**Table of Contents**

Table of Contents . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 2

1. Introduction . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
   1. Purpose
   2. Scope
      1. Description of the problem
      2. Current system
      3. Goals
   3. Definitions, acronyms and abbreviations
      1. Definitions
      2. Acronyms
      3. Abbreviations
   4. Revision history
   5. Reference documents
   6. Document structure
2. Overall Description
   1. Product perspective
   2. Product functions
   3. User characteristics
   4. Assumptions, dependencies and constraints

**1. Introduction**1.1 Purpose  
  
This is the Requirement Analysis and Specification Document (RASD).  
As defined by the IEEE Standards, the goals of this document are to give a complete description of the system-to-be, in terms of both functional and non-functional requirements, to analyse the needs of the costumer in order to model the system and to show the constraints the system itself will be subjected to.

1.2 Scope  
  
*1.2.1 Definition of the problem*

*1.2.1.1 Data4Help*

***TrackMe*** is a company that wants to develop a software-based service allowing third parties to monitor the location and health status of individuals. This service is called **Data4Help**. The service supports the registration of individuals who, by registering, agree that TrackMe acquires their data

(data acquisition can happen through smartwatches or similar devices). Also, it supports the registration of third parties. After registration, these third parties can request:

* Access to the data of some specific individuals (we can assume, for instance, that they know an individual by his/her social security number or fiscal code in Italy). In this case, TrackMe passes the request to the specific individuals who can accept or refuse it.
* Access to anonymized data of groups of individuals (for instance, all those living in a certain geographical area, all those of a specific age range, etc.). These requests are handled directly by TrackMe, that will accept any request for which the number of individuals whose data satisfy the request is higher than 1000, in order to protect the identity of its clients.

As soon as a request for data is approved, TrackMe makes the previously saved data available to the third party. Also, it allows the third party to subscribe to new data and to receive them as soon as they are produced.

*1.2.1.2 AutomatedSOS*

TrackMe decides to build a new service, called **AutomatedSOS**, on top of Data4Help. 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.   
  
*1.2.2 Current System*  
  
TrackMe has no pre-existing services that need to be integrated with the new ones, Data4Help and AutomatedSOS. The whole system-to-be is going to be build completely from scratch, therefore no current system will be considered during the design process.  
  
*1.2.3 Goals  
  
1.2.3.1 Data4Help*

* [G1]: Allow an individual to become a Registered User, by providing credentials and agreeing to TrackMe's privacy policy.
* [G2]: Allow a third-party company to become a Registered Third Party by providing credentials.
* [G3]: Allow a Registered Third Party to have an access to a specific, properly identified Registered User's data, if and only if the said User gives his/hers consent to that action.
* [G4]: Allow a Registered Third Party to access anonymized data of groups of individuals, if and only if the said group comprehends at least 1000 individuals.
* [G5]: By previous approval by the use send to a Registered Third Party with a subscription to new data the new version of the saved data, as soon as this version is produced.

*1.2.1.2 AutomatedSOS*

* [G6]: Allow a Registered User to become a Registered AutomatedSOS User
* [G7]: Send an ambulance to a Registered AutomatedSOS User's location, if and only if his/hers health parameters drop below a certain threshold.

1.3 Definitions, acronyms and abbreviations  
  
*1.3.1 Definitions*

* ***Registered User***: an individual registered to the Data4Help service, and that has therefore given consent to TrackMe to acquire their data.
* *[Registered User's]* ***Data***: information about a RU's location and health status. As no more precise information are reported in the problem description, further information regarding the definition of “health status” are reported in section XXXX
* ***Current Saved Data***: it's the data currently saved in the Data4Help system.
* ***Registered Third Party***: a third-party company, independent form TrackMe, that registered itself to the Data4Help service.
* *[Health parameters]* ***threshold***: a precisely defined threshold of the health parameters: if these parameters descend below the said threshold, an individual is considered to be in danger. As no more precise details are reported in the problem description, further information regarding the definition of “heath parameters' threshold” is reported in section XXXX.
* ***Consenting Registered User*** *[w.r.t. a request from a RTP]*: a RU that has accepted a request for his/hers data by a RTP.  
    
  *1.3.2 Acronyms*
* *RU*: Registered User
* CRU: Consenting Registered User
* *RAU*: Registered AutomatedSOS User
* *RTP*: Registered Third Party
* RASD: Requirement Analysis and Specification Document

*1.3.3 Abbreviations*

* G*n*: the goal number *n.*
* *Dn*: the domain assumption number *n.*
* *Rn*: the functional requirement number *n.*

1.4 Revision history

At the moment there is no revision history to show, as this current version is the only existent one.

1.5 Reference documents

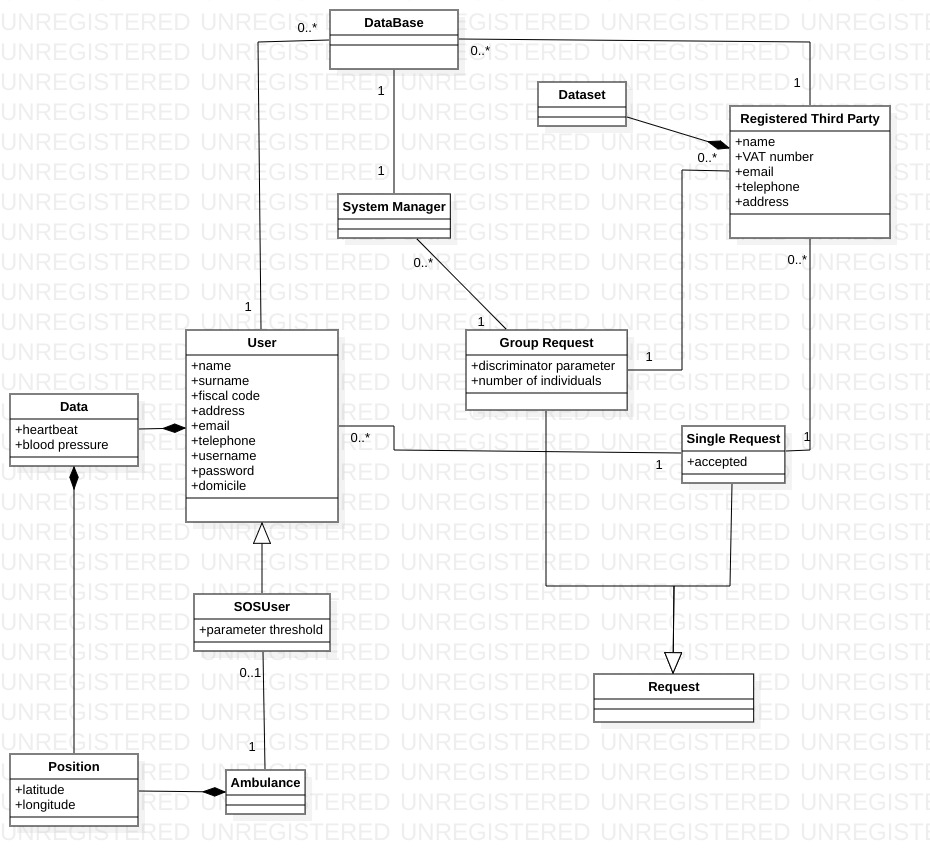
* Specification document: “Mandatory Project Assignment AY 2018-2019”
* IEEE Std 830-1993 - IEEE Guide to Software Requirements Specifications
* IEEE Std 830-1998 - IEEE Recommended Practice for Software Requirements Specifications

1.6 Document structure  
  
This document is composed by five main parts, including a final appendix.

