



POLITECNICO DI MILANO

MASTER'S DEGREE IN  
COMPUTER SCIENCE AND ENGINEERING

SOFTWARE ENGINEERING 2

---

# TrackMe

## Requirements Analysis and Specification Document

---

*Authors*

Alberto ARCHETTI  
Fabio CARMINATI

*Reference professor*

Elisabetta DI NITTO

v.0 - November 2, 2018

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Purpose . . . . .	1
1.1.1	Project description . . . . .	1
1.1.2	Goals . . . . .	1
1.2	Scope . . . . .	2
1.2.1	World . . . . .	2
1.2.2	Shared phenomena . . . . .	2
1.3	Definitions . . . . .	3
1.4	Acronyms and abbreviations . . . . .	4
1.5	Revision history . . . . .	4
1.6	Document structure . . . . .	4
<b>2</b>	<b>Overall description</b>	<b>6</b>
2.1	Product perspective . . . . .	6
2.2	Product functions . . . . .	6
2.2.1	Profile management . . . . .	7
2.2.2	Data gathering . . . . .	7
2.2.3	Data sharing . . . . .	7
2.2.4	Data management . . . . .	8
2.2.5	Payment handling . . . . .	8
2.2.6	SOS handling . . . . .	9
2.3	User characteristics . . . . .	9
2.4	Assumptions, dependencies, constraints . . . . .	10
2.4.1	Domain assumptions . . . . .	10
2.4.2	Dependencies . . . . .	11
2.4.3	Constraints . . . . .	11
<b>3</b>	<b>Specific requirements</b>	<b>12</b>
3.1	External interface requirements . . . . .	12
3.1.1	User interfaces . . . . .	12
3.1.2	Hardware interfaces . . . . .	17
3.1.3	Software interfaces . . . . .	17
3.1.4	Communication interfaces . . . . .	17
3.2	Functional requirements . . . . .	17
3.3	Scenarios . . . . .	19
3.3.1	Scenario on Stroke detection . . . . .	19

3.3.2	Scenario on Anonymous-Group Request . . . . .	19
3.3.3	Scenario on Single-User Request . . . . .	20
3.3.4	Scenario on subscribing to new data . . . . .	20
3.3.5	Scenario on Graphical Interface . . . . .	21
3.3.6	Anonymous Group Request . . . . .	22
3.4	Use cases . . . . .	22
3.5	Performance requirements . . . . .	22
3.6	Design constraints . . . . .	23
3.6.1	Standards compliance . . . . .	23
3.6.2	Hardware limitations . . . . .	23
3.6.3	Any other constraint . . . . .	25
3.7	Software system attributes . . . . .	25
3.7.1	Reliability . . . . .	25
3.7.2	Availability . . . . .	26
3.7.3	Security . . . . .	27
3.7.4	Maintainability . . . . .	29
3.7.5	Portability . . . . .	30
<b>4</b>	<b>Formal analysis using Alloy</b>	<b>31</b>
<b>5</b>	<b>Graphics</b>	<b>33</b>
<b>6</b>	<b>Effort spent</b>	<b>34</b>

# 1 Introduction

## 1.1 Purpose

### 1.1.1 Project description

TrackMe wants to develop a software-based service that allows individual users to collect health data, called **Data4Help**. This data can be retrieved from the system and visualized according to different filters by a user interface. The system allows third parties registration. Third parties can request access to users' collected data in two ways:

**Single-person data** After the the third party makes a request to the system for a single user data sharing, by providing the fiscal code of the user, the system asks the user for authorization; if positively provided, the third party is granted access to the user's data

**Anonymous-group data** Third parties can be interested in big amounts of data, but not in who are the people providing it; the system, once the request is sent by the third party, checks if the data can be effectively anonymized (it must find at least 1000 people that can provide data matching the third party request) and, if positively evaluated, grants access to the anonymized data to the third party

Third parties can subscribe to new data and receive it as soon as it is collected by the system.

Another service that TrackMe wants to develop is **AutomatedSOS**, built on **Data4Help**. This service analyzes users' data and calls a SOS whenever data exceeds the basic health parameters. For this particular purpose, system performances will be a critical aspect to be taken into account, because even the slightest delay matters in critical health situations.

### 1.1.2 Goals

Here we present the goals that will be reached once the project is completed:

G.U1 Users can collect, store and manage their health data

G.U2 Users can choose to have their health monitored; if their health is critical, an ambulance will be dispatched

- G.T1 Third parties can ask single users for their health data sharing
- G.T2 Third parties can request access to anonymized data that comes from groups of people
- G.T3 Third parties can subscribe to new data and receive it as soon as it is produced

## 1.2 Scope

### 1.2.1 World

Our *world* is composed of two main types of actors: users and third parties. Users are interested in monitoring their health parameters and third parties are interested in developing services or researches that exploit data gathered from the users. **Data4Help** is the service that acts as a bridge between these actors' needs.

Phenomena that occur in the *world* and are related to our application domain are

- physical conditions of the users
- third parties' projects, researches and interests
- SOS system that is able to dispatch ambulances
- payment system that processes payments between actors

These phenomena exist in the *world*, but cannot be observed directly by our system.

