the first log in *AutomatedSOS*. After a succesfull login, Sara has to match the grandma's smartwatch with the application. *AutomatedSOS* has a wizard to help users: the system first asks her to turn on the bluethooth on both devices (smartphone and phone), then it shows on the screen the associable devices. Sara selects the device she wants to connect and clicks on the *Done* button. After that smartwatch and phone are matched and *AutomatedSOS* is able to watch the health status of Sara's grandmother.

Use Case
The *Generic Individual Log In* use case is analyzed in Table 3.3.
The *First Individual Log In* use case is analyzed in Table 3.4.

Activity Diagram
The *Generic Log In* activity diagram is shown in Figure 3.6. The *First Log In* activity diagram is shown in Figure 3.8.

Mockup
The *Generic Log In* mockup is shown in Figure 3.7.
The *First Log In* mockup is shown in Figure 3.9.

Functional requirements

1. The **Individual user** must be already registered in the system in order to log in successfully;

2. The **Individual user** has to remember his/her e-mail address and password in order to log in successfully;

3. The password inserted by the **Individual user** must correspond with the e-mail address;

4. If the **Individual user** inserts wrong credential could not be able to access to the system;

5. If the **Individual user** clicks on the "*Forgot password?*" button, the system sends a new password to the **Individual user** e-mail address if and only if the e-mail address is valid and registered to the system;

6. The system must let the **Individual user** leave the login process at anytime;

Actor	**Individual user**
Goal	**[G.2]**
Input Condition	The **Individual user** is already registered to the system and want to log in
Event Flow	1. The **Individual user** opens the main page of *Data4Help* web site from his/her personal computer or of *AutomatedSos* or *Track4Run* apps from his smartphone; 2. The **Individual user** clicks on the "*Log In*" button; 3. The **Individual user** fills in the fields with his/her e-mail address and his/her password; 4. The **Individual user** clicks on the "*Log In*" button.
Output Condition	The system allows the login of the **Individual user** and loads his/her dashboard.
Exceptions	• If functional requirements 1 or 3 are not satisfied the system notifies the **Individual user** with an error message and the process goes back to step 3; • If the **Individual user** inserts wrong credentials for three times the system notifies him/her with an e-mail; • If the **Individual user** decides to leave the login process this one is aborted.

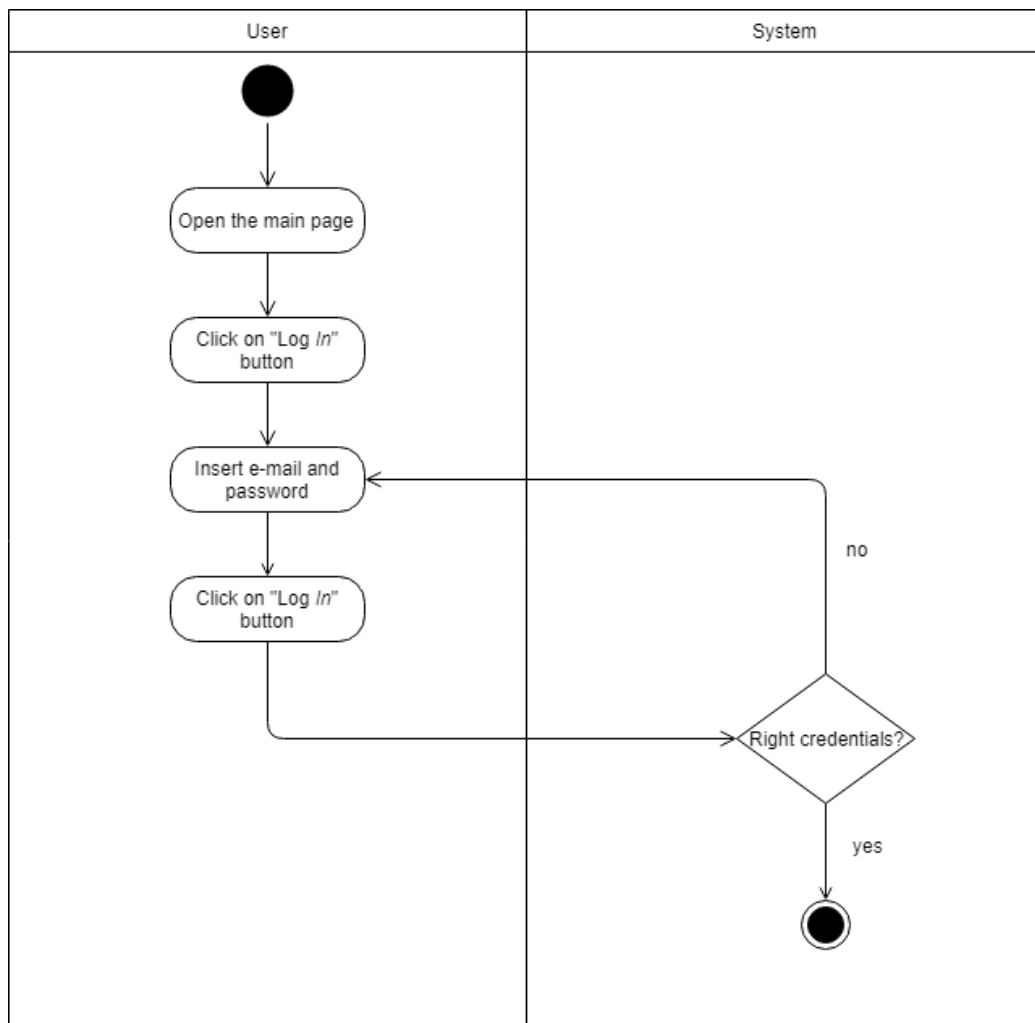Table 3.3: *Generic Individual Log In* use case
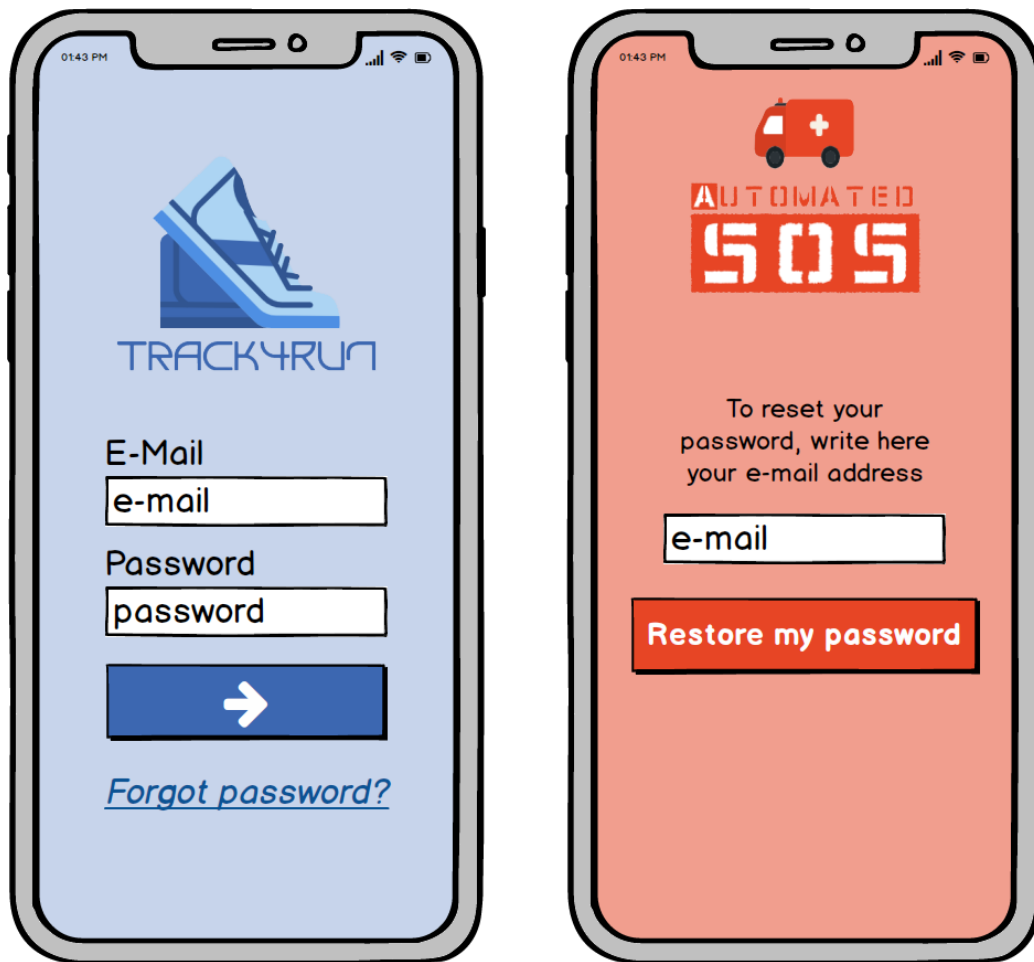
Figure 3.6: *Generic Log In* activity diagram

Figure 3.7: *Generic Log In* mockup

|	
Actor	**Individual user**
Goal	**[G.2]**
Input Condition	The **Individual user** is already registered to the system and wants to log in
Event Flow	The first part of the event flow is already explained in Table 3.3. 1. The system asks the **Individual user** to turn on the bluetooth of the smartwatch and of the smartphone; 2. The **Individual user** turns on the bluethooth; 3. The system shows on the smartphone display the associable devices that it finds with the bluetooth connection; 4. The **Individual user** selects the device he wants to associate. 5. The **Individual user** clicks on "*Done*" button.
Output Condition	The system allows the **Individual user** to log in and loads his/her dashboard.
Exceptions	All already explained exceptions in Table 3.3 are still valid. • If the **Individual user** decides to leave the connecting device process this one is aborted.

Table 3.4: *First Individual Log In* use case
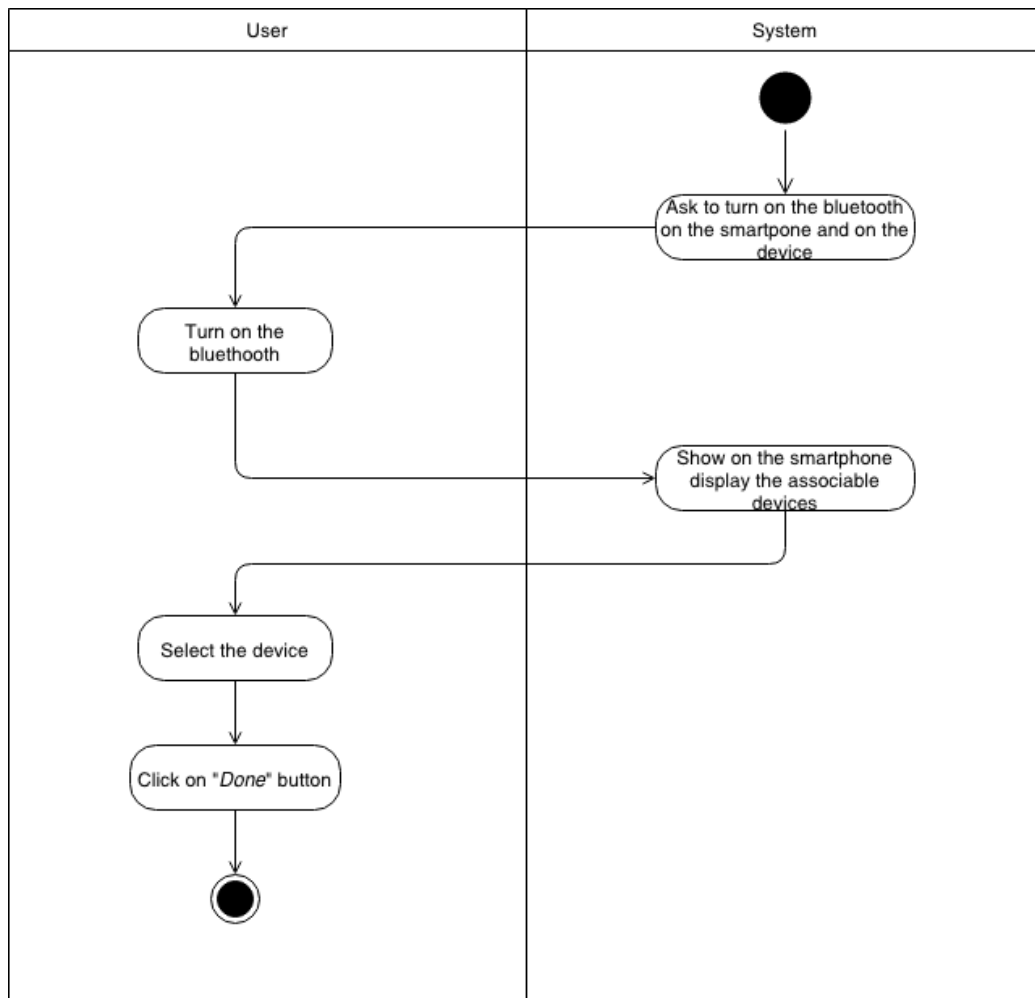
Figure 3.8: *First Log In* activity diagram

Figure 3.9: *First Log In* mockup

3.2.4 Third Party Log In

Purpose
The main goal of the login feature is to allow the access to one of the services of *Data4Help* to any registered special user. To access to tjhe services the special user has to fill out the credential form where e-mail and password are required. Moreover, there is a *Forgot password?* section where a special user could recover its password via e-mail. An e-mail is sent to the user with a new password.

Scenario 1
The Policlinico Cardiology Departement wants to acquire data of its patients, knowing their SSN. In order to see the health status of a specific patient, Francesca - head nurse - opens her laptop and goes to *Data4Help* web site. Francesca inserts the e-mail address of the departement and the password, the access is allowed, the system shows her the dashboard and she is able to check her patient status.

Use case and Functional Requirement
According to the **Individual Log In** [Section 3.2.1] functional requirements and use case are the same.
Exception is done only for the first access where device conncetion is not asked in this case.

Activity Diagram
The *Third Party Log In* activity diagram is the same of the one shown in Figure 3.6.

Mockup
The *Third Party Log In* mockup is the same of the one shown in Figure 3.7.

3.2.5 Manage Profile

Purpose

Any user can manage his personal profile both from *Data4Help* web site and from *AutomatedSOS* or *Track4Run* applications. In particular:

- The user can change some of his personal informations: his/her city of residence, his/her address and his/her occupation.

- The user can see the data acquired on him/her until this moment;

- The user can see the past received requests about seeing his/her personal data;

- The user can see the pending received requests about seeing his/her personal data and accept/refuse them;

- The user can change his/her password;

- The user can change the device associated to his/her profile (only from *AutomatedSOS* or *Track4Run* applications);

- The user can delete his/her profile.

Scenario 1

Chiara has just finished her studies and has just found a new job, so she wants to update the occupation field on her profile. She opens *Data4Help* web site from her personal computer, she logs in and goes in her "*Edit profile*" area. The system gives her the possibility to change either her city of residence or her occupation, she changes her occupation from student to emplyed and she clicks on the "*Submit changes*" button.

Scenario 2

Matteo has just finished the registration process, but he doesn't like the password he was given by the system and he wants to change it. He opens *AutomatedSOS* application on his smartphone, logs in and accesses to his "*Edit profile*" area. Now he clicks on the "*Change password*" button and inserts the old password and the new password twice as required by the system. Finally he clicks on the "*Submit changes*" button.

Scenario 3
Aldo moved to USA and so he decides to delete his profile on *Track4Run* because he was used to use it to organize amateur runs with his friends, but now he won't be able to do it anymore. He opens *Track4Run* application on his smartphone, logs in and accesses to his "*Edit profile*" area. Now he clicks on the "*Delete profile*" button and confirms his choice. The system removes all Aldo's information from the database.