1. ***Introduction.*** In the first part the problem is presented, with all the main basic information needed to fully understand the scope and the objective of the document itself. All the goals of the system-to-be are already specified and explained.
2. ***Overall Description.*** The second part is a comprehensive general description of the system: there is the presentation of its boundaries and the specification the main actors that will use the system itself.
3. ***Specific Requirements.*** The third part is a precise specification of the aspects already discussed in the previous section, in order to clarify the details for the development team. There is here an identification of every specific requirement, functional or non-functional. Furthermore, here is the use case diagrams, use cases, scenarios, performance analysis, design constraints and finally the attributes of the software system.
4. ***Analysis of the System.*** The fourth part provides a more formal analysis of the system, using a formal approach (using Alloy). This more rigorous approach is used to define the critical aspects of the system.
5. ***Appendix.*** The fifth and last part comprehends a list the tools used to redact this document and its contents and a detailed report of the hours and efforts spent by each member of the group during the project.

**2. Overall Description**  
2.1 Product perspective

The whole system is going to be developed from scratch, since there are not any pre-existing systems to integrate with.



2.2 Product functions  
  
2.2.1 Ambulances management  
The ambulances previously mentioned in the description of the AutomatedSOS service are not logistically related in any way to the new system, with the only exception being the dispatching of an ambulance by the system itself. Every other action related to the ambulance life cycle, such as for example the maintenance routine, is not supported in any way by the system.

2.3 User characteristics

*2.3.1 Actors*

* ***Registered User***: often referred to as simply “User”, a RE is an individual who has successfully registered him/herself to the Data4Help service, and therefore can log in to the system. His/her data will be sent to TrackMe.
* ***AutomatedSOS Registered User***: a RE that also subscribed to the AutomatedSOS service, on top of Data4help, and that can therefore utilise all the AutomatedSOS features.
* ***Registered Third Party***: a third-party company, independent form TrackMe, that registered itself to the Data4Help service. It can ask for a specific individual's data, or for anonymized data from a batch of individuals.
* ***System Manager***: The software in charge of the system: it makes sure that the system guarantees the privacy protection and operates properly.

2.4 Assumptions, dependencies and constraints

*2.4.1 Assumptions*

In order to clarify some points that lacked precision in the specification document are hereby introduced the following assumptions.

*2.4.1.1 Text Assumptions*

* In order to become a Registered User, and individual has to provide the following information while registering: name, surname, address, email, telephone number and a unique form of identification.
* The unique form of identification needed for individuals is here considered to be the individual's fiscal code.
* In order to become a Registered Third Party, a company has to provide the following information while registering: name, address, telephone number, email and a unique form of identification.
* The unique form of identification needed for third party companies is here considered to be its VAT number.
* The “health parameters” previously mentioned are considered to be the following: heartbeat rate, blood pressure, respiratory rate.
* The “health parameters' threshold” previously mentioned are considered to be precisely defined values of said parameters, tuned by TrackMe itself, based analysing the normal parameters of each individual.
* X: The ambulance must be able to change destination as the system manager might update the position. (The illness may happen when the user is in the car, which is moving)

*2.4.1.2 Domain Assumptions*

* [D1]: Users' locations are correctly retrieved by GPS.
* [D2]: The user must have an email prior to registration
* [D2]: Users' health parameters retrieved by smartwatches or similar devices are the actual health parameters of the user.
* [D3]: The users' unique form of identification, here considered to be the individual's fiscal code, are actually unique for each person.
* [D4] The Registered Third Party's unique form of identification, here considered to be its VAT number, is actually unique for each company.
* [DX]: When the order to dispatch an ambulance is correctly sent, then the said ambulance will surely arrive to the target location.

*2.4.2 Dependencies*

*2.4.2.1 Hardware dependencies*

* Android or iOS smartphone with GPS.
* Any smartwatch-compatible device that can scan and transmit the individual's health status.
* 2G/3G/4G connection.

*2.4.3 Constraints*  
  
*2.4.3.1 Regulatory Policies*Data4Help will ask for users' permission to access to their location and health status information, in order to acquire, store and eventually transmit their data to third-party companies.   
As mentioned before, Data4Help grants the user's anonymity when transmitting his/her data as part of a bigger batch of individuals, and directly asks the user if a Registered Third Party is interested in his/her data in particular, in order to acquire the individual's consent.

*2.4.3.1 Interface to other applications*

There will be no public interfaces, therefore third party services won't be able to interoperate with neither Data4Help or AutomatedSOS.

**3. Specific Requirements**

3.1 External Interface Requirements

In this section we will specify how the modules of this system will interact the one to each other.

* + 1. User Interfaces

Since the application only gather data about health status and position it only needs to be downloaded from the store, to allow registration of the type (individual or third party), check login credentials and then run in background. If it is a third-party it can request access to user data.

Hardware Interfaces

The hardware component (smartwatches) will be worn every day.

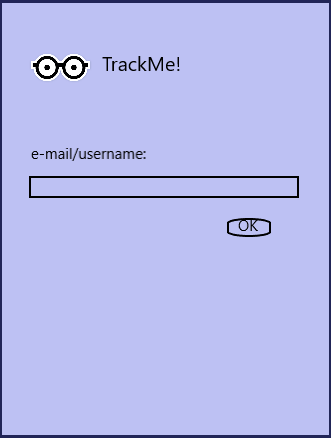
* + 1. Software Interfaces

The core of the system will reside in a central computer (server) which registers the user in the database, allows the safe login, receive updated on a regular basis. Through the app the incoming requests will be therefore processed then allowed or denied.

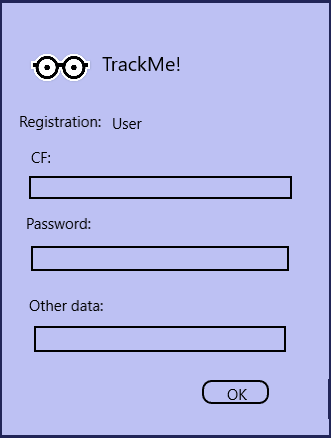
* + 1. Communication Interfaces

The connection between the modules employs the Web.