### 1.2.2 Shared phenomena

In order to communicate with the *world*, our system needs to share some aspects with it. We will list the aspects controlled by the world, but observable by the machine:

- S.1 physical parameters of the users, gathered through sensors on wearable devices
- S.2 third parties requests to the system for the data they need

S.3 users'location, acquired through GPS signals

On the other hand, the aspects that occur in the machine, but are observable by the world are

S.4 interfaces that organize the gathered data that can be filtered according to time or type of data

S.5 messages for the SOS system, that are sent in case of critical health of a user

S.6 payment requests

### 1.3 Definitions

**Data** Set of values of qualitative or quantitative variables<sup>1</sup>; in the context of **Data4Help** we will refer to quantitative variables concerning health parameters (Section 2.2.2)

**Anonymous data** *data entry* that doesn't contain information about the user from which it was produced; a *data set* is said to be anonymized if it contains only anonymous *data entries* and its cardinality is greater or equal than 1000

**Data entry** Tuple that corresponds to the user's parameters in a particular moment

**Data set** Set of *data entries*; depending on the context, it can identify a set of entries all belonging to a single user or or a set of anonymous entries belonging to more that 1000 users; a *data set*, among all *data* that the system is storing, can be identified and constructed according to the filters of a third party request (Section 2.2.3)

**Request** Third parties can ask the system for some data sharing through requests; requests are encoded through filling a form; the system, provided that the request is satisfiable, grants the third party access to the requested data (Section 2.2.3)

**Third party** Actor interested in collecting data from a single user or from an anonymous group of users (Section 2.3)

---

<sup>1</sup><https://en.wikipedia.org/wiki/Data>

**Threshold** Numerical values related to a particular health parameter; they act as boundaries between the domain of critical health status and normal health status

**User** Actor interested in his/her health data collecting and managing; a user can also be interested in automating SOS calls whenever his health status becomes critical (Section 2.3)

## 1.4 Acronyms and abbreviations

**API** Application Programming Interface

**Data** Whenever the context refers to generic groups of *data entries*, the terms *data* and *data set* are interchangeable

**System** Software product that TrackMe wants to develop; can be interchanged with *S2B*

**S2B** Software To Be

## 1.5 Revision history

Version	Log
v.0	First version of RASD completed

## 1.6 Document structure

This document uses the IEEE standards for requirement analysis documents [2] as a guideline towards a clear and logical explanation of its contents:

- Section 1 gives a brief introduction on the project to be developed, by describing its goals and the environment in which it will be released; keeps track of the revision history
- Section 2 describes the world and the shared phenomena, by defining assumptions and constraints; it identifies the main functions of the project

- Section 3, as the main part of this document, is about requirement analysis; it has also sections about interfaces of the system and software attributes
- Section 4 contains the **Alloy** model that certifies correctness of goals implication by requirements and domain assumptions
- Section 5 contains graphics, like UML diagrams, that help for a better understand of the system structure
- Section 6 lists the work sessions that drove this project's development, ordered by date, as the hour counter of effort spent by each group member



## 2 Overall description

### 2.1 Product perspective

**Data4Help** is a service oriented to data acquisition and data sharing. Its software nature rises the necessity to combine it with another service able to directly retrieve raw data from the *world*. Today we can find for sale multiple wearables that can acquire data from users and make it readable from software side. **Data4Help** users should already own these devices in order to exploit the data acquisition functionality (Section 1.2.2: S.1). Once the user registered to the service, its interface will gather the last data collected and organize it in a chart view, that can be filtered by data or type and rendered by the user interface (Section 1.2.2: S.4; Section 3.1.1).

It is mandatory for the user's wearable to provide a GPS signal, if the user wants to apply to the **AutomatedSOS** service. GPS will be used to track the user's location in case of health danger and the signal will be shared to the SOS service that already exists in the *world* (Section 1.2.2: S.3, S.5). This SOS service accepts messages that contain the GPS location of the person in health danger and an emergency log that **AutomatedSOS** generates from the acquired data. The log explains the suspected health danger and the data that passed the defined thresholds.

Third party organization are interested in data gathering. In order to allow them to make requests for specific types of data, **Data4Help** provides a user interface that is in charge of composing their request, to make it understandable by the software (Section 1.2.2: S.2; Section 3.1.1). The interface provides all the possible options that the third party can compose in order to provide the closest data request to its needs.

### 2.2 Product functions

Here we present the major functions that our product will offer. Some of them will entirely be handled internally by our system, but for others it will rely on external services. In the latter case, we will specify that the system will not directly provide the service and we will add example systems that already exist in the *world*, in order to guarantee the feasibility of the functions.

### 2.2.1 Profile management

The system will provide a registration form at which users and third parties can apply. Once registered, they will have a uniquely identifiable account, provided the requested information for its creation (see Table 1). Once the account has been successfully created, its owner can exploit the system’s functionalities.

Note that accounts for users and third parties must be distinguishable from the system perspective, as it should offer different functionalities to different account types, in order to reflect the account owners’needs.

Account type	Required information	Optional information
User	<i>email, password, fiscal code, date of birth, weight, height</i>	<i>social status, address, hours of work per day, hours of sport per week</i>
Third party	<i>email, password, third party name, third party description</i>	<i>website, research interests</i>

Table 1: Example of login form information

### 2.2.2 Data gathering

Data gathering is exploited through physical wearables that users wear and that can communicate with our software by API calls (ex. Google Fit API [6]). Data entries are identified with a timestamp and the owner user.