Scenario 4
Franco has just received a new smartwatch for his birthday and so he wants to change the device associated to his *Data4Help* profile. He opens *Track4Run* application on his smartphone, logs in and accesses to his "*Edit profile*" area. Then he clicks on "*Change device*" button and turns on the bluetooth of the new smartwatch and of his smartphone. He selects the new smartwatch among those that appears on the smartphone's screen and clicks on "*Done*" button. Now he can use his new smartwatch;

Use Case
The *Profile Visualization* use case is analyzed in Table 3.5.
The *Modify Personal Information* use case is analyzed in Table 3.6.
The *Change Password* use case is analyzed in Table 3.7.
The *Change Device* use case is analyzed in Table 3.8.
The *Delete Profile* use case is analyzed in Table 3.9

Activity Diagram
The *Profile Visualization* activity diagram is shown in Figure 3.10.
The *Modify Personal Informations* activity diagram is shown in Figure 3.11.
The *Change Password* activity diagram is shown in Figure 3.12.
The *Change Device* activity diagram is similar to the one shown in Figure 3.8.
The *Delete Profile* activity diagram is shown in Figure 3.13.

Mockup
The *Manage Profile* mocukp is shown in Figure 3.14.

Functional requirements

1. The system must let the user view his/her personal profile at anytime;

2. The system must let the user upload/change his/her personal information at anytime;

3. The system must let the user change his password only if the old one has been inserted correctly;

4. The system must not let the user change his password if the new one has not been inserted correctly twice;

5. The system must let the user change the device connected to his/her profile at anytime;

6. The system must let the user delete his/her profile at anytime;

7. The system must require to confirm a deleting request;

8. The system must not delete a profile if the choice isn't confirmed by the user;

9. The system must let the user leave the editing profile process at anytime;

10. The system must delete all user's personal information from its database when the user decides to delete his/her profile;

11. The system must let the user accept/refuse an individual data request at anytime.

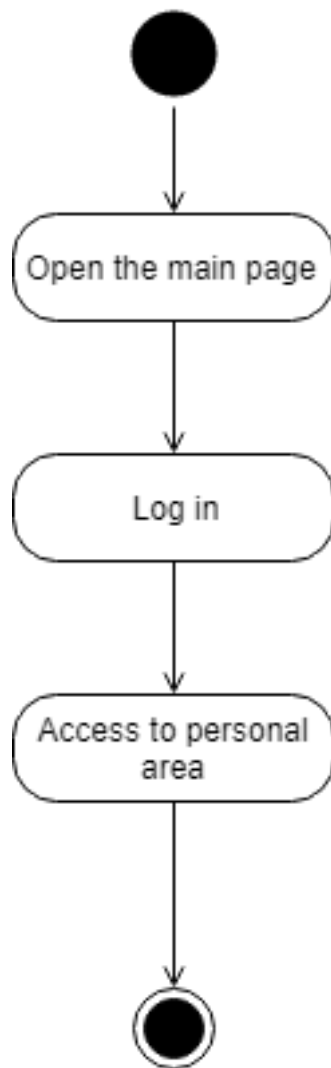| Actor	**User**
Goal	**[G.3]**
Input Condition	A **User** wants to view his personal profile
Event Flow	1. The **User** opens *Data4Help* web site or *AutomatedSOS* or *Track4Run* applications; 2. The **User** logs in; 3. The **User** accesses to his personal area; 4. The system shows to the **User** his/her "*Edit profile*" area and the buttons to move to "*Acquired Data*" area, "*Past Requests*" area and "*Pending Requests*" area.
Output Condition	The **User** views his/her personal profile
Exceptions	None

Table 3.5: *Profile Visualization* use case

Figure 3.10: *Profile Visualization* activity diagram from user's point of view

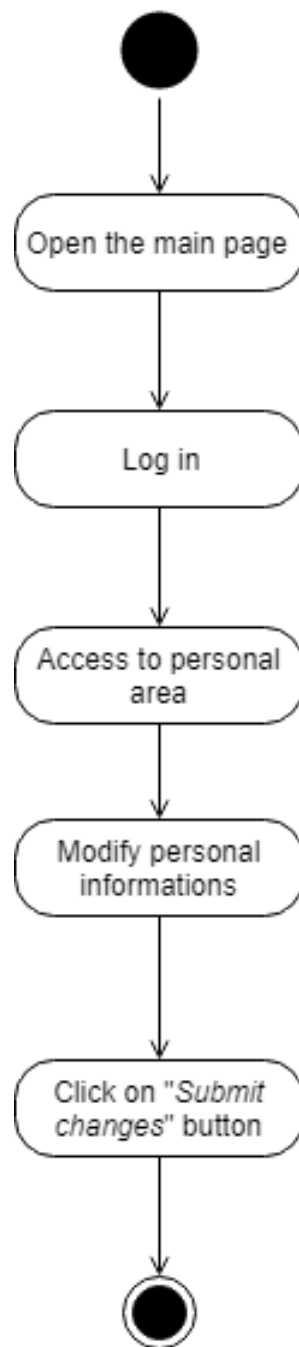| Actor	**User**
Goal	**[G.3]**
Input Condition	A **User** wants to modify his/her personal information
Event Flow	1. The **User** opens *Data4Help* web site or *AutomatedSOS* or *Track4Run* applications; 2. The **User** logs in; 3. The **User** accesses to his/her personal area; 4. The **User** goes in "*Edit profile*" area; 5. The system shows the **User** the modifiable information; 6. The **User** modifies what he/she wants; 7. The **User** clicks on the "*Submit changes*" button;
Output Condition	The **User**'s information are modified
Exceptions	• If the **User** decides to leave the editing process this one is aborted.

Table 3.6: *Modify Personal Information* use case

Figure 3.11: *Modify Personal Information* activity diagram from user's point of view

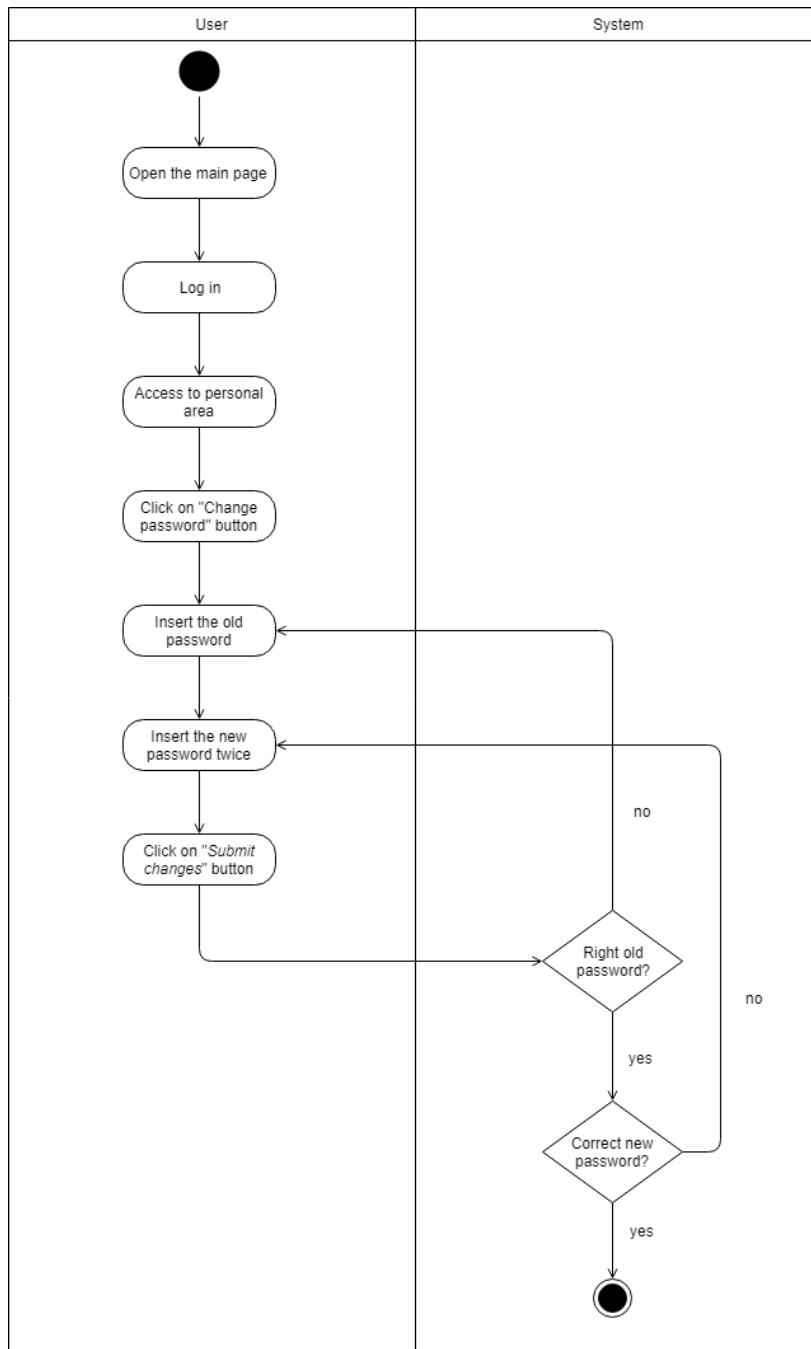| Actor	**User**
Goal	**[G.3]**
Input Condition	A **User** wants to change his password
Event Flow	1. The **User** opens *Data4Help* web site or *AutomatedSOS* or *Track4Run* applications; 2. The **User** logs in; 3. The **User** accesses to his personal area; 4. The **User** goes in "*Edit profile*" area; 5. The **User** clicks on the "*Change password*" button; 6. The system shows the **User** the fields in which he/she has to insert the old and the new password; 7. The **User** inserts the old password; 8. The **User** inserts the new password twice; 9. The **User** clicks on the "*Submit changes*" button;
Output Condition	The **User**'s password is modified
Exceptions	• If functional requirement 3 is not satisfied the system goes back to step 7; • If functional requirement 4 is not satisfied the system goes back to step 8; • If the **User** decides to leave the editing process this one is aborted.

Table 3.7: *Change Password* use case

Figure 3.12: *Change Password* activity diagram

Actor	**User**
Goal	**[G.3]**
Input Condition	A **User** wants to change the associated device
Event Flow	1. The **User** opens *AutomatedSOS* or *Track4Run* applications; 2. The **User** logs in; 3. The **User** accesses to his personal area; 4. The **User** goes in "*Edit profile*" area; 5. The **User** clicks on the "*Change device*" button; 6. The system asks the **User** to turn on the bluetooth of the smartwatch and of the smartphone; 7. The **User** turns on the bluethooth; 8. The system shows on the smartphone display the associable devices that it finds with the bluetooth connection; 9. The **User** selects the device he wants to associate. 10. The **User** clicks on "*Done*" button.
Output Condition	The new device is correctly connected to the **User**'s smartphone
Exceptions	• If the **User** decides to leave the changing device process this one is aborted.

Table 3.8: *Change Device* use case

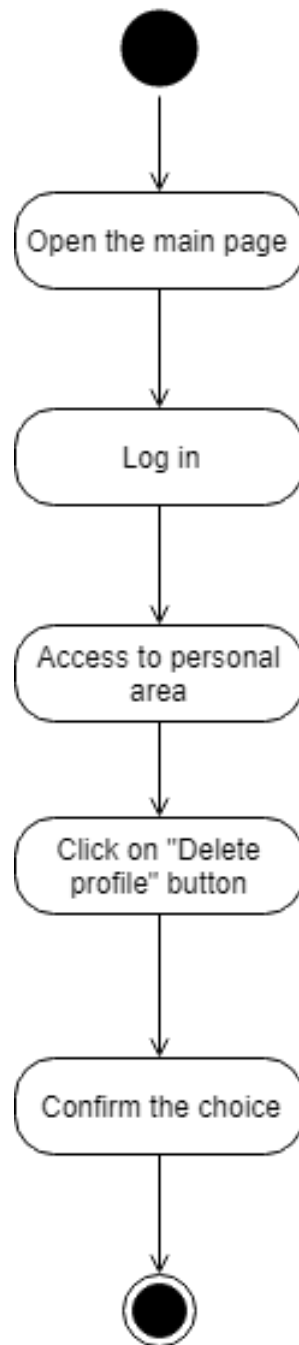| Actor	**User**
Goal	**[G.3]**
Input Condition	A **User** wants to delete his/her profile
Event Flow	1. The **User** opens *Data4Help* web site or *AutomatedSOS* or *Track4Run* applications; 2. The **User** logs in; 3. The **User** accesses to his personal area; 4. The **User** goes in "*Edit profile*" area; 5. The **User** clicks on the "*Delete profile*" button; 6. The **User** confirms his/her choice;
Output Condition	The **User**'s profile is deleted
Exceptions	- If functional requirement 8 is not satisfied the deleting process is aborted; - If the **User** decides to leave the deleting process this one is aborted.

Table 3.9: *Delete Profile* use case

Figure 3.13: *Delete Profile* activity diagram from user's point of view

Figure 3.14: *Manage Profile* mockup

3.2.6 Manage Third Party Profile

Purpose

Any special user can manage its profile from *Data4Help* web site, in particular:

- The special user can change some of its informations: its legal address, its billing address, its corporate e-mail address and the sector in which it operates;

- The special user can change all the data regarding its legal representative;

- The special user can see the data it has required and paied until this moment;

- The special user can see the requests it hasn't paied yet;

- The special user can change its password;

- The special user can delete its profile.

Scenario 1

The executive director of PincoPallo SPA had a serious fight with the legal representative of his company, and he decided to fire him a week ago. Now that he has find a new legal he wants to change the data stored in his *Data4Help* profile. He accesses to *Data4Help* web site from his personal pc, he logs in with the company profile and he goes in "*Edit profile*" area. Now he inserts all the information about the new legal in the matching fields and then clicks on the "*Submit changes*" button.

Scenario 2

AlphaAnalisi SPA wants to change the preferred payment method due to changes in its internal organization. It opens *Data4Help* main page, logs in and goes in its "*Edit profile*" area. It clicks on the "*Change payment method*" button and selects the new preferred method. Finally it clicks on the "*Submit changes*" button.

Use Case

The *Special Profile Visualization* use case is analyzed in Table 3.10.
The *Modify Personal Information* use case is analyzed in Table 3.6.
The *Change Password* use case is analyzed in Table 3.7.
The *Delete Profile* use case is analyzed in Table 3.9.

Activity Diagram

The *Special Profile Visualization* activity diagram is similar to the one shown in Figure 3.10.

The *Modify Personal Information* activity diagram is similar to the one shown in Figure 3.11.

The *Change Password* activity diagram is similar to the one shown in Figure 3.12.

The *Delete Profile* activity diagram is similar to the one shown in Figure 3.13.

Mockup

The *Special Profile Visualization* mockup is shown in Figure 3.15.