GUI for user and third party

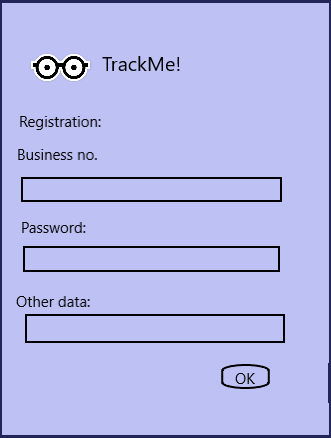


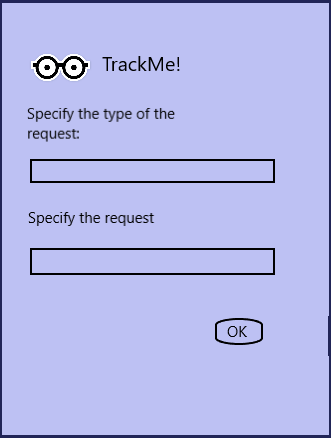
User GUIs





Third party GUI



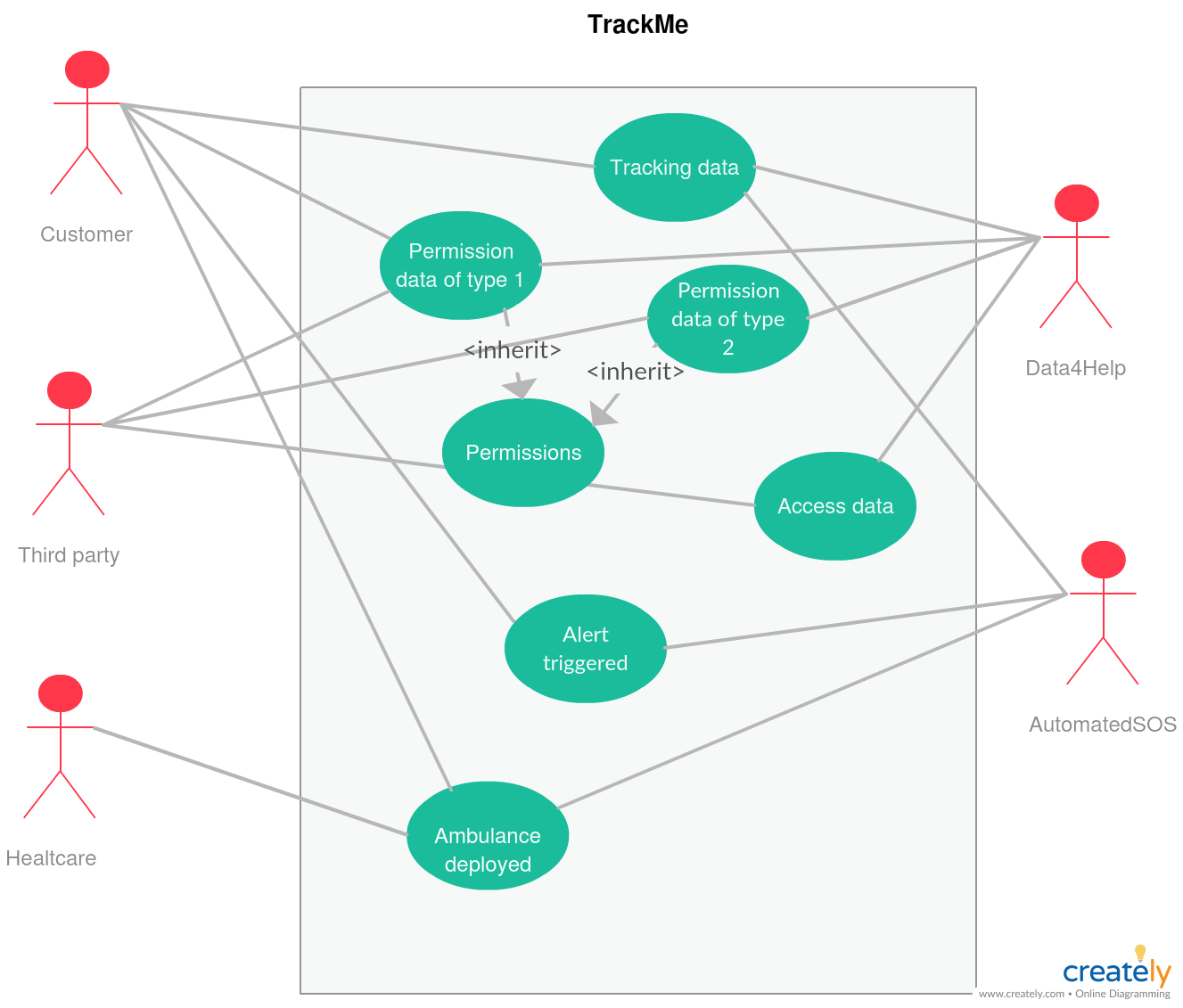


3.2 Functional Requirements

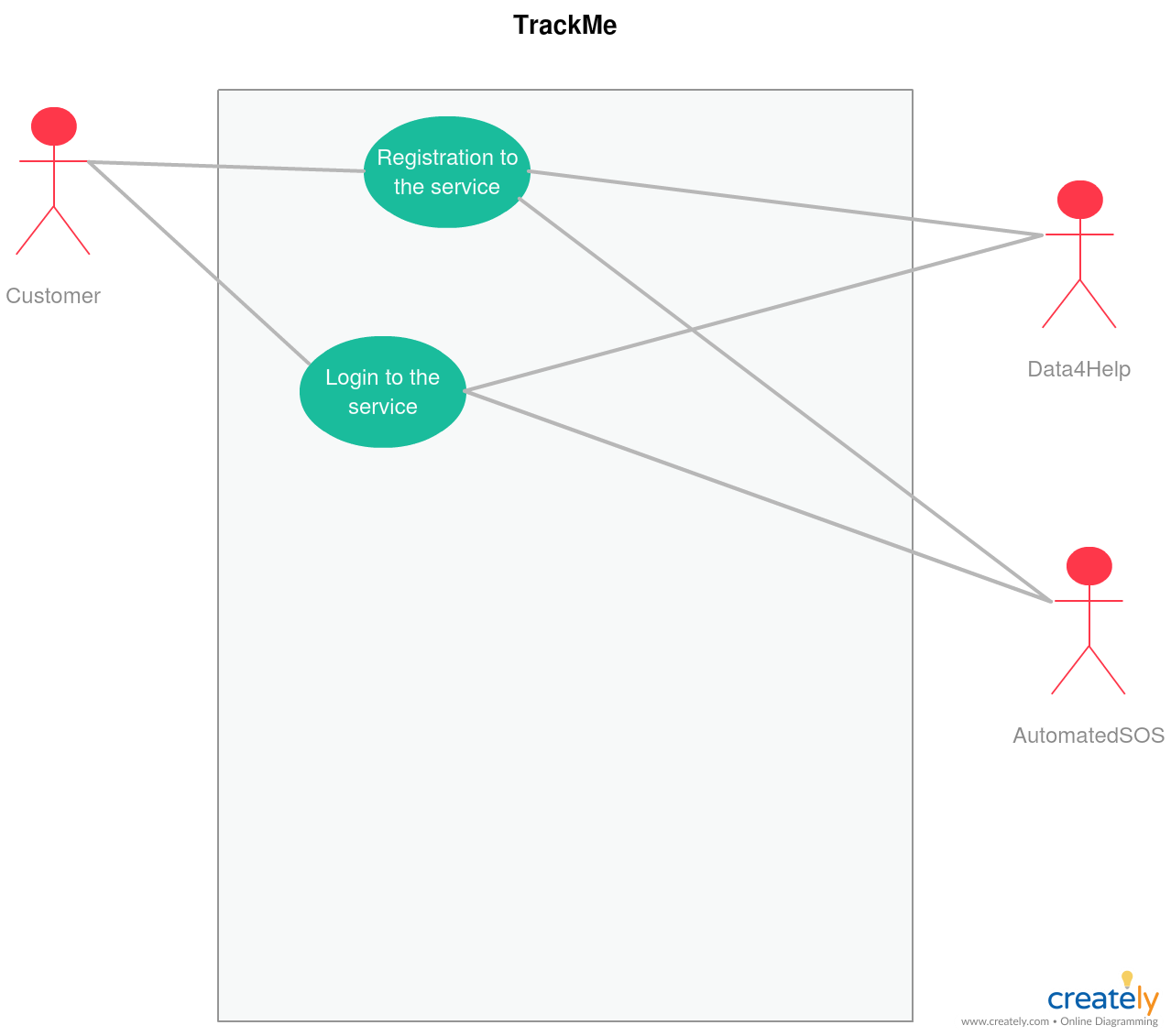
Here we will define the use case diagrams, use cases and the full state diagrams, the Goal we expect to fulfil, and the mapping on the requirements.