Collected data types depend on wearables. For example, the most common sensors for health parameters that we find on smart clothing [5] are pulse, body temperature, electrocardiogram, myocardial and blood oxygen.

### 2.2.3 Data sharing

Data4Help relies on an internal database for data storage. Once data has just been added to the system, only the user that produced it will have access. Data sharing is exploited by granting access to a copy of required data to third parties, if their requests have been accepted. Third parties can retrieve these data by downloading them from their interface.

Third party requests are encoded in the system by filling a form that contains information like Table 2. They can be accepted or rejected. In the former case the third party is granted access to the data, while in the latter it is not granted access.

Request type	Accept condition	Filters
Single-user	Target user should accept the request through his/her account	<i>fiscal code of target user, from-date, to-date, data types (weight, heart rate, etc.), time granularity (seconds, minutes, hours)</i>
Anonymous-group	Every user should have accepted the automatic sharing of requested data	<i>size, from-date, to-date, data types (weight, heart rate, etc.), user characteristics (age interval, weight interval, etc.)</i>

Table 2: Example of third party request form

#### 2.2.4 Data management

Once a user collected some data, he/she can organize it by changing the view options in the user interface. These options depend on the device the user is working on, but will always provide basic filters such as time interval or data type. Once filters have been selected by the user, the graphical interface will render a chart that organizes collected data according to them (see Section 3.1.1).

#### 2.2.5 Payment handling

When third parties initialize requests, they are asked for payment by the system. Payment details are defined by TrackMe policy, so they won't be discussed in this document.

Our system will only initialize payment requests as the effective payment operation will entirely be handled by an external software. This software should

- instantiate payment processes
- check if the payment is feasible and, if not, notify to our system
- handle the payment operation
- notify to our system if the operation has concluded correctly, otherwise notify the error occurred

There are plenty of payment handlers that exist in our *world* and can be paired to our system (ex. PayPal API [7]).

### 2.2.6 SOS handling

This function is heavily dependant on the country in which the SOS will be handled. Our system will only communicate to the external emergency service that already exists in the *world* through automatic API calls (ex. RapidSOS Emergency system for US [8]).

Calls to SOS are handled by **AutomatedSOS** that signals users GPS position and health status feedback. Calls occur, if the user previously subscribed to **AutomatedSOS**, when his/her health parameters are below certain thresholds. The time between health danger detection and emergency call must be less than 5 seconds.

## 2.3 User characteristics

The following list contains the actors that are involved in the system functions:

**User** Person interested in the **Data4Help data management** service; he/she is required to register to the **Data4Help** service in order to exploit its functionalities; he/she can also apply to the **AutomatedSOS** service; every user owns an appropriate device for **Data4Help** acquisition service that can monitor at least GPS location  
*ex. an athlete that wants to monitor his/her physical status during sport activity, an old person that suffers from heart diseases or a sedentary worker that wants to keep track of his health parameters in the working hours*

**Third Party** Entity interested in the **Data4Help data sharing** service; it is required to register to the **Data4Help** service in order to exploit its functionalities; according to the scope of its requests, it may be interested on the data of a particular user or in aggregated chunks of data

ex. *the physician of a person that suffers from heart diseases or a statistical institute*

## 2.4 Assumptions, dependencies, constraints

### 2.4.1 Domain assumptions

#### World

D.W1 Signals processed by wearables are encoded correctly and represent the status of the *world*

D.W2 Given certain health parameters<sup>2</sup>, it is possible to decide if a person is in health danger just by checking whether the parameters are above or below certain thresholds

#### Existing systems

D.E1 In the *world* already exists a SOS system that is able to dispatch ambulances and accepts emergency calls through an API

D.E2 In the *world* already exists a payment handler that is able to deliver money payments and accepts calls through an API

D.E3 In the *world* already exist wearables that encode signals for health status; these encoded signals are accessible from the software side

#### Legal constraints

D.L1 Acquired data can be sold to third parties

---

<sup>2</sup>Section 2.2.2 discusses which parameters are current wearables able to detect; we won't discuss regarding which parameters should be taken into account to determine the health status of an individual and which thresholds should be defined, as it is a topic strictly related to medical research

### 2.4.2 Dependencies

Data4Help relies on

- **payment handler**, to deal with payment of users and third parties, as discussed in Section 2.2.5
- **wearables**, to encode users' data (see Section 2.2.2)

AutomatedSOS relies on

- **wearables**, to encode GPS signals and users' parameters
- **SOS system**, for ambulance dispatching (see Section 2.2.6)

### 2.4.3 Constraints

Data4Help acquires data through external devices owned by other companies, so it must be legally authorized to sell the data it acquires. In this document we assume that there are no legal issues for the selling activity, as TrackMe will develop contracts with the wearables companies in order to solve this issue.