Functional requirements

1. The system must let the special user view its personal profile at anytime;

2. The system must let the special user upload/change its personal information at anytime;

3. The system must let the special user change its password only if the old one has been inserted correctly;

4. The system must not let the special user change its password if the new one has not been inserted correctly twice;

5. The system must let the special user delete its profile at anytime;

6. The system must require to confirm a deleting request;

7. The system must not delete a profile if the choice isn't confirmed by the special user;

8. The system must let the special user leave the editing profile process at anytime;

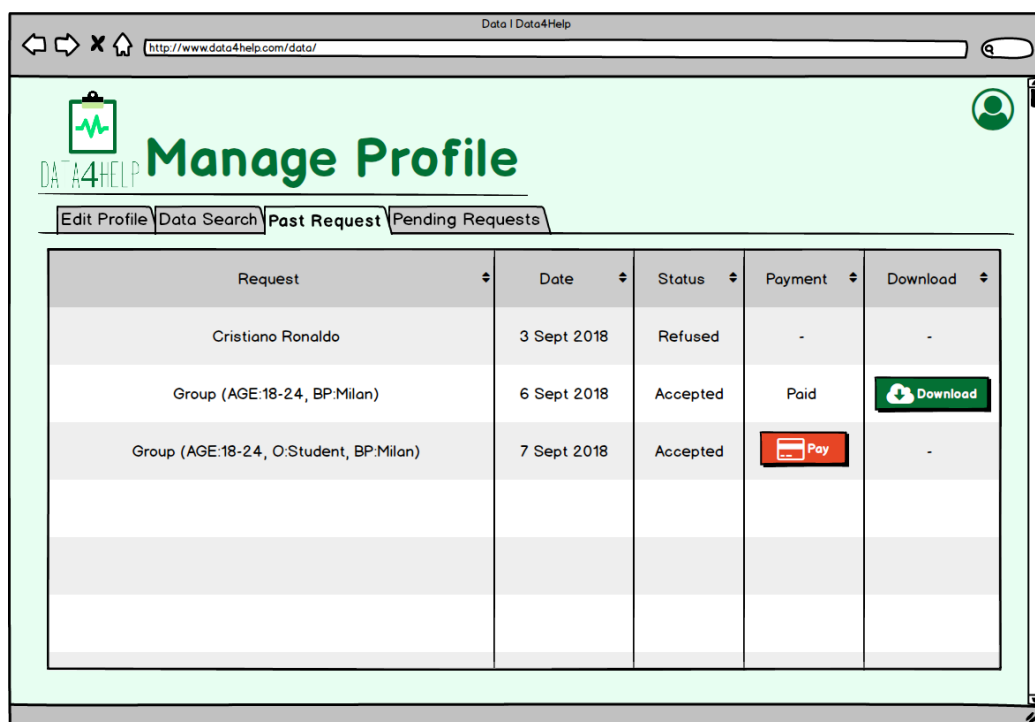| Actor	**Special user**
Goal	**[G.3]**
Input Condition	A **Special user** wants to view its profile
Event Flow	1. The **Special user** opens *Data4Help* web site; 2. The **Special user** logs in; 3. The **Special user** accesses to its personal area; 4. The system shows to the **Special user** its "*Edit profile*" area and the buttons to move to "*Past Request*" area and "*Pending Requests*" area.
Output Condition	The **Special user** views its profile withss all the related information
Exceptions	None

Table 3.10: *Special Profile Visualization* use case

Figure 3.15: *Special Profile Visualization* mockup

3.2.7 Individual Data Requirement

Purpose

Everytime a special user wants to access to the data of a specific individual it has to pass through some steps:

1. It has to insert the SSN of the targeted individual, who will receive a direct request that he/she can accept or refuse;

2. If the request has been accepted the special user will receive an e-mail with the amount it has to pay to download the required data;

3. If it pays the amount due it can download the required data.

Scenario 1

AC Milan wants to acquire information about the lifestyle of Cristino Ronaldo, so it opens the browser and searches for *Data4Help* web site, it logs in and then it goes in "*Data Search*" area. Then it inserts Ronaldo's SSN, clicks on "*Submit*" button and waits a couple of days for a response. Unfortunately, Ronaldo doesn't accept the request, AC Milan receives an e-mail containing the negative response and the process ends.

Scenario 2

The "San Gerardo" hospital of Monza is trying *Data4Help* combined with *AutomatedSOS* to monitor the health status of some of its patients after the dismissal. Lucia has been dismissed a couple of month ago and she has *AutomatedSOS* installed on her smartphone. The "San Gerardo" hospital accesses to *Data4Help* from the web site, logs in, goes in "*Data Search*" area, inserts Lucia's SSN, clicks on the "*Submit*" button and waits a couple of days for a response. Lucia accepts the request, so the "San Gerardo" hospital receives an e-mail with the amount it has to pay, pays it and downloads Lucia's data.

Use Case

The *Individual Data Requirement* use case is analyzed in Table 3.11.

Activity Diagram

The *Individual Data Requirement* activity diagram is shown in Figure 3.16.

Mockup

The *Individual Data Requirement* mocukp is shown in Figure 3.17.

Functional requirements

1. The system must refuse a non existant SSN;

2. The system must not let the special user download the required data untill it hasn't paid the amount due;

3. The system must send an e-mail in case of negative response and end the process;

4. The system must send an e-mail with the amount due in case of positive response;

5. The system must let the **Special user** leave the data requirement process at anytime.

Actors	**Special user** and targeted individual
Goal	**[G.4]**
Input Condition	A **Special user** wants to acquire data of an individual
Event Flow	1. The **Special user** opens the main page of *Data4Help* web site; 2. The **Special user** logs in; 3. The **Special user** goes in "*Data Search*" area; 4. The **Special user** inserts the SSN of the targeted individual; 5. The **Special user** clicks on the "*Submit*" button; 6. The **Special user** recieves an e-mail with the response; 7. If the response is positive the **Special user** recieves informations about the amount it has to pay; 8. The **Special user** pays the amount due (going in the *"Pending Request"* area in its personal area); 9. The **Special user** downloads the required data.
Output Condition	The **Special user** receives the required data
Exceptions	• If functional requirements 1 is not satisfied the process goes back to step 4; • If functional requirement 2 is not satisfied the process goes back to step 8; • If the targeted individual refuse to share his/her data the process is aborted; • If the **Special user** decides to leave the data requirement process this one is aborted.

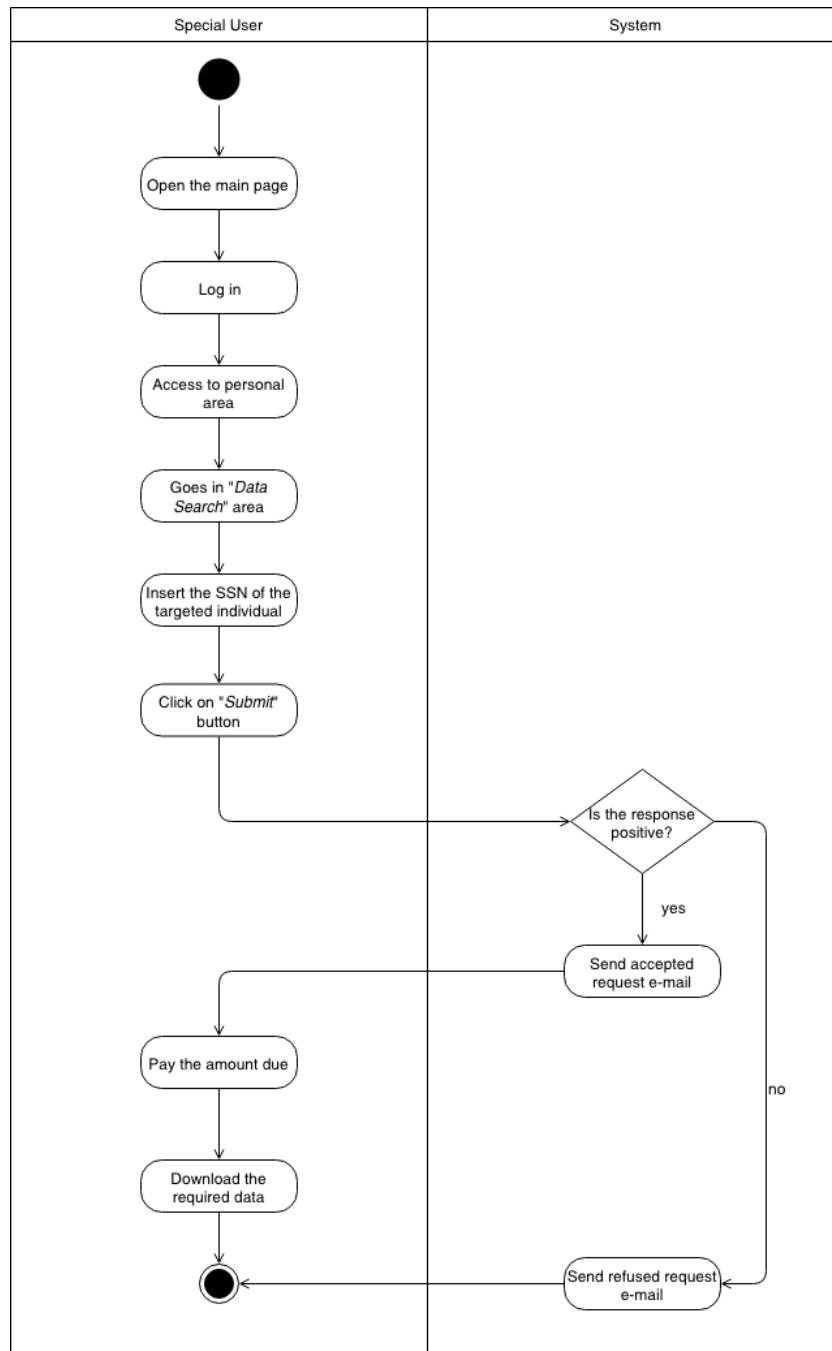Table 3.11: *Individual Data Requirement* use case

Figure 3.16: *Individual Data Requirement* activity diagram

Figure 3.17: *Individual Data Requirement* mockup

3.2.8 Group Data Requirement

Purpose
Everytime a special user wants to access to the data of a group of people it has to specify at least one of the required characteristics of it:

- It can choose among the age ranges of the members of the targeted group proposed by the system;

- It can specify the Italian city of birth of the members of the targeted group;

- It can specify the Italian city of residence of the members of the targeted group;

- It can specify the Italian province of residence of the members of the targeted group;

- It can specify the Italian province of birth of the members of the targeted group;

- It can specify the Italian region of residence of the members of the targeted group;

- It can specify the Italian region of birth of the members of the targeted group;

- It can specify the state of birth of the members of the targeted group;

- It can specify the state of residence of the members of the targeted group;

- It can specify the current occupation (students, employeds, unemployeds) of the members of the targeted group.

After this the system will check if the targetd group is composed of more than 1000 people, if it is, the special user will receieve the payment form with the amount it has to pay to download the required data. Once it pays the amount due it can download the required data.

Scenario 1
PharmaAnalisi SPA wants to acquire data of a group of students living in Lombardia in order to do an analysis about the kind of life they conduct. It opens the browser and search for *Data4Help* web site, then it logs in and goes in "*Data Search*" area. It specifies that the age range of the members

of the targeted group must be from 18 to 24 years old, that they should live in a city in Lombardia and that they should be students. Then it clicks on the "*Submit*" button and the system accepts its request, PharmaAnalisi SPA receives the payment form with the amount it has to pay, it pays the amount due and it downloads the required data.

Scenario 2
The municipality of Sondrio wants to analyze the quality of life of the people that were born in Monza and that moved to Sondrio. It opens the browser and search for *Data4Help* web site, then it logs in and goes in "*Data Search*" area. It specifies that the city of birth of the members of the targeted group must be Monza and that their city of residence must be Sondrio. Then it clicks on the "*Submit*" button but unfortunately the system tells that the required data aren't accessible because the targeted group of people is composed of less than 1000 people and so the process ends.

Use Case
The *Group Data Requirement* use case is analyzed in Table 3.12.

Activity Diagram
The *Group Data Requirement* activity diagram is shown in Figure 3.18.

Mockup
The *Group Data Requirement* mockup is shown in Figure 3.17.

Functional requirements

1. The system must not give data of groups of people composed of less than 1000 people;

2. The system must not let the special user download the required data untill it hasn't paid the amount due;

3. The system must let the **Special user** leave the data requirement process at anytime.

| Actor	**Special user**
Goal	**[G.5]**
Input Condition	A **Special user** wants to acquire data of a group of people
Event Flow	1. The **Special user** opens the main page of *Data4Help* web site; 2. The **Special user** logs in; 3. The **Special user** goes in "*Data Search*" area; 4. The **Special user** specifies the characteristics of the targeted group; 5. The **Special user** clicks on "*Submit* button; 6. If the targeted group is composed of more than 1000 people the **Special user** receieves the payment form with the amount it has to pay to download the required data; 7. The **Special user** pays the amount due; 8. The **Special user** downloads the required data.
Output Condition	The **Special user** receives the required data
Exceptions	• If the targeted group is composed of less than 1000 people the **Special user** is informed and the process is aborted; • If functional requirement 2 is not satisfied the process goes back to step 7; • If the **Special user** decides to leave the data requirement process this one is aborted.

Table 3.12: *Group Data Requirement* use case
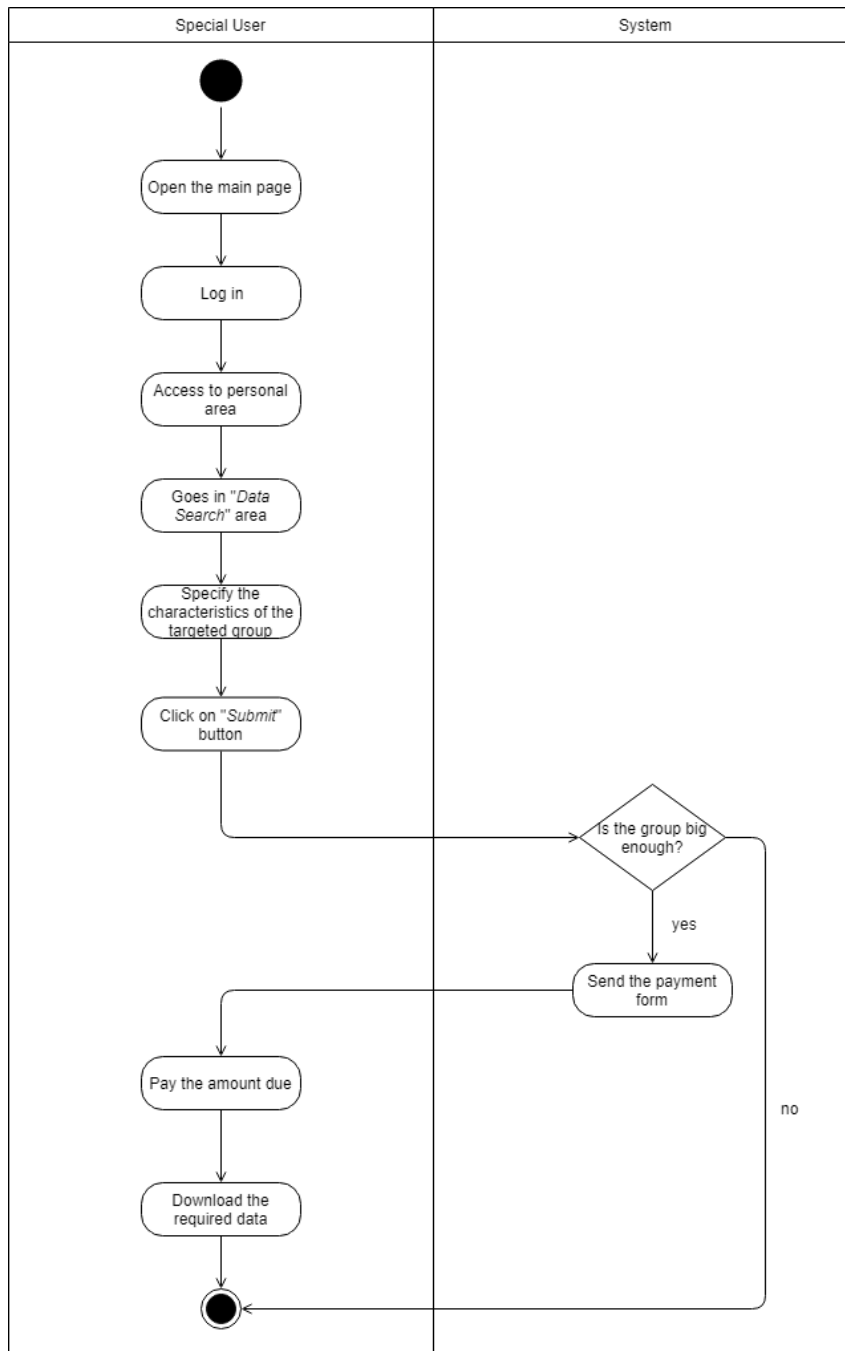
Figure 3.18: *Group Data Requirement* activity diagram

3.2.9 Health Status Visualization

Purpose

The main feature of *AutomatedSOS* is to check the health status of the user and to detect any critical situation. The detection of critical situation computes a huge number of data, which provides several information about the health status of the user, so *AutomatedSOS* provides also a service to show the health status of the user.