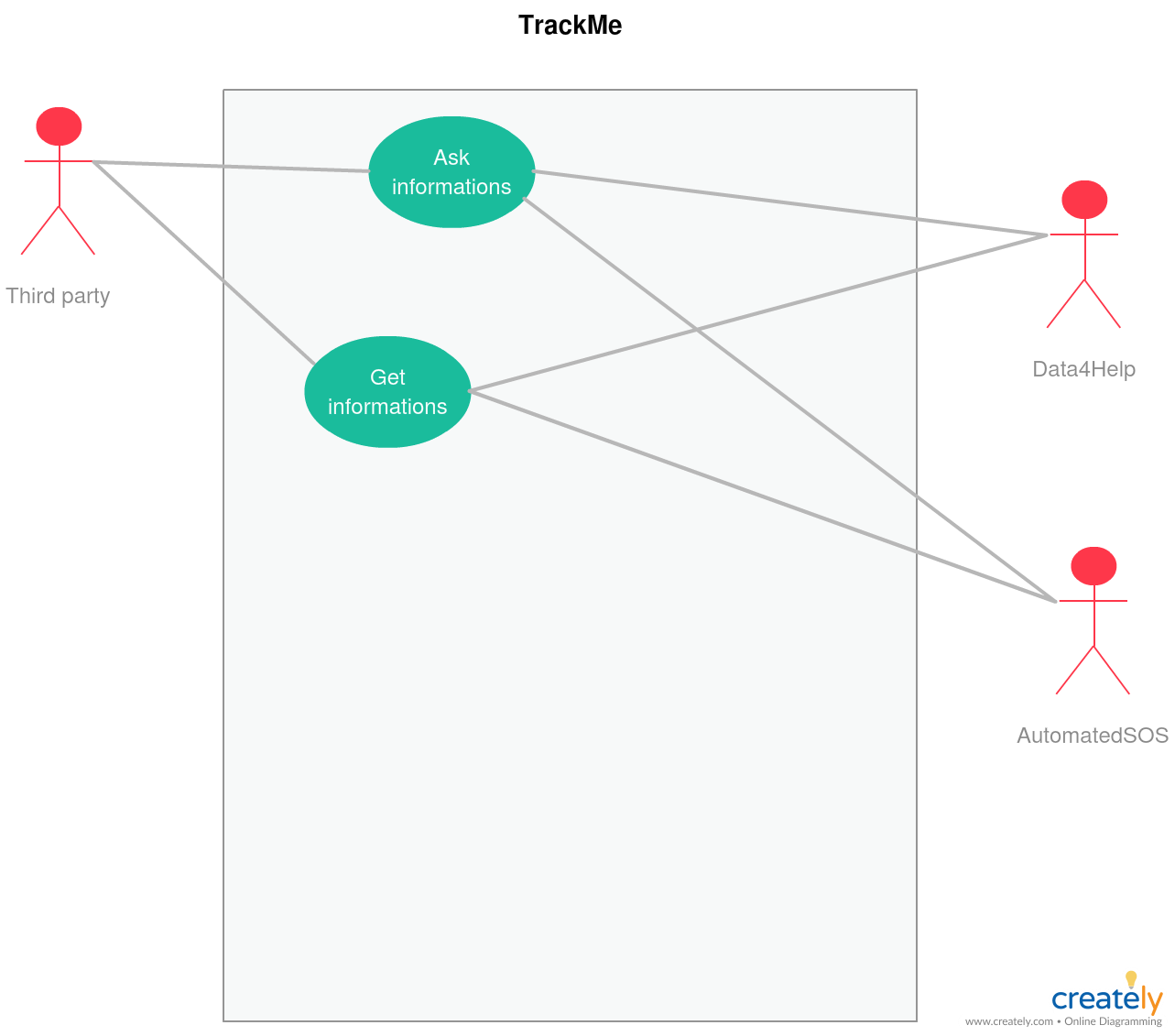
3.2.1 Use case diagrams

3.2.1.1



3.2.1.2 User use case diagram



3.2.1.2 User use case diagram

3.2.2 Scenarios

* + - * 1. Alice sees this new app, and she thinks she can give it a try. So she suggest to her boyfriend Bob to join her and sign up into the system. They then provide the requested data and agree on the permission of tracking the position and the personal data.  
           After registration they login to the platform and wear the smartwatch.
        2. Tracking business is a newly started business, which acquires data from the users and share them. Thomas is the head of this firm and he thinks it might be a good idea to join the platform. So he download the app, and then he register himself. The idea of registering as a user, not only as a third party came across his mind, but he prefers to not be tracked.   
           After the data are entered into the system, he can start by logging in the platform.
        3. The business of Thomas in now connected to the network of TrackMe. He decides then to make his first request. He starts by choosing a type of request: it can be a generalized one or a specific one.   
           It follows now the confirmation by the system and the user, eventually, and if it is approved Thomas can start doing his business exploiting the data gathered.
        4. It happens now that Thomas’ business makes a request. This request reaches the system. Let’s suppose the request, is of a specific user. Then the system request the user to choose if it will allow the business to get his data. The user will be notified by that and he can choose yes or no. The user will be able to deny the permission later, eventually. Suppose instead that Thomas wants to inquire about a generalized amount of data. Then the system will get the number of requests and evaluates whether the request is appropriate or not.
        5. Bob is an old user. He doesn’t love the fact that data is always being sent but appreciates the emergency system. He is sure that if there will be an app which performs only the emergency system he will choose that one. But as long as this is the only available app, he chose this one. Out of the blue he doesn’t fell very well, his breath frequency is becoming low, and the pressure drops in free fall. But Bob is provident, the smartwatch he always wears detects the imminent problem and send the emergency message to the system. The system then will promptly send an ambulance to rescue him.