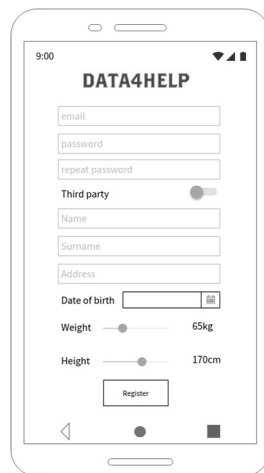
### 3 Specific requirements

#### 3.1 External interface requirements

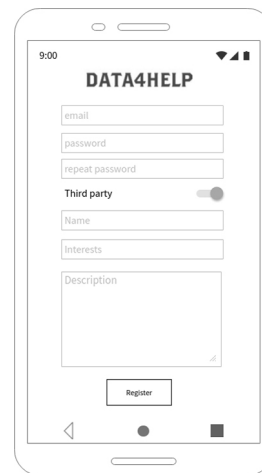
##### 3.1.1 User interfaces



Figure 1: Homepage

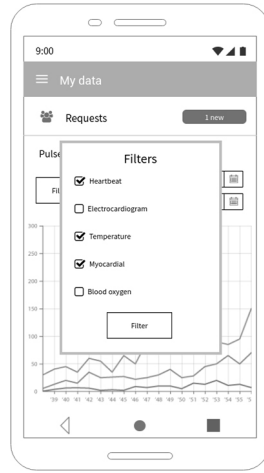


(a) User Registration

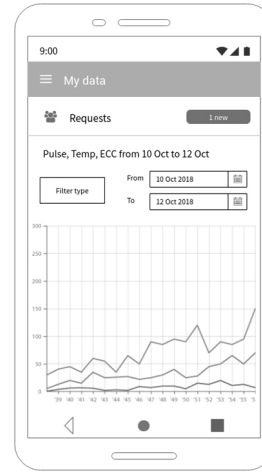


(b) Third Party Registration

Figure 2: Registration Form

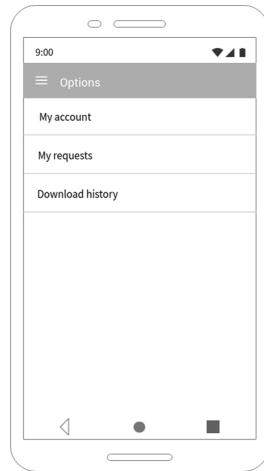


(a) User Filters

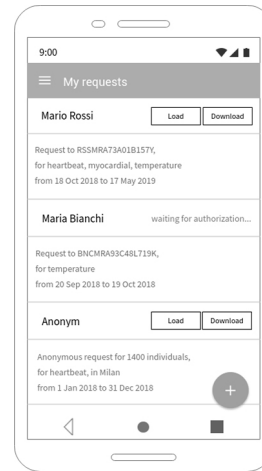


(b) User Data

Figure 3: Interface for Graphical representation of User Data



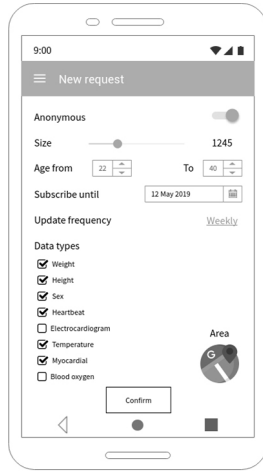
(a) Third Party Op-  
tions



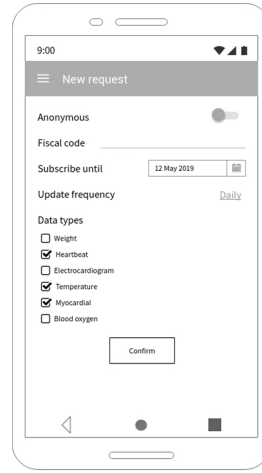
(b) Third Party Re-  
quests

Figure 4: Third Party management of Account



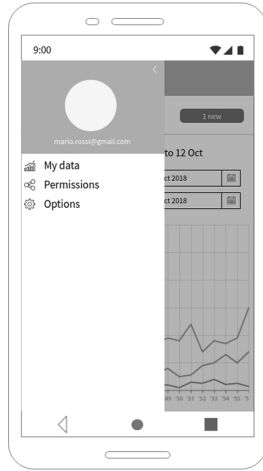


(a) Anonymous-Group Request

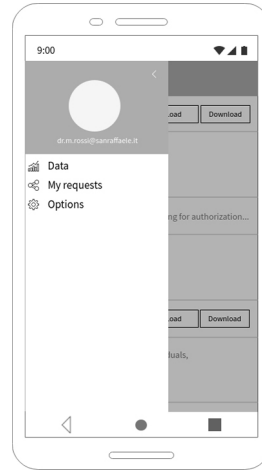


(b) Single-User Request

Figure 5: Third Party management of Account



(a) User Sidebar



(b) Third Party Sidebar

Figure 6: Sidebar

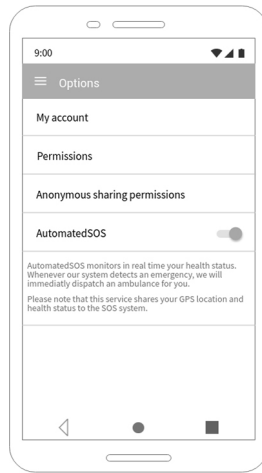
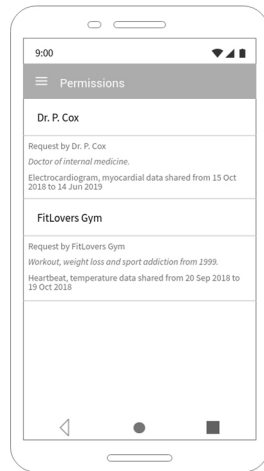
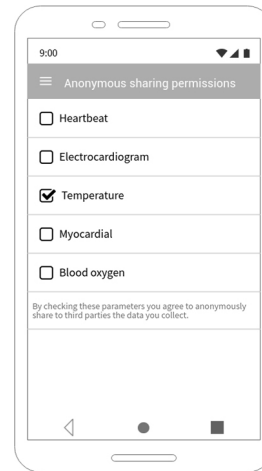