Scenario

Silvia is worried about her grandmother's health status because she is been tired for a few days. Two month ago Silvia installed on her grandmother phone *AutomatedSOS* application. In order to calm herself, Silvia takes the phone of her grandmother, opens *AutomatedSOS* and logs in. In the main page of the application there is the summary of the health status and everything looks ok. To avoid any doubt Silvia clicks on the "*Details*" button and checks all the statistics about last week and last month value of pressure and heartbeat.

Use Case

The *Health Status Visualization* use case is analyzed in Table 3.13.

Activity Diagram

The *Healt Status Visualization* activity diagram is shown in Figure 3.19.

Mockup

The *Healt Status Visualization* mockup is shown in Figure 3.20.

Functional requirements

1. The system must let the user view his personal health status at anytime;

2. The system must update the health status of the user at anytime it receives new data from the devices;

3. The system must stores the data in order to provide monthly and weekly statistics.

| Actor	**User**
Goal	**[G.7]**
Input Condition	A **User** wants to view his/her health status.
Event Flow	1. The **User** opens *AutomatedSOS* application; 2. The **User** logs in; 3. The **User** clicks on the "*Health Status*" button; 4. The system shows to the **User** the data acquired on him/her with the relative computation of the health status.
Output Condition	The **User** views his/her personal health status monitored by *AutomatedSOS*
Exceptions	If the system does not acquire enough data to produce statistics it notifies the **User** with a warning.

Table 3.13: *Health Status Visualization* use case
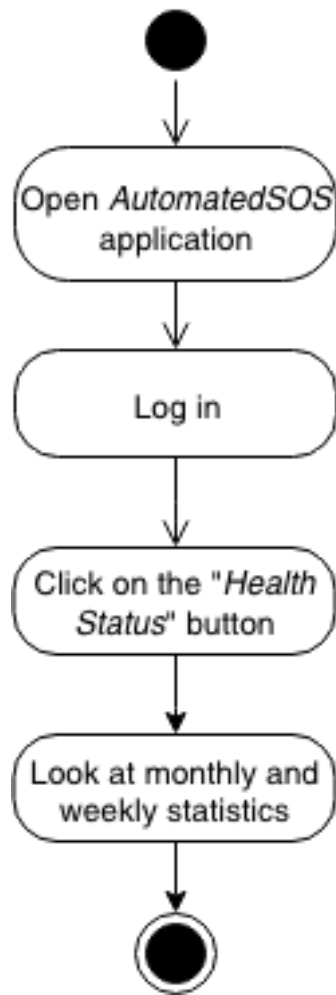
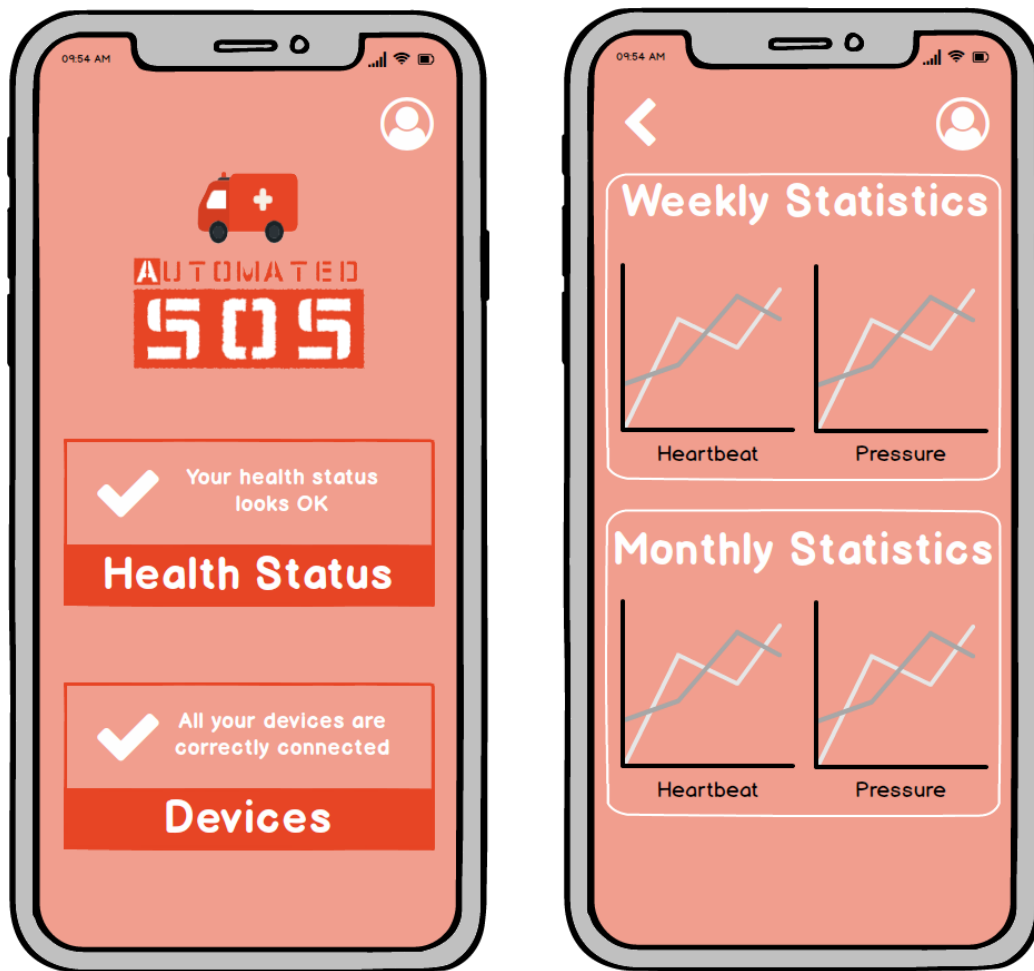Figure 3.19: *Health Status Visualization* activity diagram from user's point of view

Figure 3.20: *Health Status Visualization* mockup

3.2.10 Critical Situation

Purpose

The main feature of *AutomatedSOS* is to check the health status of the user and to detect any critical situation. *AutomatedSOS* monitors the health status of the subscribed customers and, when such parameters are below certain thresholds, sends to the location of the customer an ambulance, guaranteeing a reaction time of less than 5 seconds from the time the parameters are below the threshold.

Scenario 1

For a couple of days Vittorio felt very tired and affected of sickness. While he was walking in his house his heartbeat went down and he lied down on the floor. Luckily he had installed *AutomatedSOS* application. The application detected a critical situation, it managed to track Vittorio position and it called the ambulace. The ambulace arrived very quickly and luckily the paramedics with a cardiac massage saved Vittorio that was carried to the hospital.

Scenario 2

Cristiano is a professional runner. He decides to install on his phone *AutomatedSOS* to keep trace of his health status and avoid any possible critical situation when he runs. One day, while he was running, *AutomatedSOS* went in an alerted status but after only 1 second the application came back in a normal status and stayed in the normal status for all the duration of the run. Probably it was an abnormal device measure of life value, so *AutomatedSOS* did not call the ambulance.

Use Case

The *Critical Situation* use case is analyzed in Table 3.14.

Activity Diagram

The *Critical Situation* activity diagram is shown in Figure 3.21.

Mockup

The *Critical Situation* mockup is shown in Figure 3.22.

Functional requirements

1. The system must guarantee a reaction time of less than 5 seconds from the time it detects an alerted situation;

2. The system must send the location of the customer to an ambulance;

3. The system must be in an alerted status when maximum pressure value of the customer is more than 170 mmHg and minimum pressure value is more than 100 mmHg;

4. The system must be in an alerted status when the heartbeat is lower than 45 bmp or it is higher than 120 bpm;

5. If the system goes in an alerted status it has to increase the life parameters detection frequency.

Actor	**User**
Goal	**[G.8]**
Input Condition	The system goes in an alerted status
Event Flow	1. The system gets the GPS position of the **User**; 2. The system increases parameters detection with a frequence of 3 detection per second; 3. The system shows an alert message on the **User**'s smartphone.
Output Condition	The system calls an ambulance and it sends the location to the called ambulance if it is in an alert status from 3 seconds.
Exceptions	• If functional requirement 2 is not satisfied the system notifies the ambulance about the detected position error and sends the last detected position; • If the system could not detect life parameters of the **User**, it invites him/her to check the status and the conncetion of his/her device with a warning message.

Table 3.14: *Critical Situation* use case
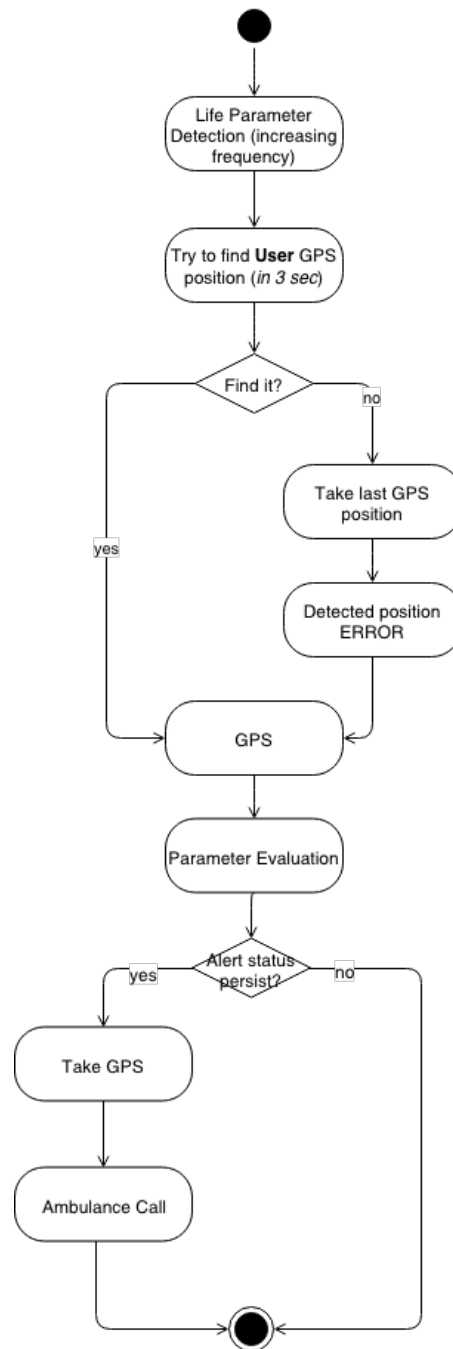
Figure 3.21: *Critical Situation* activity diagram from system's point of view

Figure 3.22: *Critical Situation* mockup

3.2.11 Create a Run

Purpose

A very important feature of *Track4Run* is the possibility for an *Organizer* to create a new *Run* defining:

- The name of the *Run*;

- The path of the *Run* through an intercative tool;

- The date of the *Run*;

- The maximum number of participants to the *Run*;

- The expiration date to enrol in the *Run*;

Scenario 1

Massimo wants to organize a charity *Run* in his little town. He is already registered to *Track4Run* as a Runner. After he has recived all bureaucratic permissions he went to *Track4Run* web site. With the same credential of the mobile application he logged in the system and in the dashboard he clicked on the "*Create a Run*" button. Massimo set the path through the intercative tool, fixed the date of the *Run* and the missing fields. When everything was completed he clicked on the "*Create*" button and the *Run* went on-line.

Use Case

The *Create a Run* use case is analyzed in Table 3.15.

Activity Diagram

The *Create a Run* activity diagram is shown in Figure 3.23.

Mockup

The *Create a Run* mockup is shown in Figure 3.24.

Functional requirements

1. The system must not accept a *Run* with date less than or equal to the current one;

2. The system must not accept a *Run* with expiration date less than or equal to the current one;

3. The system must not accept a *Run* with maximum number of participants less than or equal to 1.

4. The system must not accept a *Run* with a path duplication greater or equal to the 50 percent in the same date of an existent one;

5. The system must let the **Organizer** leave the creation process at anytime.

Actor	**Organizer**
Goal	[**G.10**]
Input Condition	The **Organizer** wants to create a new *Run*
Event Flow	1. The **Organizer** opens *Track4Run* service through web application and he/she log in; 2. The **Organizer** clicks on the "*Create a Run*" button; 3. The **Organizer** inserts path, date, expiration date and maximum number of participants of the *Run*; 4. The **Organizer** clicks on the "*Create*" button;
Output Condition	The system registers the new *Run* and it notifies the **Organizer** with a confirmation e-mail.
Exceptions	• If functional requirements 1, 2, or 3 are not satisfied the system notifies the **Organizer** with an error message and the process goes back to step 3; • In order to prevent functional requirements 4 failure, during the building phase of the path the system continuously checks satisfaction and when the functional requirement is not satisfied it notifies the **Organizer** with a warning; • If the **Organizer** decides to leave the creation process this one is aborted.

Table 3.15: *Create a Run* use case

Figure 3.23: *Create a Run* activity diagram from user's point of view
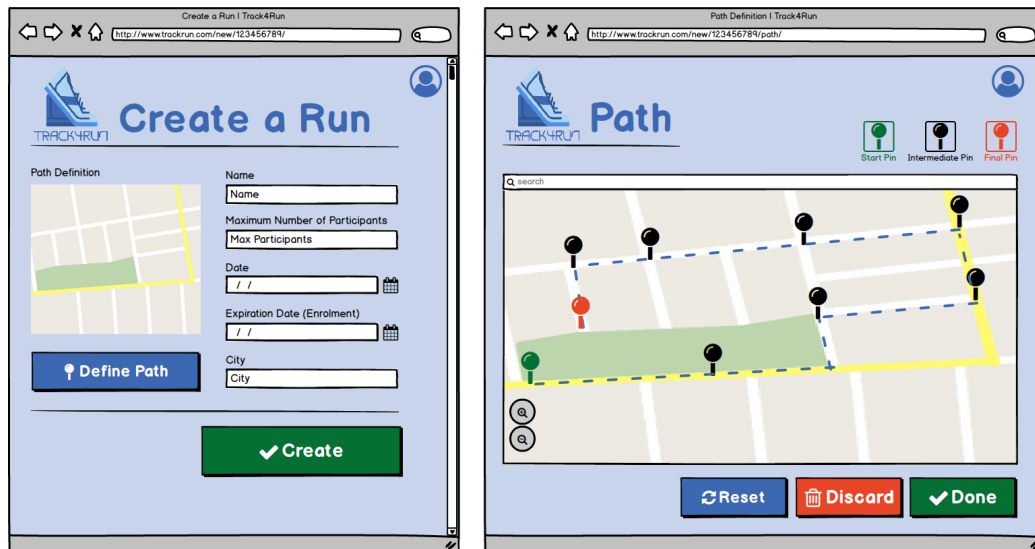
Figure 3.24: *Create a Run* mockup

3.2.12   Delete a Run

Purpose
As we have the possibilty in *Track4Run* for an organizer to create a new *Run*, the system must also be able to manage the decision of an organizer to delete a *Run*.

Scenario
Alice wants to delete a *Run* that she had organized for the next month, because the weather forecast are not optimal. In order to avoid a rainy day Alice opened *Track4Run* web-site, she logged in and she clicked on the "*Manage a Run*" button. On top of the table there was the *Run* that she was looking for and she clicked on its "*Delete*" button. After that a mail arrived in Alice mailbox, it was the confirmation of the elimination of the *Run*.

Use Case
The *Delete a Run* use case is analyzed in Table 3.16.

Activity Diagram
The *Delete a Run* activity diagram is shown in Figure 3.25.