3.2.3 Use cases

* + - 1. Individual registration

|  |  |
| --- | --- |
| ACTORS | Future user |
| GOALS | GX |
| INPUT CONDITIONS | -The user is not previously registered |
| EVENTS FLOW | -The user inserts e-mail  -The user clicks on the “Enter” button  -The user clicks on the individual button  -The user inserts its password and personal data (e.g. fiscal code) which will be validated prior to effective creation of account. |
| OUTPUT CONDITIONS | -The user is granted service of the application by logging-in |
| EXCEPTIONS | -Internet connection interrupted  -Password chosen doesn’t fulfil the constraints on it  -User data are not well formed  Exception 1 is handled by caching the data and sending when the service is available again  Exception 2 and 3 are handled by looping the insert request until correct data is provided |

* + - 1. User login

|  |  |
| --- | --- |
| ACTORS | Current user |
| GOALS | G2 |
| INPUT CONDITIONS | The user is already registered |
| EVENTS FLOW | -The user inserts e-mail and password  -The user clicks enter  -The system detects that the user is already registered  -The app shows the “Stop tracking” button and the notification bar |
| OUTPUT CONDITIONS | Tracking starts |
| EXCEPTIONS | -Internet connection interrupted  Exception 1 is handled stopping the service |

* + - 1. Third party registration

|  |  |
| --- | --- |
| ACTORS | Future third party user |
| GOALS | G4 |
| INPUT CONDITIONS | -The user is not previously registered |
| EVENTS FLOW | -The user inserts e-mail and password  -The user clicks on the “Enter” button  -The user clicks on the third-party button  -The user inserts its personal data (e.g. Business number) which will be validated prior to effective creation of account. |
| OUTPUT CONDITIONS | -The user is granted service of the application by logging-in |
| EXCEPTIONS | -Internet connection interrupted  -Password chosen doesn’t fulfil the constraints on it  -User data are not well formed  Exception 1 is handled by caching the data and sending when the service is available again  Exception 2 and 3 are handled by looping the insert request until correct data is provided |

* + - 1. Third party login

|  |  |
| --- | --- |
| ACTORS | Current user |
| GOALS | G5 |
| INPUT CONDITIONS | The user is already registered |
| EVENTS FLOW | -The user inserts e-mail and password  -The user clicks enter  -The system detects that the user is already registered  -The app shows the “Stop tracking” button and the notification bar |
| OUTPUT CONDITIONS | User can start requesting data |
| EXCEPTIONS | -Internet connection interrupted  Exception 1 is handled stopping the service |

* + - 1. App tracking at work

|  |  |
| --- | --- |
| ACTORS | User |
| GOALS | G2 |
| INPUT CONDITIONS | User successfully logged-in |
| EVENTS FLOW | No further actions needed here |
| OUTPUT CONDITIONS | Data being forwarded to the server |
| EXCEPTIONS | -GPS signal absent  -Sensors not working properly  All exceptions are handled by discarding the data |

* + - 1. Request to access private data

|  |  |
| --- | --- |
| ACTORS | Third party |
| GOALS | G5 |
| INPUT CONDITIONS | Third party is logged in |
| EVENTS FLOW | Third party sends request to the server specifying CF and the type of data he wants to access with “-subscribe” option |
| OUTPUT CONDITIONS | The request is sent |
| EXCEPTIONS | -CF does not exist  -Malformed request  Al exception are handled by notifying the issues to the business |

* + - 1. User permission about the personal data request

|  |  |
| --- | --- |
| ACTORS | User |
| GOALS | G3 |
| INPUT CONDITIONS | A request to receive data and updates is pending user approval |
| EVENTS FLOW | -User gets notified about the request  -User open the application, goes to the notification area and approve it or refuse it, considering the “-subscribe” option |
| OUTPUT CONDITIONS | The request is approved or denied |
| EXCEPTIONS | No exceptions: in case of connection problems the answer is cached and sent when connection is obtained |

* + - 1. Request to access generalized data

|  |  |
| --- | --- |
| ACTORS | Third party |
| GOALS | G5 |
| INPUT CONDITIONS | Third party is logged in |
| EVENTS FLOW | Third party sends request to the server specifying the number of people () and the type of data he wants to access with “-subscribe” option |
| OUTPUT CONDITIONS | The request is sent |
| EXCEPTIONS | -Malformed request  Exception is handled by notifying the issue to the business |

* + - 1. System permission about the generalized data request

|  |  |
| --- | --- |
| ACTORS | User |
| GOALS | G3 |
| INPUT CONDITIONS | A request to receive generalized data and updates is pending system approval |
| EVENTS FLOW | System automatically approves or refuse request |
| OUTPUT CONDITIONS | The request is approved or denied |
| EXCEPTIONS | No exceptions: in case of connection problems the answer is cached and sent when connection is obtained |

* + - 1. Vital parameters drop below alert threshold and the help-request is triggered

|  |  |
| --- | --- |
| ACTORS | User |
| GOALS | G6 |
| INPUT CONDITIONS | Actual vital parameters fall |
| EVENTS FLOW | -The application become aware of the critical situation  -The help-request is sent to the server, providing location of the patient and which parameters have fallen |
| OUTPUT CONDITIONS | The server processes the request |
| EXCEPTIONS | -The time constraints are not satisfied  -GPS not working  Those exceptions are not in our control, if those services are not available there’s nothing to do about it. |

* + - 1. Ambulance deployed when help-request gets the server

|  |  |
| --- | --- |
| ACTORS | Emergency dept. |
| GOALS | G7 |
| INPUT CONDITIONS | Help-request received from a user |
| EVENTS FLOW | System elaborate the position of the user and send the position of the user along with the status of the patient |
| OUTPUT CONDITIONS | The ambulance gets the user within the time constraints |
| EXCEPTIONS | -High congestion causing delays  Exception 2 can’t be avoided. For further expansion one would consider for example aircraft set. |

3.2.4 [G1]   
Allow an individual to become a Registered User, by agreeing to the TrackMe privacy policy and by providing credentials.

* [R1] A visitor must be able to register him/herself, by starting the registration process.
  + [R1.1] The system will ask to the user all the needed information during the registration process.
  + [R1.2] The system will ask the user to accept TrackMe's privacy policy.
* [R] The e-mail must be unique
* [D] The user must have an email prior to registration
* [D3]: The users' unique form of identification, here considered to be the individual's fiscal code, are actually unique for each person.

3.3.2 [G2]  
Allow a third-party company to become a Registered Third Party by providing credentials.

* [R2] A third party company must be able to register itself, by starting the registration process.
  + [R2.1] The system will ask to the company all the needed information during the registration process.
* [DX] XXXXX

3.2.3 [G3]  
Allow a Registered Third Party to have access to a specific, properly identified Registered User's data if and only if the said User gives its consent to that action.

* [R3] A Registered Third Party must be able to log in to the system using its credentials.
* [R4] A Registered User must be able to log in to the system using his/her credentials.
* [R5] A Registered Third Party must be able to send a request to a Registered User, identified by its unique form of identification
* [R6] A Registered User must be able to view every request sent to him/her by Registered Third Parties.
  + [R6.1] A Registered User must be able to accept or decline the said request.
* [R7] If and only if the Registered User accepted the request from a Registered Third Party, then the same Registered Third Party must be able to access the Registered User's data.