Figure 7: User Options

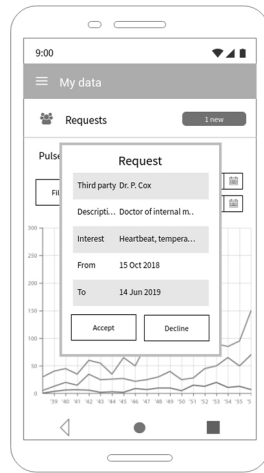


(a) List of all Third Party that has access to User Data

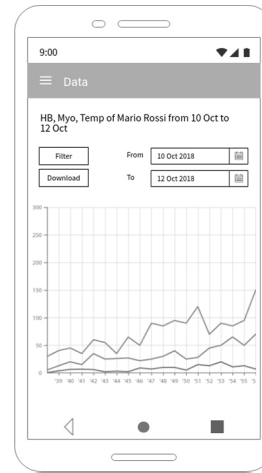


(b) User Data that can be shared with Third Parties

Figure 8: Anonymous User Data Sharing Management



(a) User request to share single Data



(b) Single-User Data shared

Figure 9: Single-User Data Management

### 3.1.2 Hardware interfaces

### 3.1.3 Software interfaces

### 3.1.4 Communication interfaces

## 3.2 Functional requirements

### Account handling

- R.A1 The system shall allow users registration
- R.A2 The system shall allow third party registration
- R.A3 The system shall distinguish between user and third party accounts
- R.A4 The system shall guardantee account uniqueness by not allowing two account to have the same email
- R.A6 The system shall allow users and third parties to access their account (*login*) only if they provide correct email and password
- R.A7 The system shall allow users and third parties to exploit **Data4Help** and **AutomatedSOS** functionalities only if they are *logged in* their account

### Data encoding

- R.D1 The system shall encode data received through wearables'sensors and store it internally<sup>3</sup>
- R.D2 The system shall be able to retrive data previously stored on request
- R.D3 The system shall not erase data once it is stored internally
- R.D4 Once the system stored data, it shall allow access to that data only to the user account that was *logged in* while the data was collected
- R.D5 The system shall be able to share the stored data to more than one account

---

<sup>3</sup>for example the system can store data into an internal database or can save it using a cloud service; the important aspect is that data must be retrivable in the future

- R.D6 The system shall be able to compose groups of data entries (*aggregate data*) and anonymize them (every data entry in the group has no information about the providing user, such as fiscal code or email)

## **Interfaces**

- R.I1 The system shall provide a registration form for users and third parties
- R.I2 The system shall provide to the user an interface able to render data graphically, allowing filters like time interval or data type
- R.I3 The system shall provide a request form to the third parties

## **Data sharing requests**

- R.R1 The system shall allow third parties to ask for data sharing of a single user
- R.R2 The system shall ask the user to accept or decline every single-user request of data sharing by third parties that has him/her as target user
- R.R3 The system shall provide access to the target user data to the third party only if the user accepted the request of the third party, otherwise the system shall notify to the third party that the user declined the request
- R.R4 The system shall allow third parties to ask for data sharing of *aggregate data*
- R.R5 The system shall be able to check if a request for *aggregate data* by a third party can be properly anonymized (there shall be at least 1000 user data entries that fit the parameters of the request)
- R.R6 The system shall provide access to the *aggregate data* to the third party only if the the request can be properly anonymized, otherwise the system shall notify the third party that its request cannot be properly anonymized
- R.R7 The system shall provide access to a third party to newly produced data if it fits the third party request

## **SOS calls**

- R.S1 The system shall provide to the user an option to apply to **AutomatedSOS**
- R.S2 If user applied to **AutomatedSOS**, the system shall monitor his/her parameters in real time by checking whether they are above or below the thresholds
- R.S3 If user applied to **AutomatedSOS** and his/her health parameters are critical (above or below thresholds), the system shall send an emergency call to the SOS system
- R.S4 When the system sends an emergency call, it shall provide to the SOS system API user's GPS location and user's health status through his/her critical parameters encoding

## **3.3 Scenarios**

### **3.3.1 Scenario on Stroke detection**

Luke is a 65 years old man that after 40 years of work at the post office finally got retired. Because of the sedentary nature of his work he is worried that he may suffer a stroke.

In order to monitor his health parameters, he downloads the **Data4Help** app. The system allows him to create an account after filling all the required information(see Table 1). The system explicitly asks Luke if he wants to join the **AutomatedSOS** service and Luke accepts sharing his heartbeat and myocardial with the app.

The system will now on check in real time Luke's health parameters that will be collected through his smartwatch.

After some days Luke's health parameters exceed thresholds because of a stroke: the system recognizes immediately the critical situation and forwards in less than 5 seconds an emergency notification to SOS system, providing the Luke's GPS location and health status feedback.

### **3.3.2 Scenario on Anonymous-Group Request**

The franchise BodySlim is opening a new gym in Milan near Parco Sempione;the most relevant aspect that characterize it, is the timetable:open 365 days, 8 hours per day.

In order to maximize the revenue the management would like to know in which time of the day the potential customers make exercise. A reliable sample can be obtained from Data4Help, therefore BodySlim create a third party account filling all the required third party information (see Table 1).