Mockup
The *Delete a Run* mockup is shown in Figure 3.26.

Functional requirements

1. The system must not show *Run*, in *Manage a Run* section, of which an **Organizer** is not the owner;

2. The system must let the **Organizer** leave the deletion process at anytime.

Actor	**Organizer**
Goal	**[G.11]**
Input Condition	An **Organizer** wants to delete a *Run*
Event Flow	1. The **Organizer** opens *Track4Run* service through web application and he/she logs in; 2. The **Organizer** clicks on the "*Manage a Run*" button; 3. The **Organizer** looks for a *Run* through the search bar or looking to the proposed ones; 4. The **Organizer** clicks on the "*Delete*" button of the targeted *Run*.
Output Condition	The system deletes the *Run* and notifies the **Organizer** with a confirmation e-mail. Moreover the system must notify all enrolled people in the *Run* with a e-mail and delete their enrolments.
Exceptions	• If the **Organizer** looks for a *Run* that is not present in the system or he/she is not the owner, the system notifies the **Organizer** with a warning message; • If the **Organizer** decides to leave the elimination process this one is aborted.

Table 3.16: *Delete a Run* use case
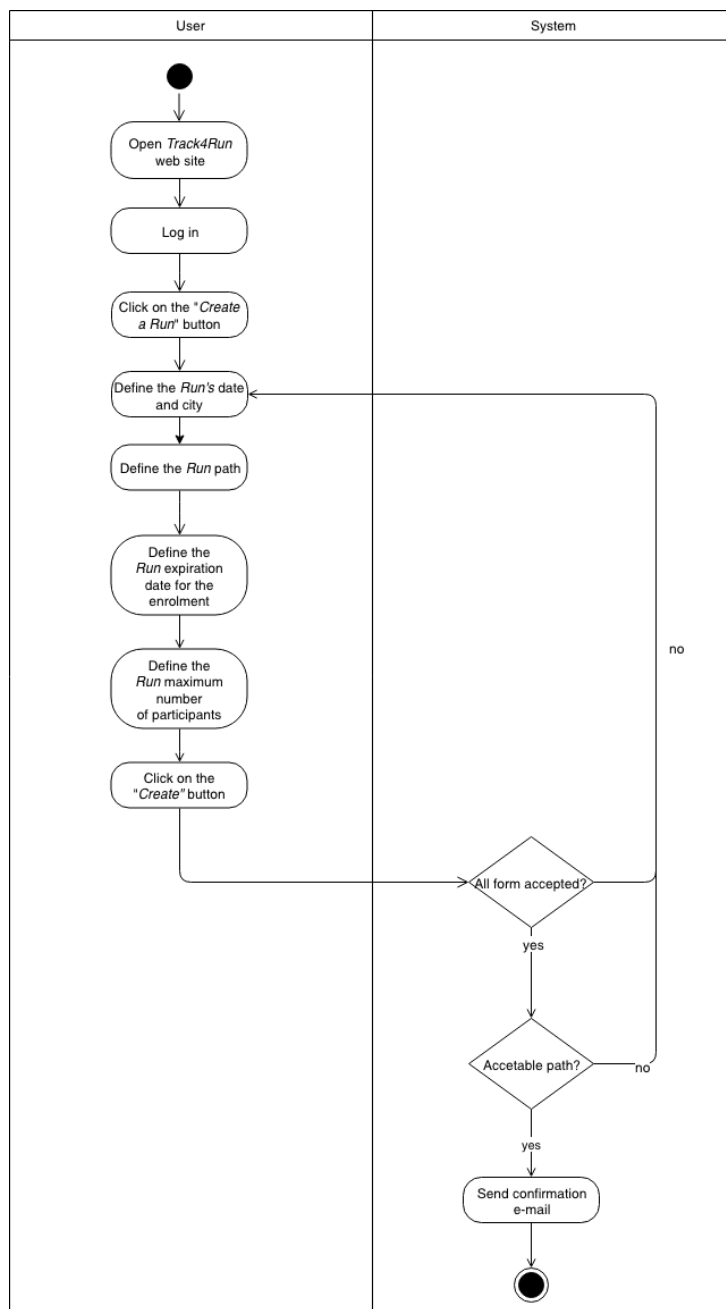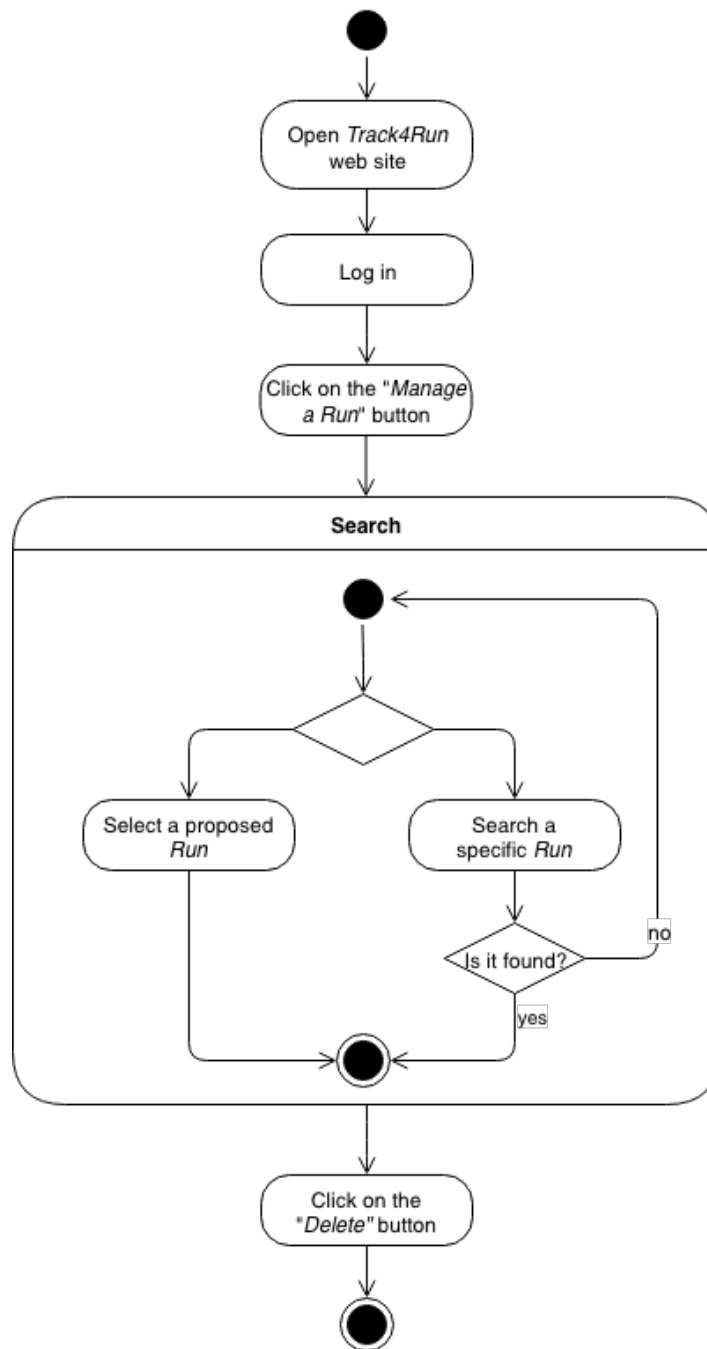
Figure 3.25: *Delete a Run* activity diagram from user's point of view
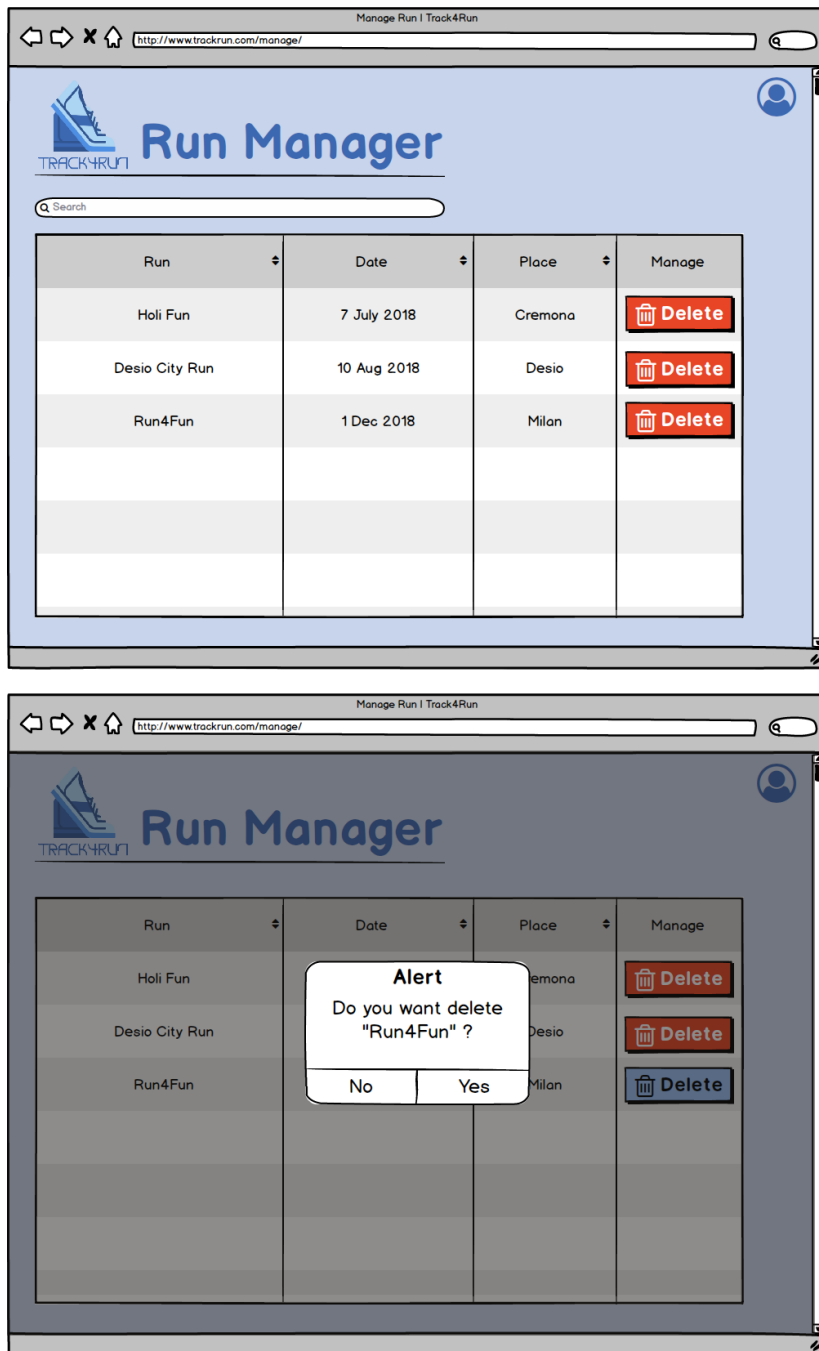
Figure 3.26: *Delete a Run* mockup

3.2.13   Enrol in a Run

Purpose
One of the great feature of *Track4Run* is the possibility for a runner to be able to enrol in a *Run*. In order to enrol in a *Run* a runner must be logged in *Track4Run* application, he/she has to search the *Run* that he/she wants in the *Enrol in a Run* section and finally enrols in it. However, the *Run* could be already done or the enrolling time expired.

Scenario 1
Andrea is a technological boy. A few weeks ago he found in the Play Store the new app *Track4Run* and shared his discovery with his friends. Yesterday, while he was hanging out with the buddies they discovered a new run for the week-end after. So, Andrea took his phone, opened *Track4Run* app, logged in, clicked on the "*Enrol in a Run*" button and found it on the top of the page. So Andrea clicked on it and when the *Run* event was opened he cliked on the "*Enrol*" button. After that Andrea received a confirmation e-mail about the correctness of the enrolment.

Scenario 2
Samanta loves walking but for a few months now she started running. She spoke about the city *Run* planned in two days with her friend Federica, but unfortunately when she took her phone and opened *Track4Run* she discovered that the time to enrol in the *Run* was expired yet.

Use Case
The *Enrol in a Run* use case is analyzed in Table 3.17.

Activity Diagram
The *Enrol in a Run* activity diagram is shown in Figure 3.27.

Mockup
The *Enrol in a Run* mockup is shown in Figure 3.28.

Functional requirements

1. The system must not accept an enrolment in a *Run* where a **Runner** is enrolled yet.

2. The system must not accept an enrolment in a *Run* where the enrolment time is expired yet.

3. The system must not accept an enrolment in a *Run* where the maximum number of enrolments is reached;

4. The system must not accept an enrolment in a *Run* where a **Runner** and the *Organizer* are the same person.

5. The system must let the **Runner** leave the enrolment process at anytime.

Actor	**Runner**
Goal	**[G.12]**
Input Condition	The **Runner** wants to enrol in a *Run*
Event Flow	1. The **Runner** opens *Track4Run* service through mobile application and he/she logs in; 2. The **Runner** clicks on "*Enrol in a Run*" button; 3. The **Runner** looks for a *Run* through the search bar or looking to the proposed ones; 4. The **Runner** clicks on the *Run* he/she wants to enrol in; 5. The **Runner** clicks on the "*Enrol*" button.
Output Condition	The system registers the enrolment of the **Runner** and it notifies him/her with a confirmation e-mail.
Exceptions	• If functional requirements 1, 2, 3 or 4 are not satisfied the system notifies the **Runner** with an error message and the process goes back to step 3; • If the **Runner** looks for a *Run* that is not present in the system, the system notifies him/her with a warning message; • If the **Runner** decides to leave the enrolment process this one is aborted.

Table 3.17: *Enrol in a Run* use case
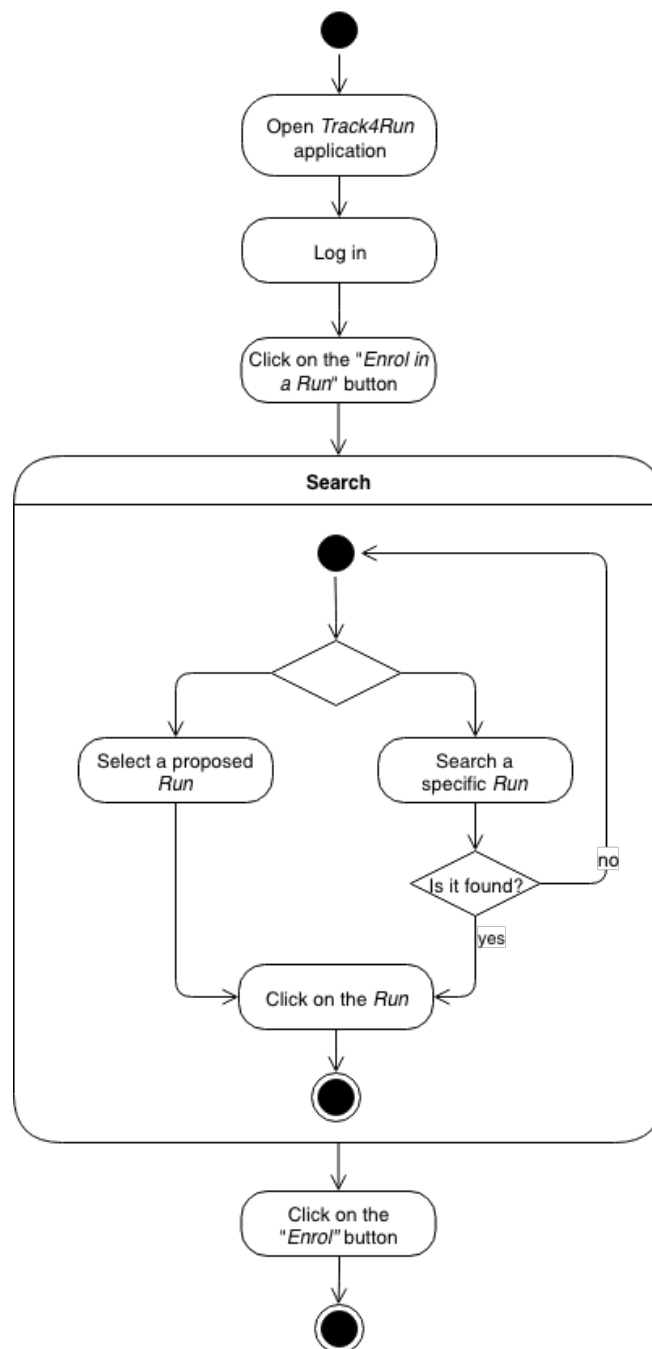
Figure 3.27: *Enrol in a Run* activity diagram from user's point of view

Figure 3.28: *Enrol in a Run* mockup

3.2.14 Delete an Enrolment in a Run

Purpose

As we have the possibilty in *Track4Run* for a runner to be able to enrol in a *Run*, the system must also be able to manage the decision of a runner to delete his/her enrolment.

Scenario

Giulia is enrolled in the annual *Run* of her neighborhood. The enrolment management was made through *Track4Run* application. Unfortunately for the day of the *Run* Giulia will be in Florence for an important work meeting. When Giulia received the meeting mail she took her phone, she opened *Track4Run* app and she clicked on "*Enrolled Run*" button. The only row in the table was the annual *Run*, Giulia clicked on it and then clicked on "*Delete Enrolment*" button. After that Giulia received a confirmation e-mail.

Use Case

The *Delete an Enrolment in a Run* use case is analyzed in Table 3.18.

Activity Diagram

The *Delete an Enrolment in a Run* activity diagram is shown in Figure 3.29.

Mockup

The *Delete an Enrolment in a Run* mockup is shown in Figure 3.30.

Functional requirements

1. The system must not show *Run*, in *Enrolled Run* section, in which a **Runner** is not enrolled;

2. The system must let the **Runner** delete his/her enrolment in a *Run* at anytime;

3. The system must let the **Runner** leave the elimination process at anytime.

Actor	**Runner**
Goal	**[G.13]**
Input Condition	A **Runner** wants to delete an enrolment in a *Run*
Event Flow	1. The **Runner** opens *Track4Run* service through mobile application and he/she logs in; 2. The **Runner** clicks on the "*Enrolled Run*" button; 3. The **Runner** looks for a *Run* through the search bar or looking to the proposed ones; 4. The **Runner** clicks on the *Run* in which he/she wants to delete the enrolment; 5. The **Runner** clicks on the "*Delete Enrolment*" button.
Output Condition	The system deletes the enrolment of the **Runner** and it notifies him/her with a confirmation e-mail.
Exceptions	• If the **Runner** looks for a *Run* that is not present in the system, the system notifies the **Runner** with a warning message; • If the **Runner** decides to leave the elimination process this one is aborted.

Table 3.18: *Delete an Enrolment in a Run* use case
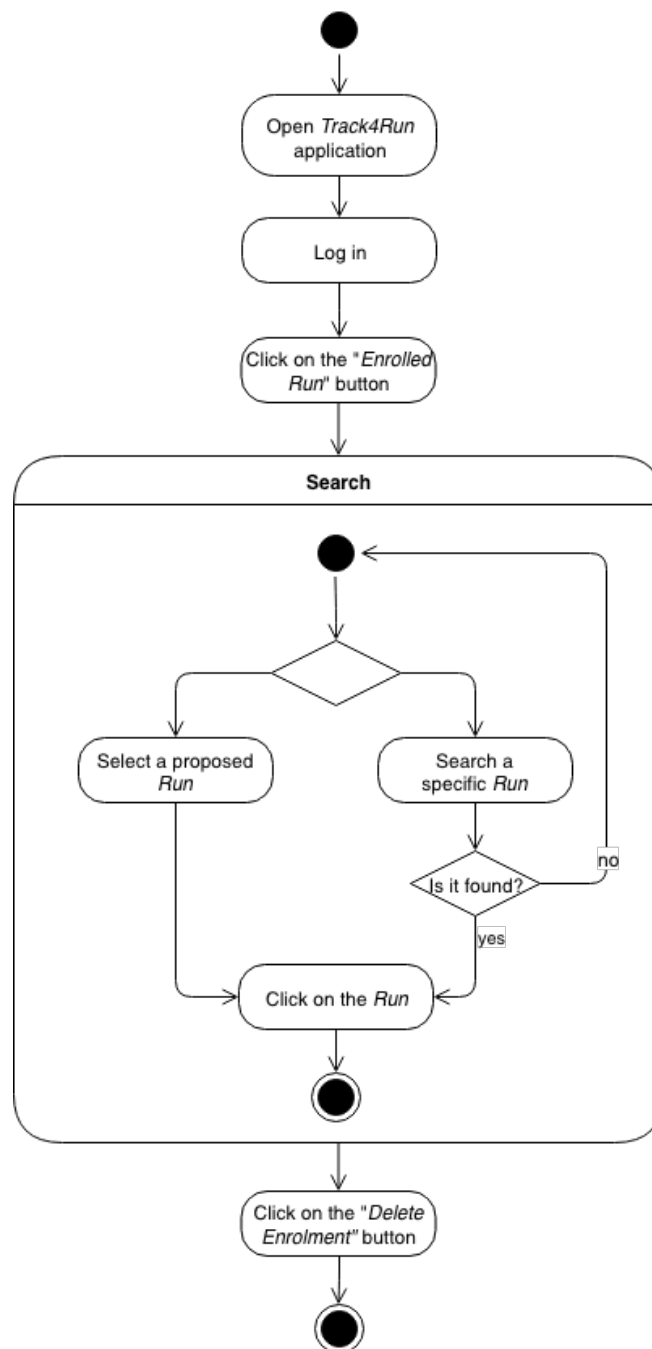
Figure 3.29: *Delete an Enrolment in a Run* activity diagram from user's point of view

Figure 3.30: *Delete an Enrolment in a Run* mockup

3.2.15   Run Watching

Purpose

For each *Run* that is present in the system, a *Specator* or an *User* must be able to watch it. The *Spectator* or the *User* can follow the positon on the *Run* path of any person that is enrolled in it and can see the placing table of the runners.

Scenario 1

Marco is a professional runner and plays for a team, unfortunately one month ago he broke his leg. Today there is an important *Run* and his teammates are enrolled in. At 4 p.m., Marco opens *Track4Run* app on his mobile phone, looks for the today's *Run* in the dashboard, clicks on it and immediately he finds himself "in the *Run*"; he follows his friend on the path (watching the position on the map) and clicking on the "*Placings*" button he also sees the current placing table.

Scenario 2

Andrea is a sport event planner. He planned with his coworker Luca the "HOLI FUN Run" of the 7th of July in Cremona. On the *Run*'s day, Andrea was in France for a meeting so when the meeting ends he went back to his hotel, opened his laptop and went to *Track4Run* web page, searched through the search bar the *Run* that unfortunately was just ended. However he was very happy because looking at the placing table he saw that his friend Marta won.

Use Case

The *Run Watching* use case is analyzed in Table 3.19.

Activity Diagram

The *Run Watching* activity diagram is shown in Figure 3.31.

Mockup

The *Run Watching* mockup is shown in Figure 3.32.

Functional requirements

1. A **Generic user** (that could be *Specator* or *User*) must be able to watch a *Run* that is still in progress or ended;

2. The system must continuously check the positon of the runner in order to keep the map and the placing table updated;

3. The system must be able to show all the runners enrolled in the *Run* thanks to their GPS position;

4. The system must notify the end of a *Run* to a **Generic user**;

5. The system must be able to compute the placing table through the GPS position of the runner enrolled in the *Run*.

Actor	**Generic User** (that could be *Specator* or *User*)
Goal	**[G.14]** and **[G.15]**
Input Condition	A **Generic User** wants to watch a *Run*
Event Flow	1. The **Generic User** opens *Track4Run* service through mobile application or web application; 2. The **Generic User** clicks on the *Watch a Run* button; 3. The **Generic User** looks for a *Run* through the search bar or looking to the proposed ones; 4. The **Generic User** clicks on the *Run* he/she wants to watch.
Output Condition	The system loads the *Run* environment (map, path and placing table) and shows it to the **Generic User**.
Exceptions	• If the **Generic User** looks for a *Run* that is not present in the system, the system notifies the **Generic User** with a warning message; • If the connection of the **Generic User** application is lost and the system couldn't be able to recover it the process goes back to step 3.

Table 3.19: *Run Watching* use case
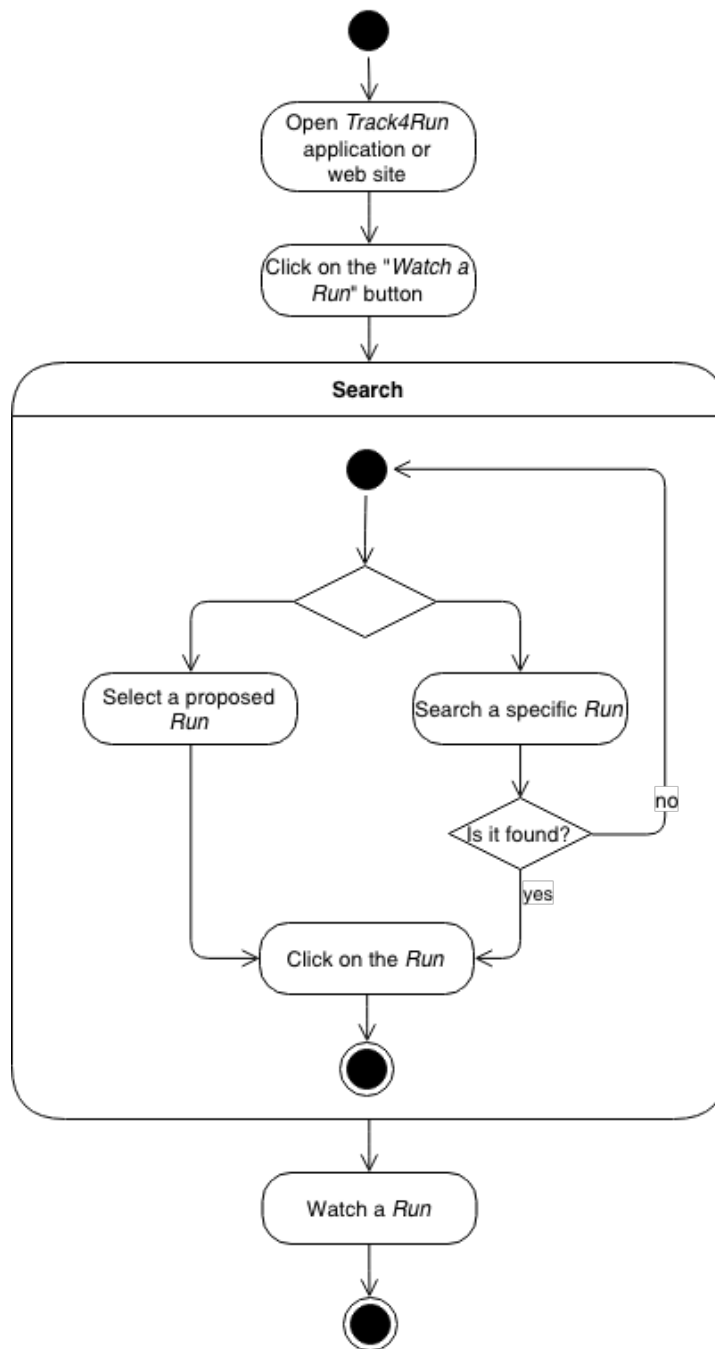
Figure 3.31: *Run Watching* activity diagram from user's point of view

Figure 3.32: *Run Watching* mockup

Section 4

Alloy

This section includes Alloy code that describes the model and checks wether it is consistent or not. At the end of the code, Class Diagram and Alloy generated World are shown.

4.1 Code

```
open util/boolean
2

4 //As in UNIX, time is represented as an offset in seconds
   //from midnight (UTC) on January 1, 1970.
6

8 //Prices are int values because the number of cents paid is
   //represented. (Just divide the value by 100 to find the
10 //cost in euros).

12

   //SmartphoneID is an Int. It does not match the serial
14 //number of the smartphone but it is a value assigned
   //by the system to recognize a device.
16 //If a smartphone is sold to others, it will be assigned a
   //new SmartphoneID.
18

20 //String abstraction
   sig StringTM {}
22
```

```
   //SIGNATURES

24

   //Standard user
26 sig StandardUser
   {
28     email: one StringTM,
       name: one StringTM,
30     surname: one StringTM,
       dateOfBirth: one Int,
32     address: one Address,
       occupation: one Occupation,
34     smartphone: set Smartphone,
   }

36

   //Special user
38 sig SpecialUser
   {
40     corporateEmail: one StringTM,
       businessName: one StringTM,
42     vat: one Int,
       referentsSurname: one StringTM,
44     referentName: one StringTM,
       legalAddress: one Address,
46     billingAddress: one Address,
       sector: one Sector,
48 }


50 //Position
   sig Position
52 {
       latitude: one Int,
54     longitude: one Int,
   }

56

   //Abstract data request
58 abstract sig DataRequest
   {
60     requestID: one Int,
       accepted: one Bool,
62     date: one Int,
       nDownload: one Int,
64     applicant: one SpecialUser,
       payment: lone Payment,
```

```alloy
66   }
     {
68       one payment implies accepted=True
         nDownload>=0
70   }

72   //Request for single user data
     sig SingleUserDataRequest extends DataRequest
74   {
         target: one StandardUser,
76   }

78   //Request for group data
     sig GroupDataRequest extends DataRequest
80   {
         target: set StandardUser,
82   }

84   //Address
     sig Address
86   {
         country: one StringTM,
88       province: one StringTM,
         city: one StringTM,
90       street: one StringTM,
         houseNumber: one Int,
92   }
     {
94       houseNumber>0
     }
96
     //Data of a run
98   sig Run
     {
100      runID: one Int,
         name: one StringTM,
102      organizer: one StandardUser,
         partecipants: set StandardUser,
104      route: set Position,
         city: one StringTM,
106      date: one Int,
         regDeadline: one Int,
108  }
```

```alloy
     {
110      regDeadline<=date
     }

112
     //Registration to a Run
114  sig RunRegistration
     {
116      runner: one StandardUser,
         registration: one Run,
118      date: one Int,
     }

120
     //Smartphone
122  sig Smartphone
     {
124      smartphoneID: one Int,
         bluetoothConnection: one Bool,
126      isWorking: one Bool,
         batteryLevel: one BatteryLevel,
128      isOnCharge: one Bool,
         records: set PositionRecord,
130      sosCall: set SOSCall,
         device: set Device,
132  }
     {
134      isWorking=True implies not (batteryLevel=Empty)
         bluetoothConnection=True implies not (batteryLevel=Empty)
136      batteryLevel=Empty implies bluetoothConnection=False
         batteryLevel=Empty implies isWorking=False
138      isWorking=True implies (some records)
     }

140
     //SOSCall
142  sig SOSCall
     {
144      callID: one Int,
         date: one Int,
146  }

148  //Device
     sig Device
150  {
         deviceID: one Int,
```

101

```alloy
152     bluetoothConnection: one Bool,
        isWorking: one Bool,
154     batteryLevel: one BatteryLevel,
        isOnCharge: one Bool,
156     records: set HealthStatusRecord,
    }
158 {

        isWorking=True implies not (batteryLevel=Empty)
160     bluetoothConnection=True implies not (batteryLevel=Empty)
        batteryLevel=Empty implies bluetoothConnection=False
162     batteryLevel=Empty implies isWorking=False
    }
164

    //Position Record
166 sig PositionRecord
    {
168     time: one Int,
        position: one Position,
170 }

172 //Health Status Record
    sig HealthStatusRecord
174 {
        time: one Int,
176     healthStatus: one HealthStatus,
    }
178

    //Health Status
180 sig HealthStatus
    {
182     minPressure: one Int,
        maxPressure: one Int,
184     heartbeat: one Int,
        health: one Health,
186 }
    {
188     maxPressure>=minPressure
        maxPressure>=0
190     minPressure>=0
        heartbeat>=0
192 }

194 //Payment
```

```alloy
      sig Payment
196   {
          paymentID: one Int,
198       amount: one Int,
          date: one Int,
200   }
      {
202       amount>=0
      }
204
      //Enum BatteryLevel
206   abstract sig BatteryLevel {}
          one sig Empty extends BatteryLevel{}
208       one sig Low extends BatteryLevel{}
          one sig Medium extends BatteryLevel{}
210       one sig High extends BatteryLevel{}

212   //Enum Job Sector
      abstract sig Sector {}
214       one sig Finance extends Sector{}
          one sig Business extends Sector{}
216       one sig Pharmaceutical extends Sector{}
          one sig Engineering extends Sector{}
218       one sig Environment extends Sector{}
          one sig Healthcare extends Sector{}
220       one sig IT extends Sector{}
          one sig Law extends Sector{}
222
      //Enum User Occupation
224   abstract sig Occupation {}
          one sig Student extends Occupation {}
226       one sig Employed extends Occupation{}
          one sig Unemployed extends Occupation{}
228
      //Enum User Health
230   abstract sig Health {}
          one sig Good extends Health{}
232       one sig Bad extends Health{}

234
      //FACTS
236
```

```
238  // Users can't have the same email address
     fact usersCannotHaveTheSameEmailAddress
240  {
         no u1, u2 : StandardUser | u1.email=u2.email and u1!=u2
242      no u1, u2 : SpecialUser | u1.corporateEmail=u2.corporateEmail
             and u1!=u2
244      no u1 : StandardUser, u2 : SpecialUser |
             u1.email = u2.corporateEmail
246  }


248  //No runs with same runID
     fact noRunsSameID
250  {
         no r1, r2 : Run | r1.runID=r2.runID and r1!=r2
252  }


254  //No requests with same requestID
     fact noRequestsSameID
256  {
         no r1, r2 : DataRequest | r1.requestID=r2.requestID and r1!=r2
258  }


260  //No payments with same paymentID
     fact noPaymentsSameID
262  {
         no p1, p2 : Payment | p1.paymentID=p2.paymentID and p1!=p2
264  }


266  //Bad health status implies a SOSCall within 5 seconds
     fact badHealthStatusImpliesASOSCallWithin5Seconds
268  {
         all h : HealthStatusRecord | h.healthStatus.health=Bad
270          implies (one c : SOSCall, u : StandardUser | c.date>=h.time
                 and c.date<=h.time+5 and c in u.smartphone.sosCall)
272  }


274  //No two SOSCall in the same moment for the same user
     fact noTwoSOSCallInSameMomentSameUser
276  {
         no c1, c2 : SOSCall, u1 : StandardUser | c1.date=c2.date
278          and c1 in (u1.smartphone.sosCall)
                 and c2 in (u1.smartphone.sosCall)
280  }
```

```
282  //No two or more SOSCall with same IDCAll
     fact noMoreSOSCallWithSameID
284  {
         no c1, c2 : SOSCall | c1.callID=c2.callID and c1!=c2
286  }


288  //All SOSCalls have Bad health status record
     fact allSOSCallWithABadHealthStatusRecord
290  {
         all  c : SOSCall, u : StandardUser | c in u.smartphone.sosCall
292      implies (one h : HealthStatusRecord, u : StandardUser |
             h.healthStatus.health=Bad and c.date=h.time
294              and h in u.smartphone.device.records)
     }
296
     //All group data requests with more than 999 users are accepted
298  fact allGroupDataRequest1000UsersOrMoreAreAccepted
     {
300      all r : GroupDataRequest | #r.target>=1000
             implies r.accepted=True
302  }


304  //All group data request for less than 1000 users are not accepted
     fact allGroupRequestForLessThan999UserNotAccepted
306  {
         all r : GroupDataRequest | #r.target<1000
             implies r.accepted=False
308  }
310
     //Only accepted group data request can be payed
312  fact OnlyAcceptedGroupDataRequestCanBePayed
     {
314      all p : Payment, r : GroupDataRequest | r.payment=p
             implies r.accepted=True
316  }


318  //Only accepted single user request can be payed
     fact OnlyAcceptedSingleUserDataRequestCanBePayed
320  {
         all p : Payment, r : SingleUserDataRequest | r.payment=p
322          implies r.accepted=True
     }
```

```
324
    //All saved addresses refer to a user
326 fact allSavedAddressesReferToAUser
    {
328     all a : Address, u1 : SpecialUser, u2 : StandardUser |
            a not in u1.legalAddress implies (a in u1.billingAddress
330         or a in u2.address)
        all a : Address, u1 : SpecialUser, u2 : StandardUser |
332         a not in u1.billingAddress implies (a in u1.legalAddress
                or a in u2.address)
334     all a : Address, u1 : SpecialUser, u2 : StandardUser |
            a not in u2.address implies (a in u1.billingAddress
336         or a in u1.legalAddress)
    }
338
    //All payments are made only after the request has been made
340 fact paymentAfterRequest
    {
342     all p : Payment, r : DataRequest | r.payment=p
            and p.date>=r.date
344 }

346 //All downloads are possible only if the request is accepted
    fact allDownloadsAfterRequestAcceptedAndPaid
348 {
        all r : DataRequest | r.nDownload>0
350         implies r.accepted=True
    }
352
    //All downloads are possible only if the request is paid
354 fact allDownloadsAfterRequestAcceptedAndPaid
    {
356     all r : DataRequest, p : Payment | r.nDownload>0
            implies r.payment=p
358 }

360 //Smartphone is working only if it has no empty battery
    fact smartphoneWorkingIfBatteryNotEmpty
362 {
        all s : Smartphone | s.isWorking=True
364         implies s.batteryLevel!=Empty
    }
366
```

```
    //Device is working only if it has no empty battery and
368 //smartphone has not empty battery
    //and bluetoothConnection is On
370 fact deviceWorkingIfBatteryNotEmpty
    {
372     all d : Device, s : Smartphone | d.isWorking=True
            implies (d in s.device and d.batteryLevel!=Empty
374             and s.batteryLevel!=Empty
                and d.bluetoothConnection=True
376             and d.bluetoothConnection=True)
    }
378
    //Max pressure over 170 implies a Bad status
380 fact maxPressureOver170
    {
382     all h : HealthStatus | h.maxPressure>170 implies h.health=Bad
    }
384
    //Min pressure under 100 implies a Bad status
386 fact minPressureOver170
    {
388     all h : HealthStatus | h.minPressure<100 implies h.health=Bad
    }
390
    //Heartbeat over 120 implies a Bad status
392 fact heartbeatUnder120
    {
394     all h : HealthStatus | h.heartbeat>120 implies h.health=Bad
    }
396
    //Heartbeat under 45 implies a Bad status
398 fact heartbeatUnder45
    {
400     all h : HealthStatus | h.heartbeat<45 implies h.health=Bad
    }
402
    //Min pressure over 100 and max pressure under 170 and
404 //heartbeat between 45 and 120 implies Good status
    fact pressureHeartbeatGoodStatus
406 {
        all h : HealthStatus | (h.maxPressure<=170
408         and h.minPressure>=100
            and h.heartbeat>=45 and h.heartbeat<=170)
```

```
410             implies h.health=Good
    }

412

414 //No runs with same name, date, city
    fact allRunsHaveSomerthindDifferent
416 {
        no r1, r2 : Run | r1.name=r2.name and r1.date=r2.date
418         and r1.city=r2.city and r1!=r2
    }

420

    //Registration before registration deadline
422 fact registrationBeforeDeadline
    {
424     all r : RunRegistration, n : Run | r.registration=n
            implies r.date<n.regDeadline
426 }

428 //All PositionRecords refer to a Smartphone
    fact positionRecordsReferToASmartphone
430 {
        all p : PositionRecord | one s : Smartphone | p in s.records
432 }

434 //A PositionRecord refers to only one Smartphone
    fact positionRecordsReferToOnlyOneSmartphone
436 {
        no p : PositionRecord, s1, s2 : Smartphone |
438         p in s1.records and p in s2.records
    }

440

    //All payments refer to a request
442 fact paymentsReferToARequest
    {
444     all p : Payment | one r : DataRequest | p=r.payment
    }

446

    //All Smartphones refer to a StandardUser
448 fact SmartphonesReferToAStandardUser
    {
450     all s : Smartphone | one u : StandardUser | s in u.smartphone
    }

452
```

```
    //All Devices refer to a Smartphone
454 fact deviceReferToASmartphone
    {
456     all d : Device | one s : Smartphone | d in s.device
    }
458
    //All SOSCalls refer to a Smartphone
460 fact sosCallsReferToASmartphone
    {
462     all c : SOSCall | some s : Smartphone| c in s.sosCall
    }
464
    //All Positions refer to a Run or to a PositionRecord
466 fact positionsReferToARunOrToAPositionRecord
    {
468     all p : Position | (some r : PositionRecord| p=r.position
            or some r : Run | p in r.route)
470 }

472 //All run registration for a run are for different runners
    fact runRegistrationSameRunDifferentRunners
474 {
        all r1, r2 : RunRegistration | r1!=r2 and r1.runner=r2.runner
476         implies r1.registration!=r2.registration
    }
478


480 // PREDICATES


482

    //Special users can make more than one request
484 pred specialUsersCanMakeMoreThanOneDataRequest
    {
486     some r1, r2 :  DataRequest |
            r1.applicant=r2.applicant and r1!=r2
488 }

490 //A user can partecipate in more than one run
    pred usersCanPartecipateInMoreThanOneRun
492 {
        some r1, r2 :  RunRegistration |
494         r1.runner=r2.runner and r1!=r2
    }
```

```
496

498   //ASSERTIONS

500
      //No accepted group data request with less
502   //than 1000 special users
      assert noLessThan1000UsersInGroupDataRequests
504   {
          no r : GroupDataRequest | #r.target<1000
506           and r.accepted=True
      }
508
      //No payment for not accepted requests
510   assert noPaymentForNotAcceptedSingleUserDataRequests
      {
512       no p : Payment, r : SingleUserDataRequest |
              p in r.payment and r.accepted=False
514   }

516   //No SOSCall without a Bad status in previous 5 seconds
      assert noSOSCallWithoutBadStatus
518   {
          all c : SOSCall, u : StandardUser | c in u.smartphone.sosCall
520           implies (one h : HealthStatusRecord |
                  h in u.smartphone.device.records
522               and c.date>=h.time and c.date<=h.time+5
                  and h.healthStatus.health=Bad)
524   }

526   //No group request for 1000 users or more not accepted
      assert noGroupRequestsMoreThan1000UsersNotAccepted
528   {
          no r : GroupDataRequest | #r.target>1000
530           and r.accepted=False
      }
532
      //No saved addresses not used
534   assert noSavedAddressesNotUsed
      {
536       no a : Address, u1 : SpecialUser, u2: StandardUser |
              (a not in u1.legalAddress)
538           and (a not in u1.billingAddress)
```

```
                and (a not in u2.address)
540  }

542  //No requests are paid before the data request
     assert noPaymentBeforeRequest
544  {
         no p : Payment, r : DataRequest |
546          r.payment=p and p.date<r.date
     }
548
     //No download before the request is accepted
550  assert noDownloadBeforeRequestAcceptedAndPaid
     {
552      all r : DataRequest, p : Payment |
             r.accepted=False or (p not in r.payment)
554          implies r.nDownload=0
     }
556
     //No device isWorking if smartphone has empty battery
558  assert noDeviceIsWorkingIfSmartphoneHasEmptyBattery
     {
560      no d : Device, s : Smartphone | d in s.device
             and d.isWorking=True and s.batteryLevel=Empty
562  }

564  //No health status with max pressure lower than min pressure
     assert noHealthStatusMaxPressureLowerThanMinPressure
566  {
         no h: HealthStatus | h.maxPressure<h.minPressure
568  }

570  //No Good health status with Bad values
     assert noGoodHealthStatusWithBadValues
572  {
         no h: HealthStatus | h.health=Good
574          and (h.maxPressure>170 or h.minPressure<100
                 or h.heartbeat>120 or h.heartbeat<45)
576  }

578  //No Bad health status with Good values
     assert noBadHealthStatusWithGoodValues
580  {
         no h: HealthStatus | h.health=Bad
```

```
582          and (h.maxPressure<=170 and h.minPressure>=100
                 and h.heartbeat<=120 and h.heartbeat>=45)
584 }

586 //No run registration after registration deadline
    assert noRegistrationAfterDeadline
588 {
        no r : RunRegistration, n : Run | r.registration=n
590         and r.date>n.regDeadline
    }
592
    //No HealthstatusRecord without a device
594 assert noHealthStatusRecordWithoutDevice
    {
596     no h : HealthStatusRecord | no d : Device | h in d.records
    }
598

600 //No Position Record without a smartphone
    assert noPositionRecordWithoutSmartphone
602 {
        no p : PositionRecord | no s : Smartphone | p in s.records
604 }

606 //No Payment without data request
    assert noPaymentWithoutDataRequest
608 {
        no p : Payment | no r : DataRequest | p in r.payment
610 }

612 //No smartphone without user
    assert noSmartphoneWithoutUser
614 {
        no s : Smartphone | no u : StandardUser | s in u.smartphone
616 }

618 //No SOSCall without smartphone
    assert noSOSCallWithoutSmartphone
620 {
        no c : SOSCall | no s : Smartphone | c in s.sosCall
622 }

624 //No device without smartphone
```

```
     assert noDeviceWithoutSmartphone
626  {
         no d : Device | no s : Smartphone | d in s.device
628  }

630  //No position without Run or PositionRecord
     assert noPositionWithoutRunOrPositionRecord
632  {
         no p : Position | no r : Run, pr : PositionRecord |
634          p in r.route or p=pr.position
     }
636
     //No more than one registration for a run for the same user
638  assert noTwoRegSameUserSameRun
     {
640      no r1, r2 : RunRegistration | r1.registration=r2.registration
             and r1.runner=r2.runner and r1!=r2
642  }


644
     run specialUsersCanMakeMoreThanOneDataRequest
646  run usersCanPartecipateInMoreThanOneRun
     check noLessThan1000UsersInGroupDataRequests
648  check noPaymentForNotAcceptedSingleUserDataRequests
     check noSOSCallWithoutBadStatus
650  check noGroupRequestsMoreThan1000UsersNotAccepted
     check noSavedAddressesNotUsed
652  check noPaymentBeforeRequest
     check noDownloadBeforeRequestAcceptedAndPaid
654  check noDeviceIsWorkingIfSmartphoneHasEmptyBattery
     check noHealthStatusMaxPressureLowerThanMinPressure
656  check noGoodHealthStatusWithBadValues
     check noBadHealthStatusWithGoodValues
658  check noRegistrationAfterDeadline
     check noHealthStatusRecordWithoutDevice
660  check noPositionRecordWithoutSmartphone
     check noPaymentWithoutDataRequest
662  check noSmartphoneWithoutUser
     check noSOSCallWithoutSmartphone
664  check noDeviceWithoutSmartphone
     check noPositionWithoutRunOrPositionRecord
666  check noTwoRegSameUserSameRun
```

```
668
    pred show(){}
670
    run show
```

4.2   Results

Executing "Run specialUsersCanMakeMoreThanOneDataRequest"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16579
vars. 1521 primary vars. 41961 clauses. 73ms.
Instance found. Predicate is consistent. 110ms.

   Executing "Run usersCanPartecipateInMoreThanOneRun"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16579
vars. 1521 primary vars. 41961 clauses. 71ms.
Instance found. Predicate is consistent. 98ms.

   Executing "Check noLessThan1000UsersInGroupDataRequests"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 0 vars.
0 primary vars. 0 clauses. 70ms.
No counterexample found. Assertion may be valid. 0ms.

   Executing "Check noPaymentForNotAcceptedSingleUserDataRequests"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16562
vars. 1521 primary vars. 41896 clauses. 97ms.
No counterexample found. Assertion may be valid. 18ms.

   Executing "Check noSOSCallWithoutBadStatus"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16776
vars. 1521 primary vars. 42909 clauses. 72ms.
No counterexample found. Assertion may be valid. 29ms.

   Executing "Check noGroupRequestsMoreThan1000UsersNotAccepted"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 0 vars.
0 primary vars. 0 clauses. 50ms.
No counterexample found. Assertion may be valid. 1ms.

114

Executing "Check noSavedAddressesNotUsed"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16454
vars. 1524 primary vars. 41696 clauses. 48ms.
No counterexample found. Assertion may be valid. 11ms.

Executing "Check noPaymentBeforeRequest"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16973
vars. 1521 primary vars. 43394 clauses. 67ms.
No counterexample found. Assertion may be valid. 74ms.

Executing "Check noDownloadBeforeRequestAcceptedAndPaid"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16629
vars. 1521 primary vars. 42008 clauses. 61ms.
No counterexample found. Assertion may be valid. 37ms.

Executing "Check noDeviceIsWorkingIfSmartphoneHasEmptyBattery"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16581
vars. 1521 primary vars. 41928 clauses. 71ms.
No counterexample found. Assertion may be valid. 10ms.

Executing "Check noHealthStatusMaxPressureLowerThanMinPressure"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16803
vars. 1518 primary vars. 43084 clauses. 65ms.
No counterexample found. Assertion may be valid. 65ms.

Executing "Check noGoodHealthStatusWithBadValues"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 0 vars.
0 primary vars. 0 clauses. 52ms.
No counterexample found. Assertion may be valid. 0ms.

Executing "Check noBadHealthStatusWithGoodValues"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 0 vars.
0 primary vars. 0 clauses. 48ms.
No counterexample found. Assertion may be valid. 1ms.

Executing "Check noRegistrationAfterDeadline"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16973
vars. 1521 primary vars. 43394 clauses. 58ms.
No counterexample found. Assertion may be valid. 57ms.

Executing "Check noHealthStatusRecordWithoutDevice"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 47ms.
No counterexample found. Assertion may be valid. 30ms.

Executing "Check noPositionRecordWithoutSmartphone"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 55ms.
No counterexample found. Assertion may be valid. 20ms.

Executing "Check noPaymentWithoutDataRequest"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 40ms.
No counterexample found. Assertion may be valid. 6ms.

Executing "Check noSmartphoneWithoutUser"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 68ms.
No counterexample found. Assertion may be valid. 36ms.

Executing "Check noSOSCallWithoutSmartphone"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 57ms.
No counterexample found. Assertion may be valid. 7ms.

Executing "Check noDeviceWithoutSmartphone"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16532
vars. 1518 primary vars. 41848 clauses. 42ms.
No counterexample found. Assertion may be valid. 5ms.

Executing "Check noPositionWithoutRunOrPositionRecord"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16583
vars. 1518 primary vars. 41932 clauses. 52ms.
No counterexample found. Assertion may be valid. 30ms.

Executing "Check noTwoRegSameUserSameRun"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16613
vars. 1521 primary vars. 42055 clauses. 42ms.
No counterexample found. Assertion may be valid. 16ms.

Executing "Run show"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20 16493
vars. 1515 primary vars. 41785 clauses. 53ms.
Instance found. Predicate is consistent. 78ms.
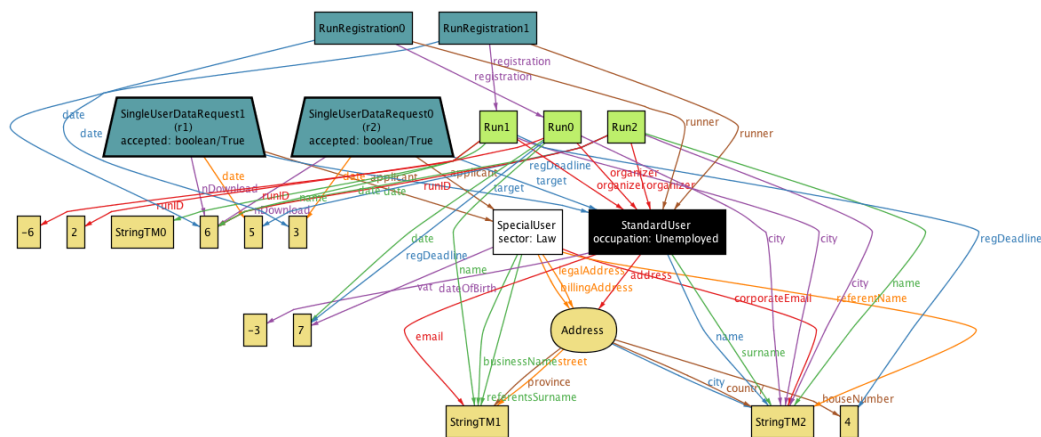
4.3 Generated World

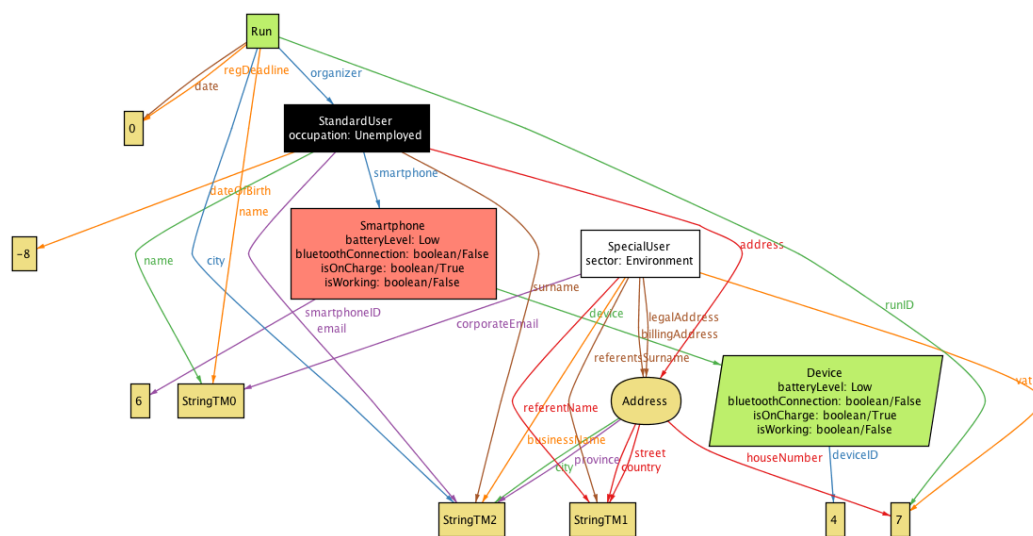Figure 4.1: An example of *Generated World*

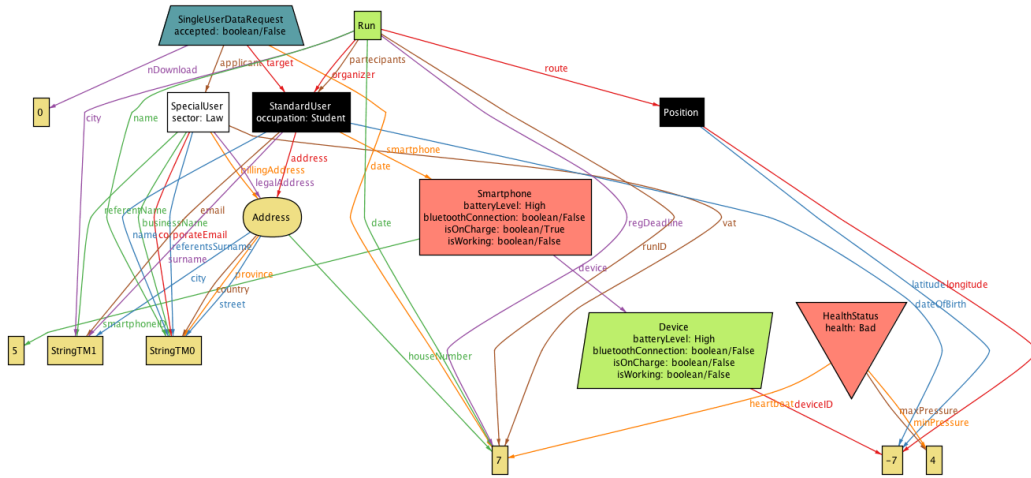Figure 4.2: Another example of *Generated World*

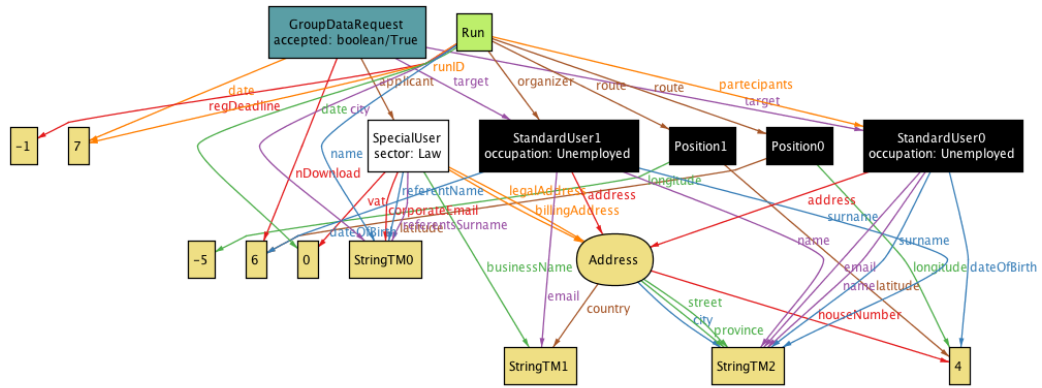Figure 4.3: Another example of *Generated World*

Figure 4.4: Another example of *Generated World*. In this case in order to generate this World, due to computational and representation problems, the minimum allowed size of a group of people for a group request has been changed from 1000 to 2.
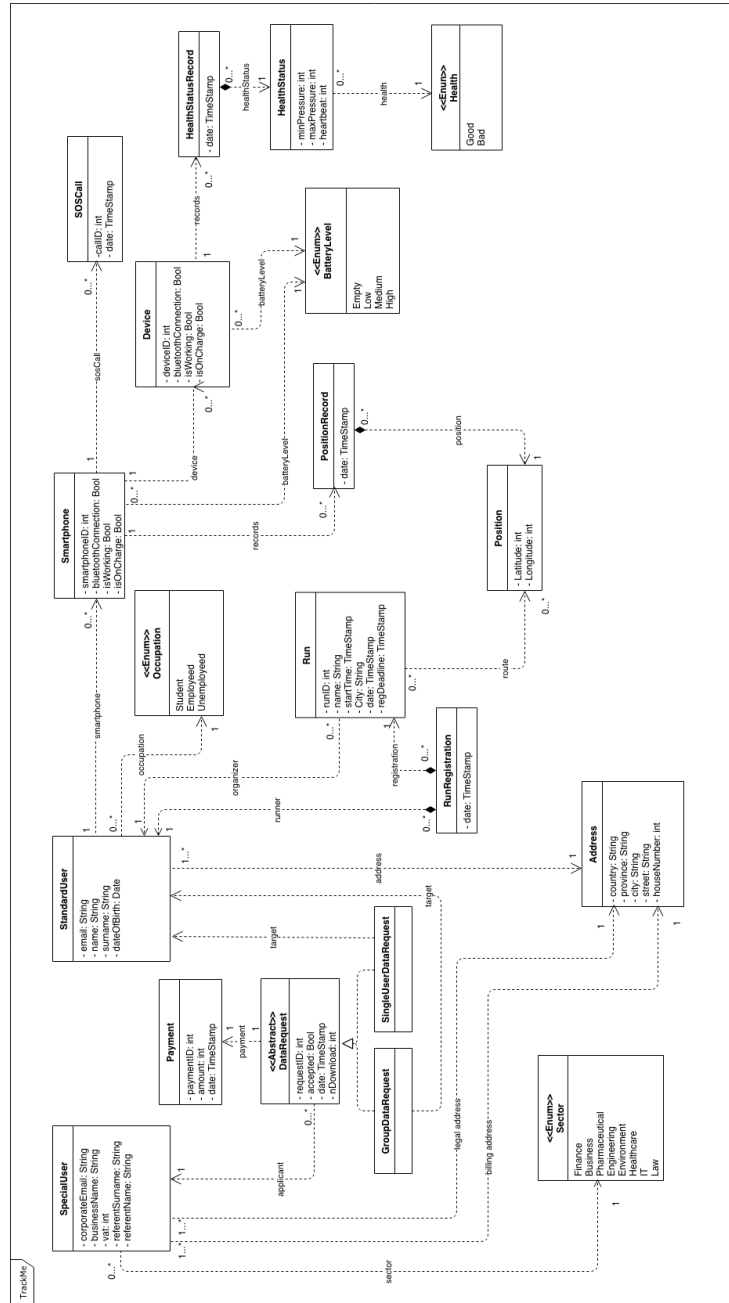
4.4   Class Diagram

Figure 4.5: *Class diagram* of the structure of the system-to-be.

Section 5

Effort Spent

5.1   Michele Gatti

Task	Hours
Purpose and Goals	1
Product Perspective and Product Functions	6
User Characteristics and Constraints	2
Assumptions and Dependencies	3
The World and the Machine	3
Team revision	1
Class Diagram	6
Alloy	14
Team work	2
Total	**38**

5.2   Federica Gianotti

Task	Hours
Purpose and Goals	4
Scope, Definitions, Acronyms and Abbreviations	2
Team revision	1
Functional Requirements	14
Functional Requirements and Mockup revision	4
Activity Diagrams	4
Class Diagram	2
Alloy	2
Team work	2
Final revision	3
Total	**38**

5.3   Mathyas Giudici

Task	Hours
GitHub and LaTeX setup	*2* *
Purpose and Goals	2
Scope, Definitions, Acronyms and Abbreviations	2
Team revision	1
Functional Requirements	14
User Interface Mockup	4
Functional Requirements and Mockup revision	4
Activity Diagrams	3
Class Diagram	2
Alloy	2
Team work	2
Final revision	2
Total	**38**

* : GitHub and LaTeX setup hours are not counted in the total of the hours

Appendix A

Appendix

A.1 Software and Tools

- L^AT_EX used to build this document;

- *GitHub* used to manage the different versions of this document;

- *draw.io* used to draw diagrams;

- *Balsamiq Mockups 3* used to draw mock-ups;

- *Alloy Analyzer* used tp analyze our specifications.

A.2 Changelog

- **1.0** : First release of this document;

- **2.0** : Corrected *Create a Run* activity diagram;

- **2.0** : Corrected *Fist Individual Log In* and *Change device*;

- **2.0** : Corrected *Use Case Diagram*;

- **2.0** : Corrected *Reliability requirement*;

- **2.0** : Corrected *Class Diagram* and *Alloy.*

Bibliography

[1] ISO/IEC/IEEE 29148:2011 *Systems and software engineering - Life cycle processes - Requirements engineering*

[2] IEEE 830:1998 *Recommended Practice for Software Requirements Specifications*

[3] M.Jackson & P. Zave, *The World and The Machine*, 1995

[4] Elisabetta Di Nitto - Software Engineering 2 Slides (AY 2018/2019) *Project goal, schedule and rules*