3.2.4 [G4]  
Allow a Registered Third Party to access anonymized data of groups of individuals, if and only if the said group comprehends at least 1000 individuals.

* [R3] A Registered Third Party must be able to log in to the system using its credentials.
* [R4] A Registered User must be able to log in to the system using his/her credentials.
* [R7] A Registered Third Party must be able to send a request to the system for access to anonymized data of groups of individuals.
  + [R7.1] The said groups must be identified by some common characteristics, that can be their age, location, genre, etc.
* [R8] The system must be able to assess if the group identified by the request sent by the Registered Third Party is composed by at least 1000 individuals.
* [R9] If and only if the group identified by the request sent by the Registered Third Party is composed by at least 1000 individuals, then the same Registered Third Party must be able to access the said group's anonymized data.

3.2.5 [G5]

Allow a Registered Third Party to subscribe to new data, in order to receive the latest data as soon it is produced.

* [R3] A Registered Third Party must be able to log in to the system using its credentials.
* [R10] A Registered Third Party must be able to subscribe to the new data.
* [R11] The system must send the new data to every Registered Third Party subscribed to the said service, as soon as the new data is produced.

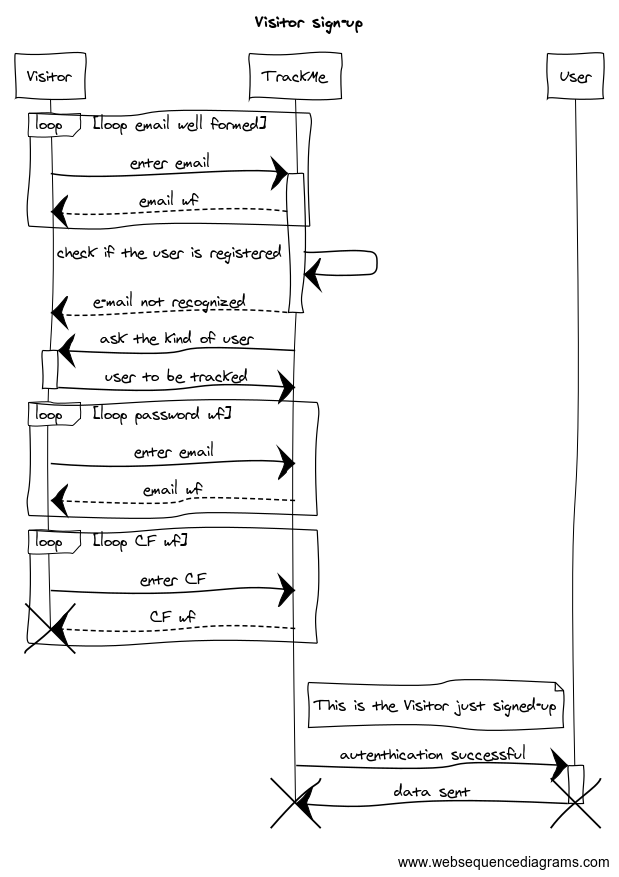
3.2.6 [G6]: Allow a Registered User to become a Registered AutomatedSOS User

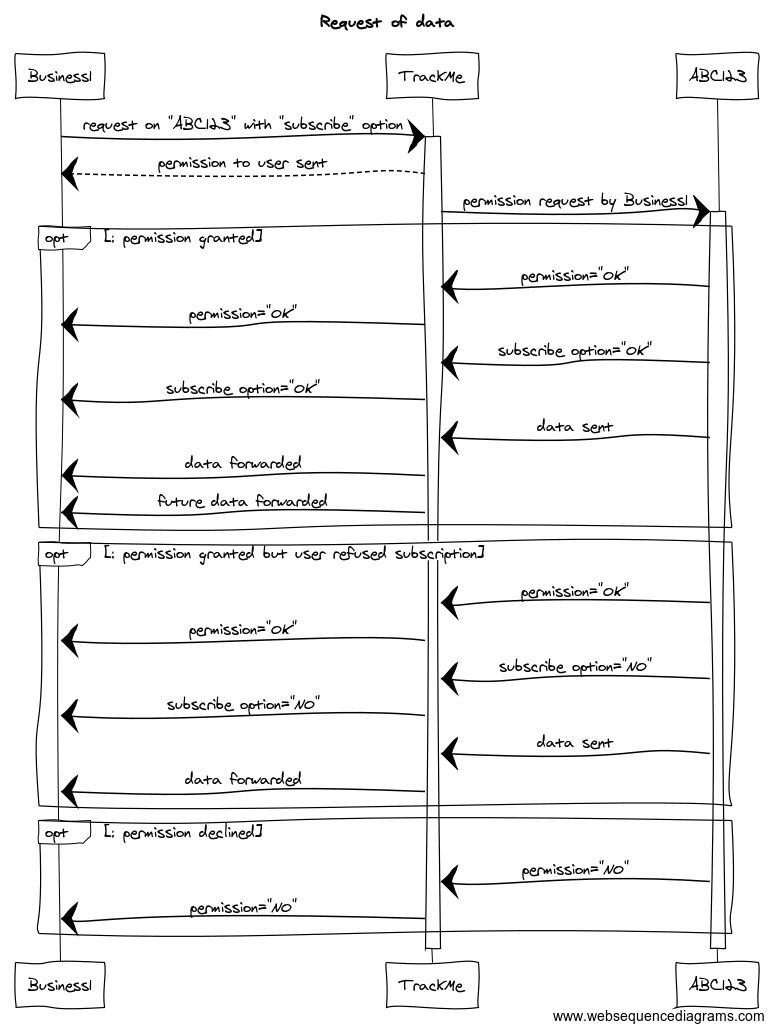
* [RX]: The system must allow a registered user to exploit the AutomatedSOS functionality
* [DX]: The health parameters are not distorted
* [DX]: The user must be logged in to use this function
* [DX]: The position must be trusted

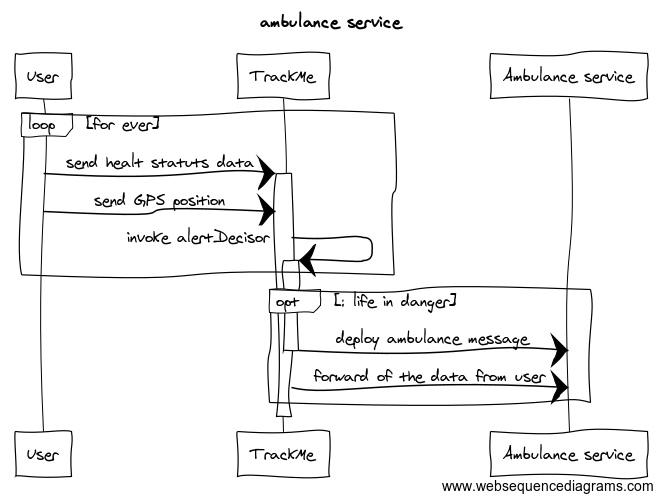
3.2.7 [G7]: Send an ambulance to a Registered AutomatedSOS User's location, if and only if his/hers health parameters drop below a certain threshold.