Then the franchise make an anonymous-group request (see Table 2) for 2000 individuals with particular characteristics: age between 20 and 60, at least once a week they are running in Parco Sempione (this can be checked with GPS position) to Data4Help.

The app find in his internal database these 2000 users and therefore the system immediately shares the required information with BodySlim without asking to the users the authorization .

Now looking at the most common hours chosen for running by these individuals the manager can easily identify the most profitable hours in which the gym should be opened.

### **3.3.3 Scenario on Single-User Request**

Rose has just discovered that is expecting a daughter. Harold, her gynecologist, needs to keep monitoring bloody oxygen, electrocardiogram and pulse in order to understand if the pregnancy is proceeding well or not.

Therefore he creates a third party account of Data4Help and after the system has checked if the required information (see Table 1) has been correctly inserted, he requires these data to the app.

According to the single-user request (see Table 2) policy Data4Help asks to Rose the consensus for sharing these data. Rose, which really care about her daughter's health, immediately gave the authorization to the app that can make them available to her gynecologist.

Nine months later the beautiful and healthy Marie is born.

### **3.3.4 Scenario on subscribing to new data**

Jack is a policeman that has decided to spend his holidays in a spa called "BeautySPA" which allow him to loss 15 kg thanks to fitness activity and a healthy diet. In order to understand if this treatment is efficient over time the spa and Jack decides that user's health parameters like height, weight and body temperature must be monitored.

Therefore the policeman and BeautySPA create respectively a user and a third party account of Data4Help and the system ,after checking that all the required information (see Table 1)are properly added, allow them to exploit all Data4Help functionalities.

Indeed BeautySPA immediately apply for the three Jack’s health parameters to the app.

The app recognize this as a single-user request(see Table 2) so asks to Jack the authorization whether to provide them to the spa or not.

Obviously he accept and therefore Data4Help make available these data.

Unfortunately after just 1 month the policeman has already had a weight increase of 6 kg. So in order to better understand which may be the reasons the spa asks more health parameters such as blood oxygen and heartbeat to Data4Help which again,before sharing these data with BeautySPA, will ask the authorization of the user.

### **3.3.5 Scenario on Graphical Interface**

Paul is studying Telecommunication Engineering at Politecnico di Milano and would like to participate at the PolimiRun therefore he is training twice a week since January(the PolimiRun occurs in May).

The telecommunication student decides to create a User Account sharing heartbeat and bloody oxygen with Data4Help.

The system checks that all the required information are properly added then it starts to periodically collect the two user data entries creating a Dataset for the student.

At the end of April Paul is worried that despite all the efforts the PoliRun is beyond his physical capacities so he resolves to unsubscribe to that competition but then remember that Data4Help lets the users to see in a personalized way all the data collected provided that they are logged in their account.

Consequently Paul,after the login in, asked to the app to show his bloody oxygen and heartbeat from January to April.

The system retrieves the required data and make them available to the student through the graphical interface of the app.

In this way Paul can see the evolution of these two health parameters during the four months and immediately understands that the PolimiRun is absolutely realistic for him thanks to an impressive improvement in the concentration of bloody oxygen and a reduction of heartbeat per minute.



### 3.3.6 Anonymous Group Request

FastAndEasy, a car-sharing company, has a business model based on how much time his users rent one of his car in a week. After 5 years the society has to decide whether change this model with a new one based on the kilometers travelled in a week or keep going with the old one.

In order to take the right decision FastAndEasy needs to know the GPS location of all of his users when they are driving his cars(so not 24 hours a day) for 3 months.

Because of this is a temporary solution it will be a waste of money for the car-sharing company to implement his own system for picking up these data. Therefore FastAndEasy creates a third party account of Data4Help filling all the required third party information (see Table 1) and asks to the app to provide the GPS location only of his users(which can be identified by the fiscal code).

This is a Single-user(Table 2) request therefore Data4Help will ask directly to the specified users whether accept or deny it.

Indeed only for the users who accept the request the app will grant the access to their GPS location to the company which, looking at the data collected after 3 months, can compute the possible earnings of the new model,compare them with the old ones and finally make a decision.

## 3.4 Use cases

## 3.5 Performance requirements

**AutomatedSOS** implements a very important function:monitoring the human health; in this field having a lower time response can save a life.

Therefore, when the system detects than at least one of the health parameters exceeds the thresholds (user is in health danger),it must collect all the required data(GPS location and user's health status) and send them through an emergency call to an external service(SOS System) within 5 seconds.

Moreover Data4Help has to send every 1 second the required data to the subscribed third parties(both for Single-User requests and for Anonymous-Group requests) in order to grant them accurate enough data.

**AutomatedSOS** time performances are critical: the emergency call must be delivered no more than 5 seconds after the health danger detection, in order to have an ambulance dispatched from the SOS system as soon as possible.

<b>Name</b>	Sign in of User in Data4Help
<b>Actors</b>	User
<b>Entry Condition</b>	Data4Help is downloaded on the smartphone
<b>Flow of Events</b>	1.User click on "Create a new User account" button in the homepage of the app. 2.User fills all the mandatory fields("Age", "Name", "Surname", "Email", "Sex", "Password", "Fiscal Code"). 3.User insert in the fields "Shared Data" the data that he want to share with the app. 4.User clicks on the "Confirm" button. 5.The system create a DataSet for the User in his internal database.
<b>Exit Condition</b>	The User account has been successfully created.
<b>Exceptions</b>	1.The system finds that User filled the field "Email" with an email that is non-existent or that has already been associated with another user. 2.The system finds that User filled the field "Fiscal Code" with an expression not long 16 characters. 3.The system finds that User filled the field "Password" with an expression shorter than 8 characters. All exceptions are handled notifying the issue to the User and taking back the <b>Flow of Events</b> to the point 2.

It is important to note again that **AutomatedSOS** is in charge only of the emergency call.

## 3.6 Design constraints

### 3.6.1 Standards compliance

### 3.6.2 Hardware limitations

The smartphone, in which the app(both for User account and Third Party Account) is downloaded, must guarantee that all the system functionalities can be correctly performed; hence the only compatible smartphones are the ones with : iOS or Android as operating system,3G/4G connection and can

<b>Name</b>	Log in of User
<b>Actors</b>	User
<b>Entry Condition</b>	User has already created an account
<b>Flow of Events</b>	1.User opens the app on his smartphone. 2.User insert his email in the field "Email" in the homepage of Data4Help. 3.User insert his password in the field "Password" in the homepage of Data4Help. 4.User Click on "Log In" button in the homepage of the app.
<b>Exit Condition</b>	User is successfully Logged in and can exploit all the functionalities of the app.
<b>Exceptions</b>	1.The system doesn't find the email address inserted in the field "Email" in his internal DataBase. In this case the <b>Flow of Events</b> has to be executed again from step 2. 2.The system doesn't find the password inserted in the field "Password" in his internal DataBase. In this case the <b>Flow of Events</b> has to be executed again from step 3. 3.The system find both the email and the password inserted by User but these refers to 2 different users. In this case the <b>Flow of Events</b> has to be executed again from step 2.

retrieve GPS location.

Moreover is necessary that each user has his own physical device and this device must be able to collect all the data that the user decides to share with the app otherwise the system won't be able to perform the Data Gathering function(2.2.2).

<b>Name</b>	Accept Single-User request
<b>Actors</b>	Third party and User
<b>Entry Condition</b>	Third party and User have already created respectively a third party and a user account and they are already logged in in Data4Help.
<b>Flow of Events</b>	1.Third party clicks on the button "Single-User Request". 2.Third party inserts the Fiscal Code of the User in the field "Fiscal Code" of the Single-User Request page. 3.Third party lists to the system the data of the User which it would like to access filling the field "DataRequired" in the Single-User Request page. 4.The system asks to the User whether he accepts or denies to share those specified data with the Third party. 5.User answers clicking in the field "Yes" in his home-page. 6.The system make available the data for the third party.
<b>Exit Condition</b>	Third party can exploit the data
<b>Exceptions</b>	1.The system doesn't find the fiscal code inserted by third party in the field "Field Code" in his internal database. In this case the <b>Flow of Events</b> has to be executed again from step 2. 2.The system discovers that the User has denied the request therefore Data4Help notifies this information to the third party which won't have access to the required data. The third party can make a new Single-User request referred to a new User: in this case the <b>Flow of Events</b> has to be executed again from the first step.

### 3.6.3 Any other constraint

## 3.7 Software system attributes

### 3.7.1 Reliability

The software is expected to work continuously for 24 hours a day despite small deviations are accepted.

<b>Name</b>	Call an ambulance
<b>Actors</b>	User and SOS System
<b>Entry Condition</b>	User has actived AutomatedSOS and his health parameters exceeds thresholds
<b>Flow of Events</b>	1.The system collects GPS location and health status parameters of the User from his internal database. 2.The system makes an emergency call to SOS System by providing GPS location and health status feedback of the User. 3.The SOS System accepts the call and dispatches the ambulance to the User location. 4.The SOS System notifies the system when the ambulance arrives at User Location.
<b>Exit Condition</b>	The ambulance is taking care of the User.
<b>Exceptions</b>	
<b>Special Requirements</b>	1.The system has to collect the GPS location,health parameters of the User and make the emergency call to SOS System within five seconds from the time the health parameters exceeds thresholds.

Moreover as section 3.6.4 says, the software changes during time, so each time the software is modified is necessary to check the consequences on the reliability.

### 3.7.2 Availability

The system is expected to be available for 99.99 % of the time but in case of failure it must be able to recover within 1 second otherwise in case of user's health problem AutomatedSOS won't respect the constraint descibed in 3.5 and therefore the user will be in a critical situation.

There 2 different situations which may result in a failure:failed components or due to unexepcted increasing in the requests.

At this level of abstraction is not strickly required but in both of these cases a distributed architecture is an optimal solution:indeed in case of failure one of the redundant component can replace the one offline and in case of

<b>Name</b>	Accept Anonymous-Group request
<b>Actors</b>	Third Party
<b>Entry Condition</b>	Third party has already created a third party account and is already logged in in Data4Help.
<b>Flow of Events</b>	1.Third party clicks on the button "Anonymous-Group Request". 2.Third party lists to the system which data requires filling the field "Data Required" in the Anonymous-Group request page. 3.The system looks for, in his internal database, the User's Datasets which fit with the request made by the third party. 4.The system aggregate the data found and anonymize them. 5.The system provides the anonymous data to the third party.
<b>Exit Condition</b>	Third party can exploit the data
<b>Exceptions</b>	1.The data cannot be anonymized by the system because it doesn't find at least 1000 User Datasets which match with the third party request. Therefore the system cannot grant access to the data to the third party and has to notify her this information . The third party can make a new Anonymous-Group request asking for a new set of data: in this case the <b>Flow of Events</b> has to be executed again from the first step.

high demand a load balancing between redundant component can minimize the probability of failure of the system.