* [RX]: The system must be connected to the ambulance service
* [RX]: The system must be able to detect a hazardous situation by the health parameters sent
* [DX]: The place where help is needed is reachable by earth-way

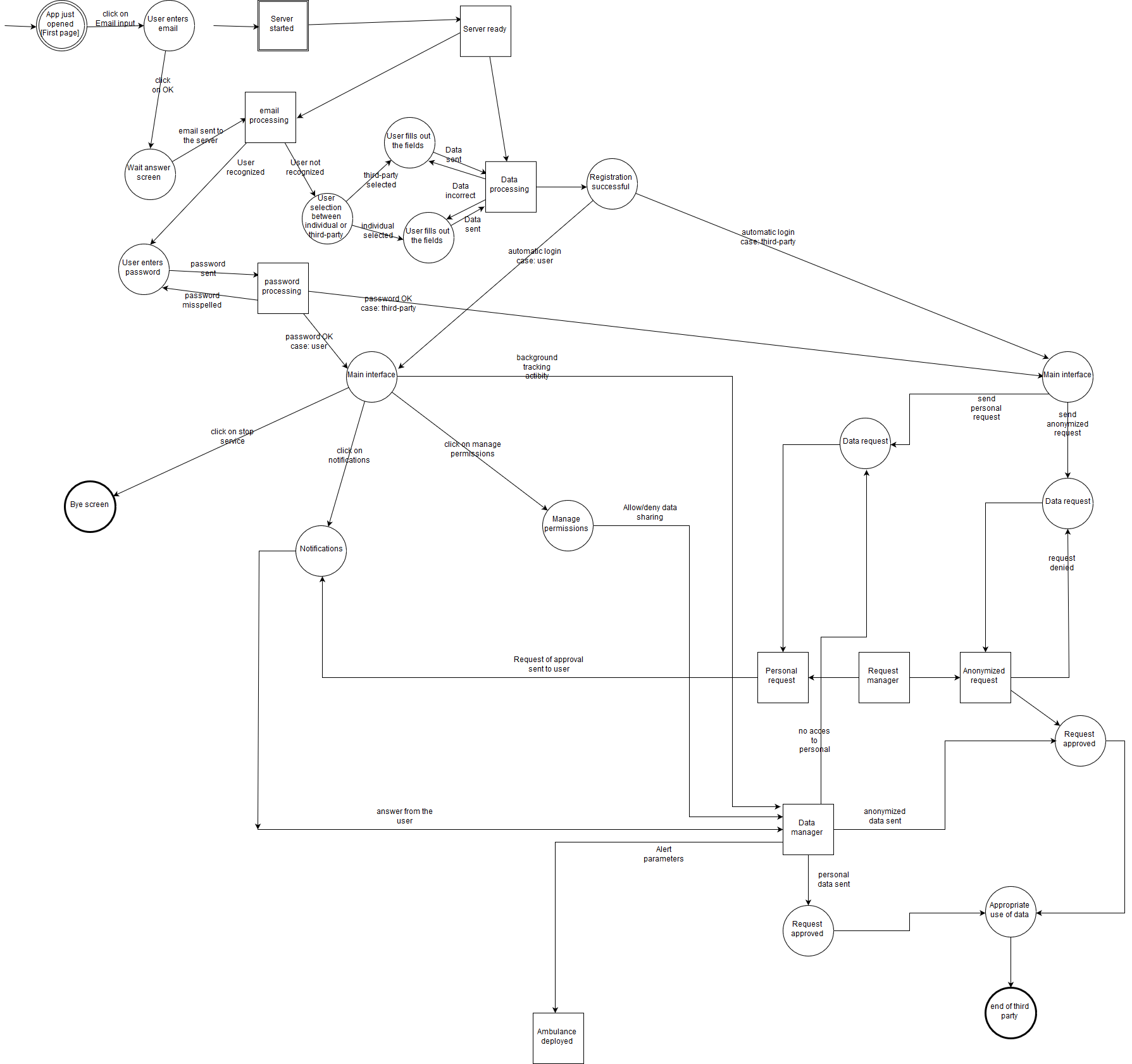
3.2.5 Sequence diagram







3.2.6 full state diagram



3.3 Performance requirements

When it comes to performance requirements, we can speak about two main areas: Computational complexity and the capabilities of the system.

* + - 1. Computational complexity

Find an email will take being the number of user.

The search algorithm will be a binary search and therefore the insert of a new user must be in order.

The insertion time will be again a , adopting a slightly modified version of the binary search with a lexicographic ordering criterion defines a total ordering relation which is defined as follows: we have , and if being the smallest index in which they differ. If instead there not exists a such , which is the case when , then we must have .

The deletion time is in the same complexity of the search time.

Every other action is straightforward (constant time) when a object is found

* + - 1. Capabilities of the system

The system is expected to handle at least 10000 users

3.4. Design constraints

* + - 1. Standard compliance

The strictest request is to send an ambulance within 5 secs. Of course, every user/ third party expects to be notified of the answer of the other in a reasonable time.

* 1. Software system attributes
     + 1. Reliability

The reliability is of the main importance especially when the system detects a harmful situation. The system will do its best to perform at its best.

* + - 1. Availability

As above, the system, the ambulance dept. must be available for all the time the system is needed. The forwarding of the tracked data instead can’t be always available to be always up to date, since it relays on services which can’t be always guaranteed.

* + - 1. Security

The security is one of the main topics, since we are dealing with personal data of the customers. A close sight will be dedicated to this issue in the implementation phase, and countermeasures must be taken in case an intrusion is detected.

* + - 1. Maintainability

For the application to be scalable we must have every system in the network seen as a functional module. In this case maintain and detect bugs should be easier. Very closely related to this is the scalability: in case we see the necessity of an expansion a modular system suits well the replacing of a subsystem or a module itself.

* + - 1. Portability  
         The portability is obtained by using a language cross-platform such as Java. It is easy to install a Java Virtual Machine and then run the program. A future expansion foresees a web based service for the third party, the compatibility problems for it can be then reduced substantially.

**8. Alloy Model**

8.1 Introduction

In this section it's reported the formal model of the system, created and formalised in Alloy.   
Hereby are listed and modelled only some sections of the system itself, in order to better specify some delicate details. Particular focus is on the following points:

* A
* B
* C
* D

8.2 Simplifications

In order to simplify the model, but without losing any meaning, the following simplifications were adopted while developing the model itself:

* The only characteristic reported for the users, other than their location and health parameters, is their age. That said age is considered to be an integer ranging only from 0 to 2. Other factors, like their gender, domicile, etc. were omitted, but their behaviour is exactly the same as their age's.
* The users' position is composed by its geographical coordinates, latitude and longitude; anyway, these details are hereby overlooked, in order to simplify the
* Instead of 1000 individual, the threshold of anonymity is set to 2, in order to simplify the model without depriving it from any meaning. As previously mentioned in the first point, the only parameter considered, as an example, to define groups of users is their age.