### 3.7.3 Security

The system requires the users(both Data4Help and AutomatedSOS) and third Parties(only in Data4Help), in order to exploit all the functionalities of the app and avoid unauthorized access, to authenticate through email and password.

These 2 credentials and all the Data collected by the system has to be

<b>Name</b>	Show Data previously acquired
<b>Actors</b>	User
<b>Entry Condition</b>	User has already created a User account and is already logged in in Data4Help.
<b>Flow of Events</b>	1.User clicks on the button "Data Management". 2.User decides which filter has to be applied to his DataSet filling the field "Filters" in the Data Management page. 3.The system organize the User's DataSet following the User's instructions. 4.The system show to the User these data thanks to a graphical interface.
<b>Exit Condition</b>	User can see his data in the specified way
<b>Exceptions</b>	1 The system hasn't gathered any data yet because the user has just created an account. Thanks to the graphical interface the system notifies the user that no data can be shown. Only after having collected some data the User can specified filters:the <b>Flow of Events</b> must be executed from the first step.

Use Case ID	Name
1	Sign in of User in Data4Help
2	Log in of User
3	Accept Single-User request
4	Call an Ambulance
5	Accept Anonymous-Group request
6	Show Data Previously Acquired

Table 3: Traceability Matrix

securely stored in the internal database.

Moreover as requirement **R.D6** says for privacy issues for all the Anonymous-Group requests the system must not allow the third parties to link data collected to the users that produced them.

Raw ID	Goal ID	Req ID	Use Case ID
r1	G.U.1	R.A1 R.A3 R.A4 R.I.2	1
r2	G.U.1	R.A3 R.A6 R.A7	2
r3	G.T.1	R.A3 R.I.3 R.R.1 R.R.2 R.R.3	3
r4	G.U2	R.S1 R.S2 R.S3 R.S4	4
r5	G.T2	R.A3 R.D2 R.D5 R.D6 R.I.3 R.R4 R.R5 R.R6	5
r6	G.U1	R.A3 R.A7 R.I2 R.D2	6

### 3.7.4 Maintainability

Data4Help, as 2.2.2 says, has to collect data through multiple physical wearables devices therefore Data type handling should be implemented by the system in a flexible and extensible way, because wearables technologies change rapidly and our software should adapt in the least invasive way possible.



### **3.7.5 Portability**

The system has to gather user's data through multiple devices as 2.2.2 says. Therefore, must be guarantee that different physical wearables can easily send data to this product function for example thanks to some form of abstraction.

In this way I will also have a significant cost reduction and an improvement in maintainability of the system.

## 4 Formal analysis using Alloy

```
1 open util/integer
2 open util/time
3 open util/boolean
4
5 sig User {
6     myData : some DataEntry
7 }
8
9 sig DataType {
10     threshold : one Int
11 }
12
13 sig DataEntry {
14     timestamp : one Int,
15     type : one DataType,
16     value : one Int,
17     location : one GPSLocation
18 } {
19     timestamp > 0
20 }
21
22 sig GPSLocation {}
23
24 fact UniqueDataEntries {
25     all u : User | all disj de, de' : u.myData |
26         ↪ de.timestamp ≠ de'.timestamp
27 }
28
29 fact NoDataEntryWithoutUser {
30     all de : DataEntry | some u : User | de in u
31         ↪ .myData
32 }
33
34 pred addDataEntry(u, u' : User, de : DataEntry) {
35     u'.myData = u.myData + de
36 }
37
38 pred show{}
39
40 run addDataEntry for 3 but exactly 1 User, exactly 3
41     ↪ DataEntry
42
43 //run show for 2
```

```
41 //run show for 4 but exactly 4 DataEntry, exactly 2  
    ↪ User
```

## 5 Graphics

## 6 Effort spent

## References

- [1] Mandatory Project Assignment AY 2018-2019
- [2] IEEE 830-1993 - IEEE Recommended Practice for Software Requirements Specifications
- [3] ISO/IEC/IEEE 29148 - Systems and software engineering — Life cycle processes — Requirements engineering
- [4] Collection and Processing of Data from Wrist Wearable Devices in Heterogeneous and Multiple-User Scenarios  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038811/>
- [5] Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices  
<https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5334130/>
- [6] Google Fit API  
<https://developers.google.com/fit/overview>
- [7] PayPal API  
<https://developer.paypal.com/docs/>
- [8] RapidSOS Emergency API  
<https://info.rapidsos.com/blog/product-spotlight-rapidsos-emergency-api>
- [9] Slides of the course by Prof. Di Nitto  
<https://beep.metid.polimi.it/>
- [10] L<sup>A</sup>T<sub>E</sub>X templates  
<http://www.latextemplates.com/>
- [11] Alloy L<sup>A</sup>T<sub>E</sub>X Highlighting  
<https://github.com/Angtrim/alloy-latex-highlighting>
- [12] Draw.io  
<https://www.draw.io/>