8.3 Signatures

8.4 Facts

8.5 Assertions and predicates  
8.6 Results

open util/boolean

open util/integer

sig RegUser{

fiscalCode: one FiscalCode,

age: one Int,

position: one Position,

heartbeat: one Int

}

--boundaries for age and heartbeat

{ age >= 0 and age <= 4 and heartbeat > 0 }

sig RegThirdParty{

vat: one VAT,

requests: set Request,

dataset: one Dataset

}

sig Dataset{

users: set RegUser,

anon: set Position

}

--In order to simplify the model, these entities are not further specified.

sig FiscalCode, VAT, Position{}

sig Ambulance{

patient: one SOSUser,

destination: one Position

}

abstract sig Request{}

sig GroupRequest extends Request{

ageSelector: one Int

}

--boundaries for the age discriminator, used to select individuals

{ ageSelector >= 0 and ageSelector <= 2 }

sig SingleRequest extends Request{

userFC: one FiscalCode,

accepted: one Bool

}

sig SOSUser extends RegUser{

threshold: one Int

}

--

{ threshold > 0 }

lone sig InDanger{

users: set SOSUser

}

--XXXXXXXXX

fun isItAnonymous [a: Int] : set RegUser{

{ u: RegUser | u.age = a}

}

-------------------------------------------------------------------------------------------

--Two diffenrent users cannot have the same fiscal code.

fact fiscalCodeIsUnique{

no disjoint u1, u2: RegUser | u1.fiscalCode = u2.fiscalCode

}

--Each fiscal code must be associated to a user.

fact noLoneFiscalCode{

all f1: FiscalCode | some u1: RegUser | u1.fiscalCode = f1

}

--Two different Registered Third Party cannot have the same VAT.

fact thirdPartyIdIsUnique{

no disjoint t1, t2: RegThirdParty | t1.vat = t2.vat

}

--Each VAT must be associated to a RTP.

fact noLoneVAT{

all v1: VAT | some t1: RegThirdParty | t1.vat = v1

}

--Each position must be associated to a user.

fact noLonePosition{

all p1: Position | some u1: RegUser | u1.position = p1

}

-----------------------------------------------------------------------------------------

--A RTP cannot send more than one request to the same user.

fact oneSingleRequestPerUserPerRTP{

(all t1: RegThirdParty |

no disjoint r1, r2: SingleRequest |

r1 in t1.requests and r2 in t1.requests and r1.userFC = r2.userFC) and

(no disjoint t1, t2: RegThirdParty |

some r: Request |

r in t1.requests and r in t2.requests)

}

--Two different RTP cannot have the same dataset

fact datasetIsUnique {

no disjoint t1, t2: RegThirdParty |

t1.dataset = t2.dataset

}

--Each dataset must be associated to a RTP.

fact noLoneDataset {

all d1: Dataset | some t1: RegThirdParty | t1.dataset = d1

}

fact noLoneRequest {

all r: Request | some t: RegThirdParty | r in t.requests

}

fact noDoubleGroupRequests {

all t: RegThirdParty |

no disjoint gr1, gr2: GroupRequest |

gr1 in t.requests and gr2 in t.requests and

gr1.ageSelector = gr2.ageSelector

}

/\* A user is present in a RTP's dataset if and only if the said RTP has requested that individual

for his/hers data and he/she has given consent \*/

fact inDatasetIffConsenting{

all u: RegUser, t: RegThirdParty |

u in t.dataset.users iff (some sr: SingleRequest |

sr in t.requests and sr.userFC = u.fiscalCode and sr.accepted = True)

}

/\* Anonymized users' data is present in a RTP's dataset if and only if the said RTP has

issued a group request that grants users' anonymity \*/

fact inDatasetIffAnon{

all t: RegThirdParty, u: RegUser |

u.position in t.dataset.anon iff some gr : GroupRequest |

(gr in t.requests and gr.ageSelector = u.age

and (#isItAnonymous[gr.ageSelector] >= 1))

}

---------------------------------------------------------------------------------------

--A SOSUser is in danger if and only if his/hers heartbeat is below the threshold

fact inDangerIffBelowThreshold{

all s: SOSUser |

(s.heartbeat < s.threshold iff s in InDanger.users)

}

--Each SOSUser in danger must have an ambulance sent to his/hers position

fact ambulanceInDanger{

all u: SOSUser |

u in InDanger.users implies some a: Ambulance |

a.patient = u

}

--No ambulance is dispatched without an emergency

fact ambulanceInDanger{

all a: Ambulance |

a.patient in InDanger.users

}

--Only one ambulance per user must be dispatched

fact noDoubledAmbulances{

no disjoint a1, a2: Ambulance |

a1.patient = a2.patient

}

--The destination of the ambulance must be the position oof the user in danger

fact ambulanceCorrectDestination{

all a: Ambulance, u:SOSUser |

a.patient = u implies a.destination = u.position

}

----------------------------------------------------------------------------------------------

/\* If and only if a SOSUser's health parameters drop below his/hers threshold

an ambulance is dispatched. \*/

assert ambulanceAlwaysWhenNeeded{

(all u: SOSUser |

u.heartbeat < u.threshold implies one a: Ambulance |

(a.patient = u and a.destination = u.position)) and

(all a: Ambulance |

a.patient.heartbeat < a.patient.threshold)

}

assert singleRequestAcceptedCase{

all u: RegUser, t: RegThirdParty, sr: SingleRequest |

sr in t.requests and sr.userFC = u.fiscalCode and sr.accepted = True implies

u in t.dataset.users

}

assert singleRequestDeniedCase{

all u: RegUser, t: RegThirdParty, sr: SingleRequest |

sr in t.requests and sr.userFC = u.fiscalCode and sr.accepted = False implies

u not in t.dataset.users

}

--check ambulanceAlwaysWhenNeeded

--check singleRequestAcceptedCase

--check singleRequestDeniedCase

pred show{

#RegUser = 3

#SOSUser = 2

#InDanger.users = 2

}

run show for 8 but exactly 3 SingleRequest, exactly 2 GroupRequest, exactly 2 RegThirdParty

**7. Appendix**7.1 Used tools

* OpenOffice Writer
* Microsoft Word
* StarUML
* Alloy Analyzer 4.2\_2015-02-22
* Sequencediagrams.com
* Creately.com
* Github

7.2 Hours of work

Alberto Tiraboschi

|  |  |  |
| --- | --- | --- |
| **DATE** | **TASK** | **HOURS** |
| 21/10/2018 | Outline of the chapter | 2 |
| 28/10/2018 | Use case diagrams produced, outline of the goals | 6 |
| 30/10/2018 | Chapter 3.2 | 4 |
| 31/10/2018 | Chapter 3.3 | 5 |
| 3/11/2018 | Revision | 2 |
| 5/11/2018 | End of Chapter 3 | 6 |
| 10/11/2018 | Merge of the work done | 5 |

7.2.1 Giorgio Polla

|  |  |  |
| --- | --- | --- |
| **DATE** | **TASK** | **HOURS** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |