

**TrackMe project - Argiro' Anna Sofia,  
Battaglia Gabriele, Bernardo Casasole**



**POLITECNICO**  
MILANO 1863

# **Requirement Analysis and Specification Document**

---

<b>Deliverable:</b>	RASD
<b>Title:</b>	Requirement Analysis and Verification Document
<b>Authors:</b>	Argiro' Anna Sofia, Battaglia Gabriele, Bernardo Casasole
<b>Version:</b>	1.2
<b>Date:</b>	December 14, 2018
<b>Download page:</b>	<a href="https://github.com/BernardoCasasole/ArgiroBattagliaCasasole.git">https://github.com/BernardoCasasole/ArgiroBattagliaCasasole.git</a>

---

# Contents

Table of Contents	3
<b>1 Introduction</b>	<b>5</b>
1.1 Purpose	5
1.1.1 Goals	5
1.2 Scope	5
1.3 Definitions	6
1.4 Acronyms	7
1.5 Abbreviations	7
1.6 Revision history	8
1.7 Document Structure	8
<b>2 Overview</b>	<b>9</b>
2.1 Product perspective	9
2.2 Product functions	11
2.2.1 Data management	11
2.2.2 Data access	11
2.2.3 Data subscription	11
2.2.4 Emergency call	11
2.2.5 Running competition organization	11
2.2.6 Running competition monitoring	12
2.2.7 Running competition enrolling	12
2.3 User characteristics	12
2.4 Assumptions	12
2.5 Constraints	13
2.6 Dependencies	13
<b>3 Specific Requirements</b>	<b>14</b>
3.1 External Interface Requirements	14
3.1.1 Individual User Interface	14
3.1.2 Third-party User Interface	20
3.1.3 Hardware interfaces	22
3.1.4 Software interfaces	22
3.1.5 Communication interfaces	22
3.2 Functional Requirements	23
3.2.1 Scenarios	23
3.2.2 Use cases	25
3.2.3 Sequence diagram	44
3.2.4 Requirements mapping	50
3.3 Performance Requirements	51
3.4 Design Constraints	51

3.4.1	Standards Compliance . . . . .	51
3.5	Software System Attributes . . . . .	52
3.5.1	Reliability . . . . .	52
3.5.2	Availability . . . . .	52
3.5.3	Security . . . . .	52
3.5.4	Maintainability . . . . .	53
3.5.5	Portability . . . . .	53
<b>4</b>	<b>Formal Analysis Using Alloy</b>	<b>54</b>
4.1	Data4Help . . . . .	54
4.1.1	Alloy model . . . . .	54
4.1.2	Generated world . . . . .	59
4.1.3	Alloy Result . . . . .	60
4.2	AutomatedSOS . . . . .	61
4.2.1	Alloy model . . . . .	61
4.2.2	Generated world . . . . .	64
4.2.3	Alloy Result . . . . .	65
4.3	Track4Run . . . . .	66
4.3.1	Generated world . . . . .	70
4.3.2	Alloy Result . . . . .	71
<b>5</b>	<b>Effort Spent</b>	<b>72</b>
5.1	ARGIRO' ANNA SOFIA . . . . .	72
5.2	BATTAGLIA GABRIELE . . . . .	73
5.3	CASASOLE BERNARDO . . . . .	74
<b>6</b>	<b>References</b>	<b>75</b>
6.1	Reference Documents . . . . .	75
6.2	Software . . . . .	77

# 1. Introduction

## 1.1 Purpose

This document, representing the Requirement Analysis and Specification Document, is addressed to every person and entity interested in the project.

The aim of this document is both to picture the world the product would live in, including high level information that could be useful for stakeholders, and to provide specific information for developers or any technical staff interested in implementing the product. It was also prominent concern to give a correspondence between client desires (and necessities) and the editing of functional and non-functional requirements.

In this document it has been took particularly regard toward the comprehensibility and accuracy in order to be accessible to every individual that could have a perspective for the project.

### 1.1.1 Goals

- G1** Allow users to properly use the services they wish to employ
- G2** Allow visitors to register as individual or third-party user
- G3** Allow individual users to monitor their location and health parameters
- G4** Allow third-party users to request the data on specific users
- G5** Allow individual users to approve or deny the specific request for their data
- G6** Allow third-party users to request data on anonymized groups of individual users
- G7** Call an ambulance if the system detects a critical health condition
- G8** Allow third-party users to organize running competitions
- G9** Allow Individual users to enrol in existing running competitions as participants
- G10** Allow Individual users to subscribe in existing running competition as spectators to monitor underway competitions

## 1.2 Scope

The Data4Help System includes the three services that TrackMe wants to develop: Data4Help, which is the main service, Track4Run and AutomatedSOS. The Data4Help Mobile Application and the Data4Help Web Page allow users to make use of the three Data4Help System services. To be specific, IU are allowed to activate or deactivate Track4Run and AutomatedSOS services whenever they want to.

The service Data4Help is designed to monitor the location and health status of individuals registered to it using wearable devices. Upon registration individual users agree to the acquisition and usage of data by Data4Help and will be able to see their own data. The collected data is available to be requested by registered third-party users. The first possibility is requesting data on a specific individual user, for which it is required an identification code of the individual user (Social security number, fiscal code, etc.); the second possibility is accessing anonymized data on groups of individual users, so the system will have to allow third-party users to filter the users by age, geographical area, weight, etc. while keeping the data properly anonymized. For both possibilities third-party users can simply acquire data stored or subscribe to receive data as soon as they are produced.

AutomatedSOS is a service designed to provide emergency health support for elderly people, to be integrated on top of Data4Help. Those individual users parameters about blood pressure, heartbeat and oxygen saturation are constantly checked: whenever they represent a critical health condition, AutomatedSOS, within 5 seconds from the detection of the drop of those values, contacts an Emergency Service asking to send an ambulance at the user location.

The main purpose of the Track4Run is to support running competition organizer and jogging lovers in convening great sport events with little effort. In particular Track4Run is a service-to-be focused on organizing running competition and on monitoring competitors.

Third-party users will be able to:

- organize a running competition specifying time, path of the competition, restrictions on participants and messages for those who wants to enroll;
- see the participants;
- cancel a run they have organized.

Individual users will be able to:

- discover new running competition nearby since their creation by an app notification;
- be notified if a running competition they are enrolled in is canceled;
- search for organized running competitions and register to participate in a future run or spectate an ongoing run.

During the run:

- organizer, participants and spectators will be able to see the name and position of participants;

### 1.3 Definitions

- *Data4Help System*: the whole system, offering Data4Help, AutomateSOS and Track4Run services.
- *Visitor*: someone that has not registered in yet;
- *User*: an individual or third-party who has registered;
- *Individual User*: every registered person from whom the system collects data;
- *Run Participant*: an individual who enrolled in a run;

- *Run Spectator*: an individual user who spectates a run;
- *Third-Party User*: every entity registered with the purpose to request data for external use;
- *Run Organizer*: third-party user that wants to organize a run;
- *Emergency Service*: external participant that receives and takes charge of the request for an ambulance;
- *Identification code*: a unique code for individual users or third-party users which identify them; for individual users it can be the Social security number, fiscal code, etc.; for third-party users it can be also the VAT number.
- *Stored Data*: the data on a IU collected so far.
- *Data Request*: a request for data made from a TPU.
- *Stored Data Request*: a data request for stored data.
- *Subscription Request*: a request for subscribing to newly generated data.

## 1.4 Acronyms

- RASD: Requirement Analysis and Specification Document.
- API: Application Programming Interface
- IU: Individual User
- TPU: Third-party User
- RO: Run Organizer
- RS: Run Spectator
- RP: Run Participant
- ES: Emergency Service
- ID: Identification code
- D4H: Data4Help
- ASOS: AutomatedSOS
- T4R: Track4Run

## 1.5 Abbreviations

- Gn: n-goal
- Dn: n-Domain assumption
- Rn: n-Requirement

## 1.6 Revision history

- **v0.1 - 19/10/18** Added a partial version of the introduction section
- **v0.2 - 26/10/18** Added the whole Overall description section
- **v0.3 - 1/11/18** Added the specific requirements - external interface requirements section
- **v0.4 - 3/11/18** Added the specific requirements - functional requirements, performance requirements sections
- **v0.5 - 8/11/18** Updated user interfaces section, added design constraints and software system attributes sections
- **v1.0 - 10/11/18** Added the alloy and the effort sections, updated introduction section
- **v1.1 - 11/11/18** Minor changes
- **v1.2 - 14/12/18** Nomenclature corrections, grammar corrections, minor changes on definitions, assumption and requirements.

## 1.7 Document Structure

**Introduction** In this first section the scope of the project and its goals are described in general terms and it's introduced an outline of the Data4Help System that is examined in further details in the others sections. Furthermore, recurring Acronyms, Definitions and Abbreviations used in the document are introduced to ease their understanding.

**Overview** The second section provides a more precise overview of the shared phenomena, via the product perspective and the class diagram and it also describes the main function the system will have, defining it further. It contains also the assumptions made on the outside world, the constraints on the project and the dependency between Data4Help, AutmatedSos and Track4Run.

**Specific Requirements** The third section contains a visual representation of the communication interface with the users and a description of the software and hardware interfaces. Moreover the requirements are outlined in details, via the use of scenarios and relative use cases and sequence diagrams, and mapped on the goals. Furthermore the system attributes are introduced in details.

**Formal Analysis Using Alloy** The fourth section, being D4H, ASOS and T4R fairly independent, contains three alloy models, one for each, and the relative results and generated world.

**Effort Spent** The fifth section contains the effort spent by each member of the group.

**References** Reference documents and software used to edit the document



## 2. Overview

### 2.1 Product perspective

The services are going to be used by Third party users interested in having access to statistical, individual data or in organizing competitions. To provide those services there will be a web page, a mobile application and a server application.

Individual users interact with the Data4Help System via mobile application while third-party users interact via web page.

All services provided by the Data4Help System are based on the collection of data associated to an individual user: since his registration, data coming from users device are stored in a database on the server and continuously updated.

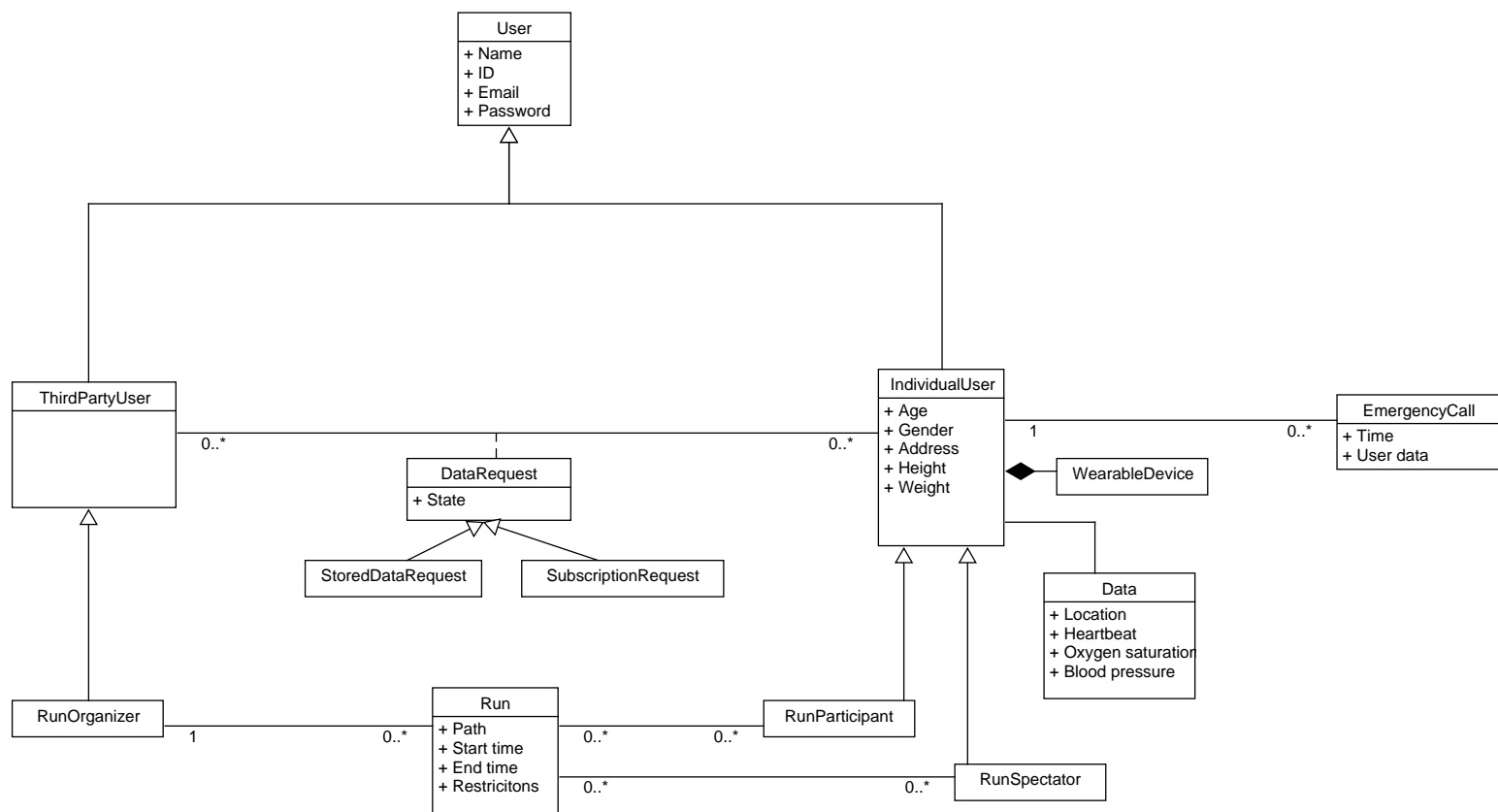
API documentation for Third-parties use will be available on the web page.

Every entity, registered as Third party, can request statistics and, if the Data4Help System can properly anonymize them, those data are available on the web page. Third-party users need previous Individual-User authorization to access individual data. An Individual-User gives or denies authorization through the Data4Help Mobile Application. Individual data are accessible on the web page to authorized Third-party users.

Real time individual data are accessible on the web page to subscribed third-party users: the authorization to the subscription must be previously given by the Individual-User on the mobile application.

Track4Run competitions organizers need to register as Third-party on the web page.

The web page provides organizers with all the features they need to arrange, delete, schedule and follow the run in real time from a map.



## **2.2 Product functions**

### **2.2.1 Data management**

During registration on the mobile application individual-users will provide their personal data (name, age, gender, address, ID, weight, height). Data4Help will grant or deny access to TPUs requests for anonymous data on a group of IUs; in case of requests for a specific IU data, it will warn the IU that a TPU is requesting access and ask for their approval; furthermore, the system will keep a record of TPUs subscribed to their data and it will allow the IUs to cancel the approval.

### **2.2.2 Data access**

The Data4Help service requires Third-party users to provide an ID during registration and then will allow them to request stored data on individual users. Data4Help will allow them to request access to data on a specific individual user after providing the ID of the individual user; it will notify them after the request is sent and when the user decides upon its approval.

Third-party users will be able also to request anonymized data by specifying a group of people according to relevant filters such as age, weight, geographical area, etc. Data4Help will analyse the request and decide whether it is possible to properly anonymize the requested data and will notify the third-party upon its decision. If possible, will grant access to the requested data. Third-party users will be able to interact with the Data4Help System with the web page or using the released APIs.

### **2.2.3 Data subscription**

The Data4Help service will allow registered third-party users to subscribe to data on users. This service will provide data to the third-party users as soon as they are produced without the need for constant approvals. In case a third-party subscribes to have real-time data for a specific user the system will act as with a stored data. If a IU deletes a previously granted access, Data4Help will warn the third-party user that had access to those data and won't provide IU's data anymore. Third-party users will be able to request data by web page or using the released APIs.

### **2.2.4 Emergency call**

AutomatesSOS service will be able to forward a request for help to the Emergency Service within 5 seconds from the moment it detects a dangerous drop in health parameters values. The Emergency Service takes charge of the request and sends an ambulance.

### **2.2.5 Running competition organization**

Track4Run service will allow users to organize running competitions. It will require to a RO:

- time of the competition
- path of the competition
- restrictions on participants
- an optional message for those who wants to enroll

The system provides the RO with data about users enrolled.

### 2.2.6 Running competition monitoring

Track4Run service will allow users to monitor a running competition. The system will provide the RS with names and real-time position of all the participants.

### 2.2.7 Running competition enrolling

Track4Run service will allow users to enrol to an already existing competition. Track4Run will provide participants with the final ranking and their arrival time.

## 2.3 User characteristics

- *Visitor*: an individual or a third-party, that has not registered in yet and has access only to registration;
- *User*: an individual or a third-party who has registered;
- *Individual user*: every registered person from whom the system collects location and health data. The user has access to the AutomatedSOS and Track4Run services;
- *Run participant*: an individual user who enrolled and takes part in a run;
- *Run spectator*: an individual user who spectates a run interested in having access to all information about it;
- *Third-party user*: every entity registered with the purpose to request data from Data4Help for external use or to organize a running competition taking advantage of Track4Run services;
- *Run organizer*: third-party users that want to organize a run competition and receive all information about it and its participants
- *Emergency service*: external participant that receives and takes charge of the request for an ambulance.

## 2.4 Assumptions

- D1** The age, height, weight, address, gender and name provided by the user on themselves are correct
- D2** The ID is unique
- D3** Email address in unique
- D4** The email is currently in use
- D5** The kind and accuracy of data collected on the users health conditions is comprehensive and enough to determine their health status and their location within 10 meters
- D6** The user has a wearable device connected to their smartphone
- D7** The smartphone can successfully contact the Emergency Service 24/7
- D8** Emergency Service takes charge of every request sending an ambulance
- D9** The ROs organize only authorized running competitions respecting laws and common sense
- D10** The IU's device has an internet connection and GPS integrated

## **2.5 Constraints**

- IUs' personal data and GPS signal must be acquired with their consent.
- IUs' personal data must be stored and transmitted safely.
- Data coming from IUs device are collected since their registration, stored in a server and continuously updated.

## **2.6 Dependencies**

Due to its nature, the Data4Help System can be divided in three services having no dependencies among themselves: Data4Help, AutomatedSOS and Track4Run. However, the D4H Mobile Application will host T4R and ASOS services.

## 3. Specific Requirements

### 3.1 External Interface Requirements

#### 3.1.1 Individual User Interface

The following mock-ups intend to represent a likely mobile application layout

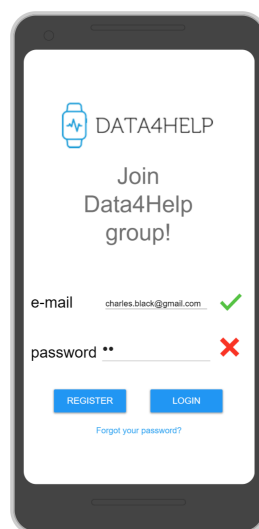


Figure 3.1: **Login screen:** Individual Users must insert their data in order to log in their (personal) account. A user can also recover his password whether it has been forgotten. Once all fields are filled, visitors need to press register button.

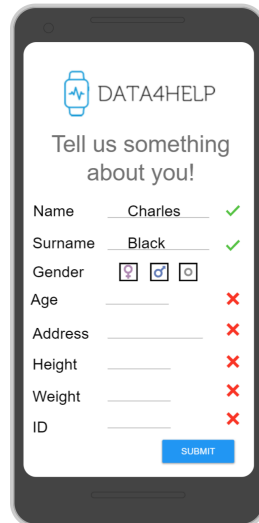


Figure 3.2: **Sign in screen:** Visitors must submit their data in order to become individual users.

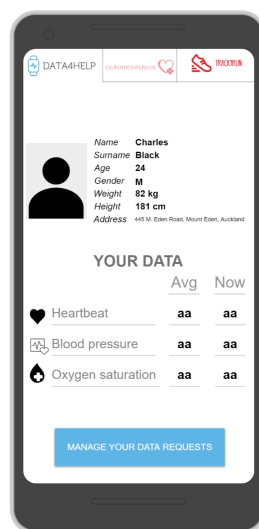


Figure 3.3: **D4H Main Screen:** Individual Users can look at their personal data and monitor data about their heartbeat, blood pressure and oxygen saturation. From this screen they can also access the Request Monitoring Screen.

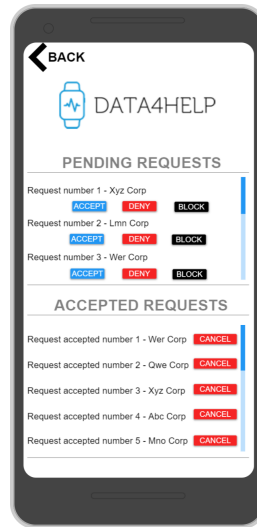


Figure 3.4: **D4H Requests Monitoring Screen:** Individual Users can accept or deny a request, block a request sender or cancel an already accepted request.

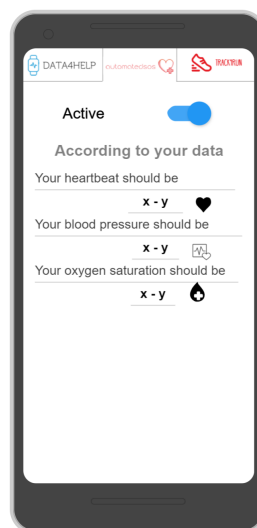


Figure 3.5: **ASOS Main Screen:** Individual Users can consult the recommended range of health parameters



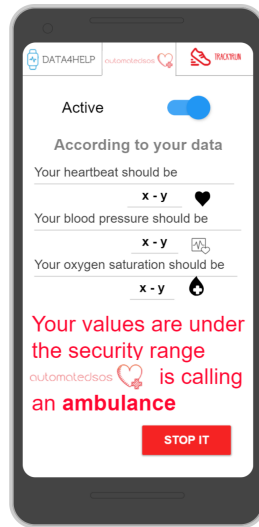


Figure 3.6: **ASOS Emergency Screen:** This screen is displayed whenever a parameter going out of health range has been detected.

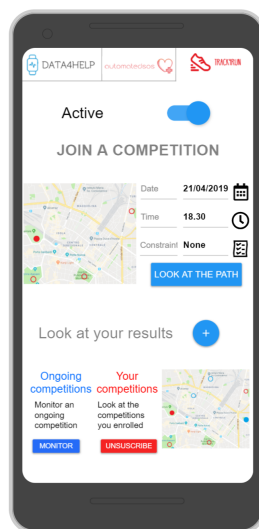


Figure 3.7: **T4R Main Screen:** This screen is predisposed to display competitions on a map that users can select and enrol to. On the bottom Individual User can consult competition he took part, monitor an ongoing competition or unsubscribe to a competition he previously enrolled to.

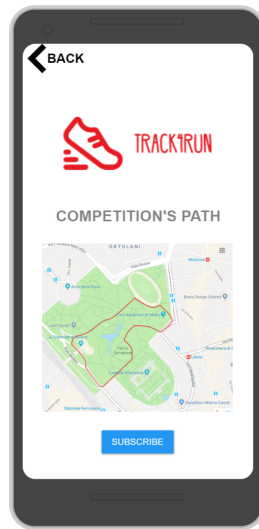


Figure 3.8: **T4R Path Run Screen:** Individual Users can look at a competition path.

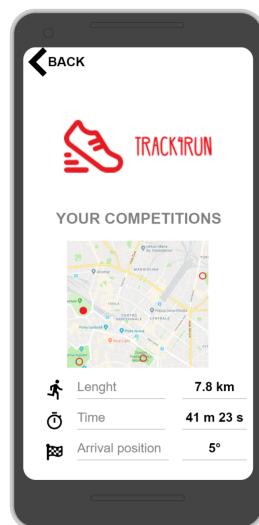


Figure 3.9: **T4R Previous Competitions Screen:** Individual Users can consult their (personal) previous competitions results.

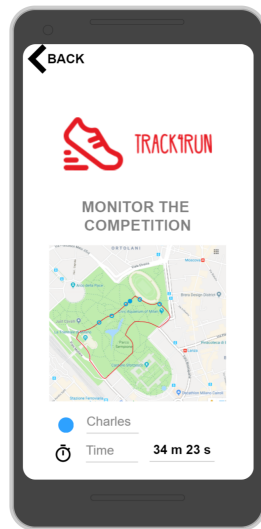
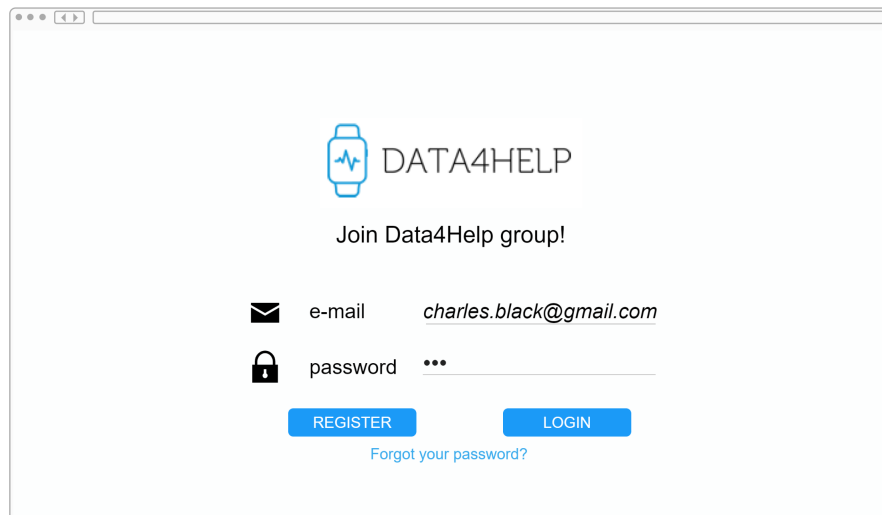


Figure 3.10: **T4R Monitoring Competitions Screen:** Individual Users can monitor an ongoing competition

### 3.1.2 Third-party User Interface

The following mock-ups intend to represent a likely web page layout



DATA4HELP

Join Data4Help group!

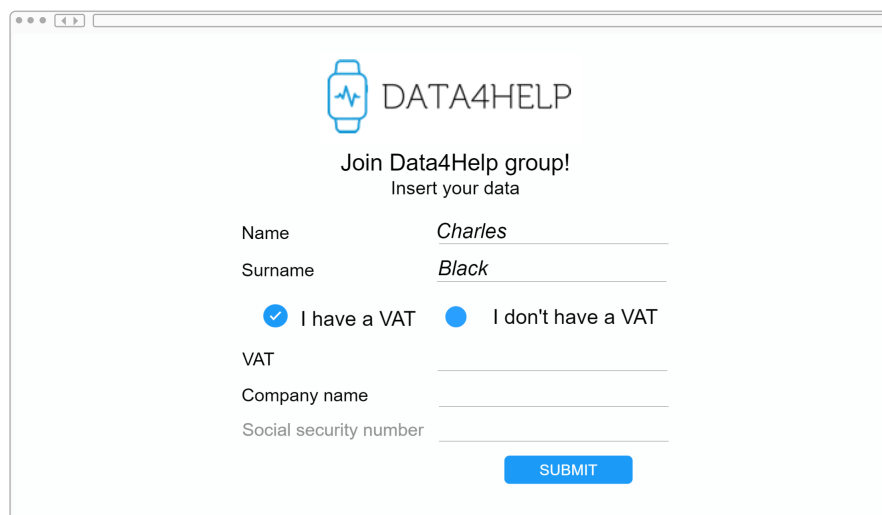
e-mail *charles.black@gmail.com*

password \*\*\*

REGISTER LOGIN

[Forgot your password?](#)

Figure 3.11: **Login window:** Third-Party users must insert their data in order to log in their (personal) account. Password can be recovered whenever it has been forgotten. Once all fields are filled, third-parties users need to press register button.



DATA4HELP

Join Data4Help group!  
Insert your data

Name *Charles*

Surname *Black*

☒ I have a VAT ☐ I don't have a VAT

VAT

Company name

Social security number

SUBMIT

Figure 3.12: **Sign In window:** Third-Party users have to insert their data in order to become Third-Parties users: if the visitor has a VAT he has to insert it as ID, otherwise another unique identifier must be inserted.

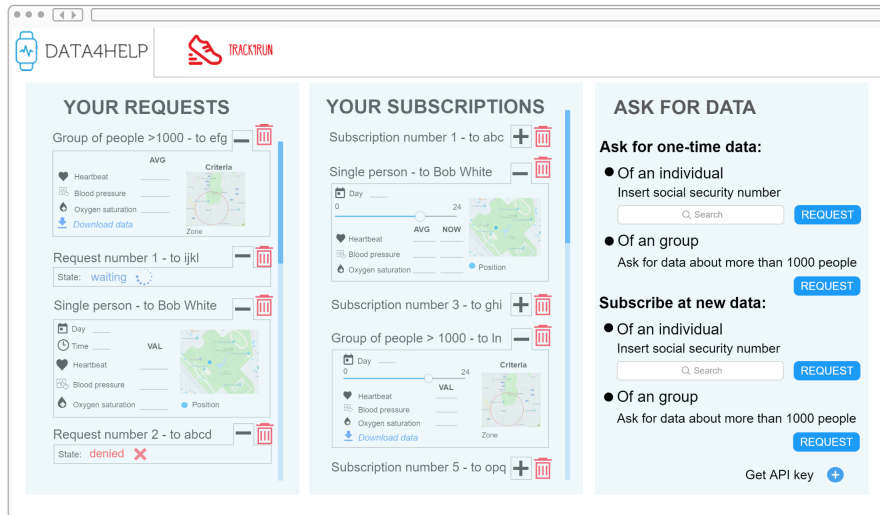


Figure 3.13: **D4H Main window:** Third-party Users can monitor their requests for one-time data and their subscriptions. They can make a new request for a subscription or a one-time data of an individual (typing the social security number or any equivalent security code). They can also get API key or make requests for data regarding a group of people (one-time data or subscription, the D4H data request constrains window will follow).

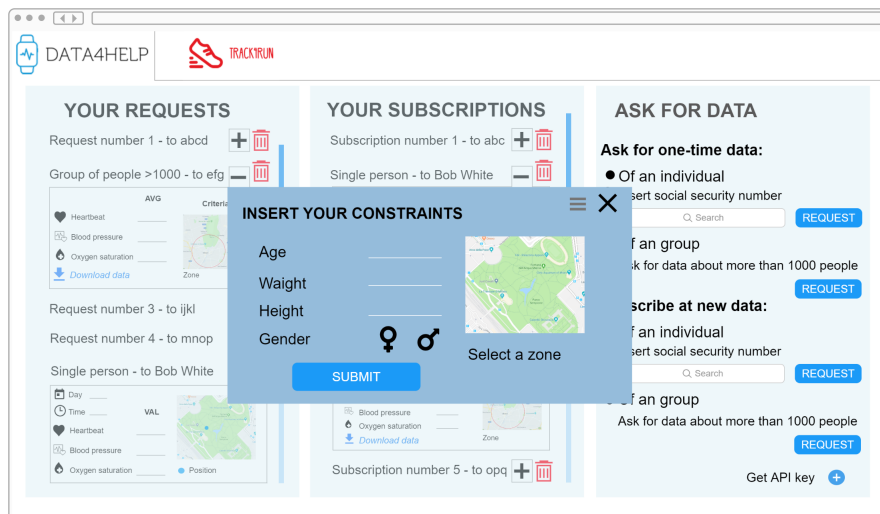


Figure 3.14: **D4H data request constraints window:** Third-Party Users can specify the constraint to be applied to group data request.

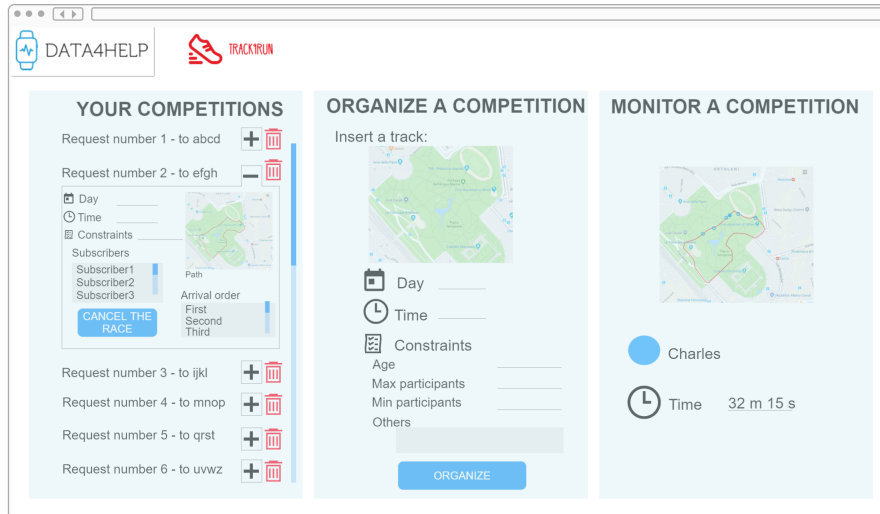


Figure 3.15: **T4R Main window:** Third-Party Users can manage the competition they organized: check subscribers, delete the competition before it occurs and monitor the competition while takes place. They can organize a new competition (field about constraints can be empty) or monitor an ongoing one.

### 3.1.3 Hardware interfaces

The Data4Help System consists just in a software application and is not using any hardware interfaces indeed; the IU, as already mentioned in the assumption [D10], must be provided with a smartphone internet connection and GPS, a smartwatch or an equivalent wearable device. Bluetooth might also be required in the wearable device-smartphone connection.

### 3.1.4 Software interfaces

The Data4Help System provides a series of functionalities that are based on the precise location of the user in every moment: none of the services provided by AutomatedSOS and Track4Run are effective without a map. The map to be used needs to be user-friendly and available on every smartphone: Google Maps meets these needs. The development will be done with tools and API provided by Google Maps Platform (<https://cloud.google.com/maps-platform/>).

To monitor health parameters the Data4Help System needs the user to constantly wear a smartwatch: it access data measured by the smartwatch making use of Data Layer APIs.

The Data4Help System needs to interface with a database that is accessed with a DBMS (PostgreSQL).

### 3.1.5 Communication interfaces

The Data4Help System implementation should support real time communication: developers are up to ensure a secure and rapid data transmission availing themselves of communication protocols at transport layer such as UDP, TCP, real time protocols.

At application layer must be used HTTPS (HTTP + SSL certificate) to transmit safely sensitive data while at network layer it could be used IP v6 to have a more efficient routing. SMTP protocol could be used to verify the e-mail provided by the user.

## **3.2 Functional Requirements**

### **3.2.1 Scenarios**

#### **Scenario 1**

Alice's gym would like to collect data on their members, so Alice was asked to install the Data4Help Mobile Application. She downloaded it, entered her name, age, ID, password etc., she accepted the data sharing policy and, after receiving an email, confirmed her email address. Now she received a notification from the app, asking if she wants to accept or deny the request for her data, made from her gym, which she accepts.

#### **Scenario 2**

Bob, historic user of the Data4Help System, is getting older and older so he wants to activate the Emergency Service, AutomatedSOS. He logs in the application and requests the activation of the service. Incidentally the very next day, while working from home, he has a cardiac arrest. Immediately the application calls the Emergency Service that sends an ambulance to his house, saving his life.

#### **Scenario 3**

Charles and Devon are passionate runners and love to compete one with the other. To be able to find more opportunities to race they installed the Data4Help Mobile Application and activated the Track4Run service. Soon they enroll in a race as participants but unfortunately Devon, the day before of the race, got injured, so he forfeit from the run. The day after, wishing to know how his friend Charles is doing he opens the app and spectates the run. After Charles arrives first he gets a call from Devon telling him that next time he will arrive second.

#### **Scenario 4**

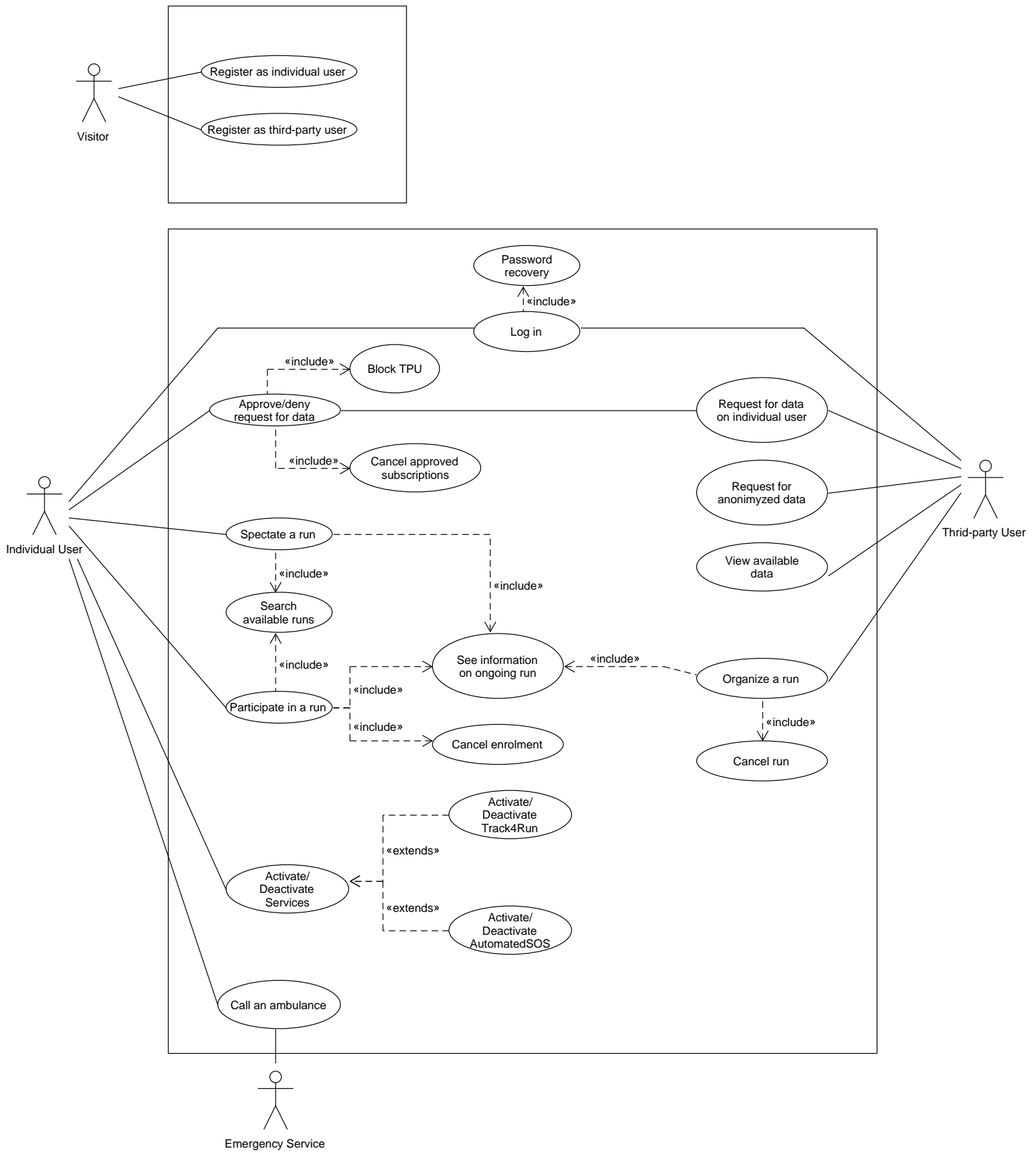
EvilCorp wants to collect data on the inhabitants of a small town. Felix the supervisor registers the company on the web page, entering name, ID(VAT), etc. and waits for a confirmation email. Then he enters the web page again and enters the data required to make a request for data on individual users living in the area. Unfortunately for EvilCorp, there were not enough users to properly anonymize the data, so the request is rejected and Felix is fired.

#### **Scenario 5**

GymForEveryone wants to personalize more the workout session of its premium members. In order to do so Helen, the manager, logs in the gym account and sends, one by one, the request to the data of the premium members. The day after some of them have accepted the request while some of them denied it. With the data obtained she starts creating new workout sessions.

#### **Scenario 6**

IHealth wants to monitor large groups of people in order to collect data for research. In order to do that its researchers created a company account via web page. Jhon the scientist then makes a request to monitor the data of people living in Milan to evaluate stress levels. Being able to properly anonymize the data, the system gives him access to live data. After some days, shocked by the results, starts funding a campaign to urge Milan to work less and relax more.





### 3.2.2 Use cases

#### Registration as IU

<b>ACTORS</b>	Visitor
<b>GOALS</b>	<b>G2</b>
<b>INPUT CONDITIONS</b>	None
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The visitor downloads the mobile application</li><li>2. The visitor selects the "Sign in" option to register</li><li>3. The visitor fills all the mandatory fields (Name, ID, Age, Gender, Height, Weight, Address, Email, Password)</li><li>4. The visitor accepts the data policy</li><li>5. The system sends an email to the Visitor</li><li>6. The Visitor confirms his email</li></ol>
<b>OUTPUT CONDITIONS</b>	The visitor is an IU
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. The email or the ID is already in use</li><li>3. The visitor inserts invalid data</li></ol> Exceptions cause the Visitor to be notified and the flow to go back to point 2.

## Registration as TPU

<b>ACTORS</b>	Visitor
<b>GOALS</b>	<b>G2</b>
<b>INPUT CONDITIONS</b>	None
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The visitor opens the website</li><li>2. The visitor selects the "Sign in" option to register</li><li>3. The visitor fills all the mandatory fields (Name, ID(VAT), Address, Email, Password)</li><li>4. The system verifies the VAT</li><li>5. The Visitor receives a confirmation email</li></ol>
<b>OUTPUT CONDITIONS</b>	The visitor becomes a TPU
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. The visitor is already a TPU</li><li>2. The email is already in use</li><li>3. The visitor doesn't fill all the fields</li><li>4. The visitor inserts invalid data</li></ol> <p>Exceptions cause the Visitor to be notified and the flow to go back to point 2.</p>

## Login as IU

<b>ACTORS</b>	Individual User
<b>GOALS</b>	<b>G2, G3</b>
<b>INPUT CONDITIONS</b>	The IU is registered
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The IU opens the mobile application</li><li>2. The system shows them the home page</li><li>3. The IU inserts email and password</li><li>4. The IU clicks the "Log in" button</li><li>5. The system validates the fields entered</li><li>6. The system shows the IU health and location data</li></ol>
<b>OUTPUT CONDITIONS</b>	The IU is logged in
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. The IU enters invalid data</li></ol> Exceptions bring the flow back to point 2

### Login as TPU

<b>ACTORS</b>	Third-Party User
<b>GOALS</b>	<b>G1</b>
<b>INPUT CONDITIONS</b>	The TPU is registered
<b>EVENT FLOW</b>	1. The TPU opens the website home page 2. The TPU inserts email and password 3. The TPU clicks the "Log in" button 4. The system validates the fields entered
<b>OUTPUT CONDITIONS</b>	The TPU is logged in
<b>EXCEPTIONS</b>	1. The TPU enters invalid data Exceptions bring the flow back to point 1

## Password recovery

s

<b>ACTORS</b>	User
<b>GOALS</b>	<b>G1</b>
<b>INPUT CONDITIONS</b>	None
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The User opens the website</li><li>2. The User selects the "Forgot your password" option</li><li>3. The User provides their email</li><li>4. The system sends a mail with a link to change password</li><li>5. The User follows the link and enters a new password</li></ol>
<b>OUTPUT CONDITIONS</b>	The User has changed password
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. The email doesn't exist</li></ol> Exceptions cause the User to be notified and the flow to go back to point 1.

### Request for IU data

<b>ACTORS</b>	Third-Party User, Individual User
<b>GOALS</b>	<b>G4</b>
<b>INPUT CONDITIONS</b>	The TPU is logged in
<b>EVENT FLOW</b>	1. The TPU opens the "request data on IU" section 2. The TPU provides the IU ID 3. The system sends the IU a notification
<b>OUTPUT CONDITIONS</b>	The IU has a new request for data
<b>EXCEPTIONS</b>	1. The ID doesn't match any IU Exceptions bring the flow back to point 1

## Request Approval

<b>ACTORS</b>	Third-Party User, Individual User
<b>GOALS</b>	<b>G5</b>
<b>INPUT CONDITIONS</b>	The TPU is logged in
<b>EVENT FLOW</b>	1. The IU opens the "request of data" section 2. The IU clicks on the "Approve" button of a request 3. The system sends a notification to the TPU that made the request
<b>OUTPUT CONDITIONS</b>	The request is approved
<b>EXCEPTIONS</b>	1. The IU doesn't have any requests Exceptions bring the flow back to point 1

## Request Denied

<b>ACTORS</b>	Third-Party User, Individual User
<b>GOALS</b>	<b>G5</b>
<b>INPUT CONDITIONS</b>	The TPU is logged in
<b>EVENT FLOW</b>	1. The IU opens the "request of data" section 2. The IU clicks on the "Deny" button of a request 3. The system sends a notification to the TPU that made the request
<b>OUTPUT CONDITIONS</b>	The request is denied
<b>EXCEPTIONS</b>	1. The IU doesn't have any requests Exceptions bring the flow back to point 1



**TPU blocked**

<b>ACTORS</b>	Third-Party User, Individual User
<b>GOALS</b>	<b>G5</b>
<b>INPUT CONDITIONS</b>	The IU is logged in
<b>EVENT FLOW</b>	1. The IU opens the "request of data" section 2. The IU clicks on the "Block" button of a request 3. The system sends a notification to the TPU that made the request 5. The other requests from that TPU are denied
<b>OUTPUT CONDITIONS</b>	The TPU is blocked
<b>EXCEPTIONS</b>	1. The IU doesn't have any requests Exceptions bring the flow back to point 1

### Request for anonymized data

<b>ACTORS</b>	Third-Party User
<b>GOALS</b>	<b>G6</b>
<b>INPUT CONDITIONS</b>	The TPU is logged in
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The TPU opens the "request anonymized data" section</li><li>2. The TPU provides the request parameters</li><li>3. The system evaluates whether the data requested are properly anonymized</li><li>4. The system sends a notification to the TPU</li></ol>
<b>OUTPUT CONDITIONS</b>	The TPU has access to the data
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. The data cannot be properly anonymized</li><li>2. The data entered is incorrect</li></ol> Exceptions bring the flow back to point 1

### AutomatedSos activation

The activation of the Track4Run is omitted since is equivalent

<b>ACTORS</b>	IU
<b>GOALS</b>	<b>G1</b>
<b>INPUT CONDITIONS</b>	The IU is already registered and has not yet activated the AutomatedSOS service
<b>EVENT FLOW</b>	1. The IU opens the application 2. The IU opens the AutomatedSOS section 3. The system shows the relevant information 4. The IU clicks on the button to activate the service
<b>OUTPUT CONDITIONS</b>	The IU joined the service
<b>EXCEPTIONS</b>	None

### AutomatedSos deactivation

<b>ACTORS</b>	IU
<b>GOALS</b>	<b>G1</b>
<b>INPUT CONDITIONS</b>	The IU is already registered and has activated the AutomatedSOS service
<b>EVENT FLOW</b>	1. The IU opens the application 2. The IU opens the AutomatedSOS section 3. The IU opens the options menu 4. The IU clicks on the button to deactivate the service
<b>OUTPUT CONDITIONS</b>	The service is deactivated
<b>EXCEPTIONS</b>	None

### Track4Run deactivation

<b>ACTORS</b>	IU
<b>GOALS</b>	G1
<b>INPUT CONDITIONS</b>	The IU is already registered and has activated the Track4Run service
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The IU opens the application</li><li>2. The IU opens the Track4Run section</li><li>3. The IU opens the options menu</li><li>4. The IU clicks on the button to deactivate the service</li><li>5. The system unregisters the IU as participant in any run they enrolled</li></ol>
<b>OUTPUT CONDITIONS</b>	The service is deactivated
<b>EXCEPTIONS</b>	None

## Emergency call

<b>ACTORS</b>	IU, Emergency Service
<b>GOALS</b>	<b>G7</b>
<b>INPUT CONDITIONS</b>	IU has activated the AutomatedSOS service
<b>EVENT FLOW</b>	1. The system registers a critical health condition 2. The system prompts a warning and a "Don't call" button for a few seconds 3. The system contacts the ES and gives him all relevant information
<b>OUTPUT CONDITIONS</b>	The ES is in charge of the call
<b>EXCEPTIONS</b>	1. The IU clicks the "Don't call" button Exceptions cause the IU to be notified and the flow to go back to point 1.

### Organize a run

<b>ACTORS</b>	TPU
<b>GOALS</b>	<b>G8</b>
<b>INPUT CONDITIONS</b>	TPU is logged in the website
<b>EVENT FLOW</b>	1. The TPU opens the "Organize" section 2. The TPU is shown a map to choose the path 3. The TPU enters a time, duration, restriction on participants and a message for the participants 4. The TPU confirms the creation
<b>OUTPUT CONDITIONS</b>	The run is organized
<b>EXCEPTIONS</b>	None

### Cancel run

<b>ACTORS</b>	TPU, IU
<b>GOALS</b>	<b>G8</b>
<b>INPUT CONDITIONS</b>	The TPU is on the home page
<b>EVENT FLOW</b>	1. The TPU opens the "Organized runs" section 2. The TPU selects a run 3. The system shows the enrolled participants and their number 4. The TPU clicks the "Cancel run" button 5. The Participants are notified by the system
<b>OUTPUT CONDITIONS</b>	The run is canceled
<b>EXCEPTIONS</b>	1. The TPU hasn't organized any run Exceptions cause the flow to go back to point 1.



### Spectate a run

<b>ACTORS</b>	IU
<b>GOALS</b>	<b>G10</b>
<b>INPUT CONDITIONS</b>	IU is in the Track4Run section of the application
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The system shows a list of available runs, from closest to farthest</li><li>2. The IU selects one run</li><li>3. The IU clicks the "Monitor" button</li><li>4. The system shows the map and all info on the participants</li></ol>
<b>OUTPUT CONDITIONS</b>	The IU is spectating the run
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. There are no run to spectate</li><li>2. The selected run has not yet started</li></ol> Exceptions cause the flow to go back to point 1.

### Participate in a run

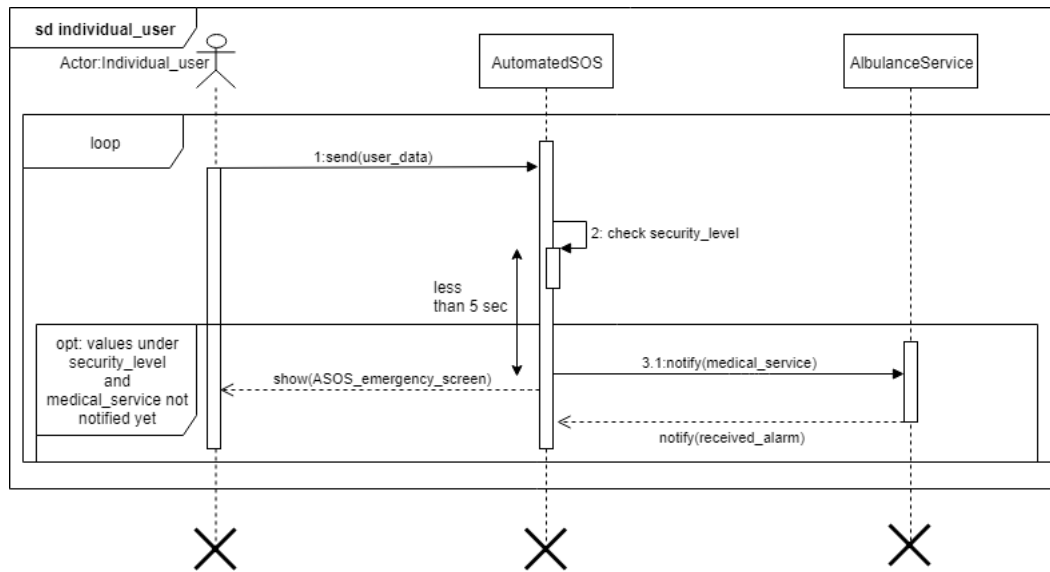
<b>ACTORS</b>	IU
<b>GOALS</b>	<b>G9</b>
<b>INPUT CONDITIONS</b>	The IU is in the Track4Run section of the application
<b>EVENT FLOW</b>	<ol style="list-style-type: none"><li>1. The system shows a list of available runs, from closest to farthest</li><li>2. The IU selects one run</li><li>3. The system shows date, duration, path and the message of the organizer</li><li>4. The IU clicks the "Participate" button</li><li>5. The system adds the IU to the participants</li></ol>
<b>OUTPUT CONDITIONS</b>	The IU is a participant of the run
<b>EXCEPTIONS</b>	<ol style="list-style-type: none"><li>1. There is no run to take part in</li><li>2. The selected run has already started</li><li>3. The run restrictions prevent the IU from participating</li></ol> Exceptions cause the flow to go back to point 1.

### Cancel enrolement

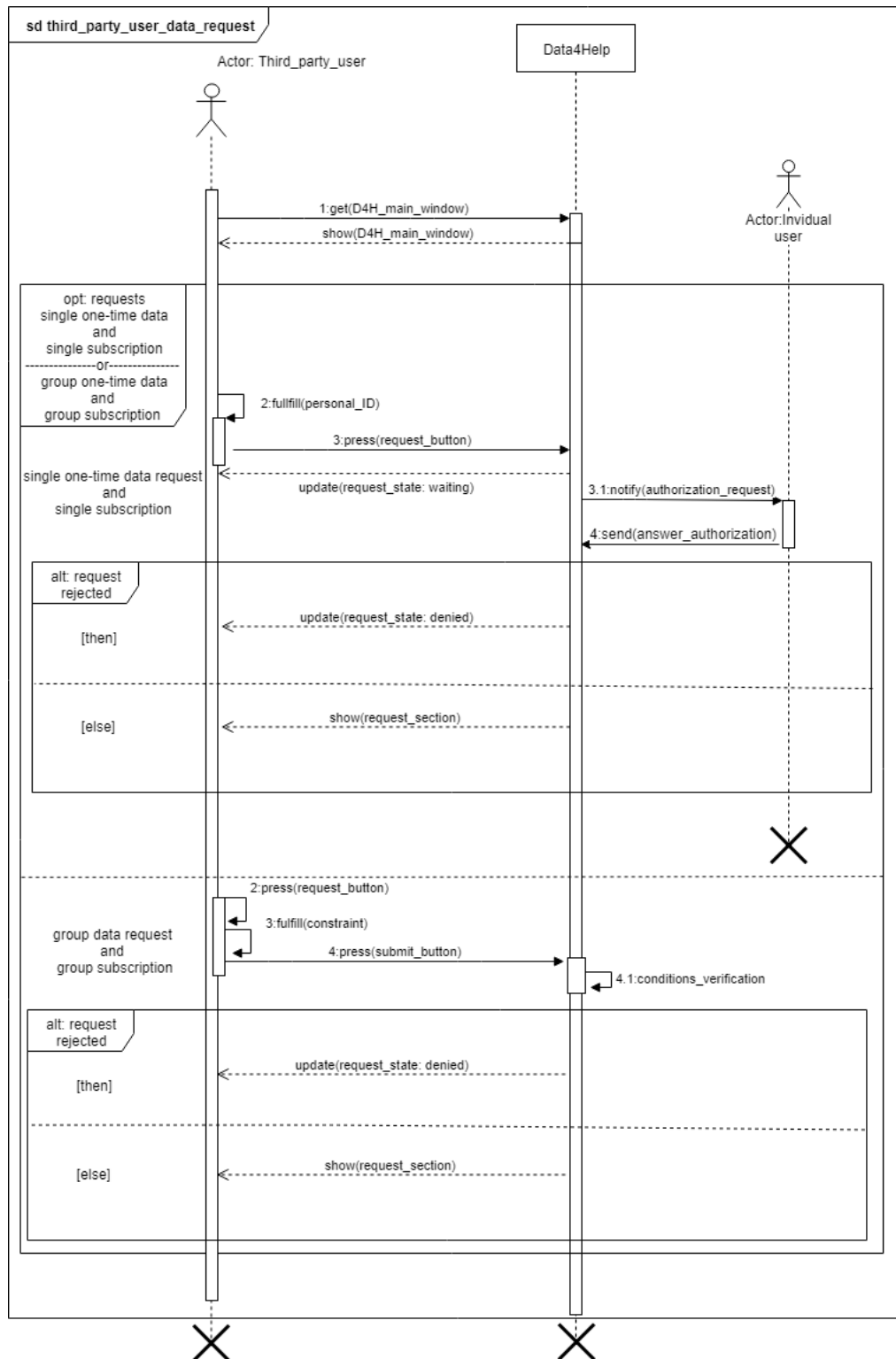
<b>ACTORS</b>	IU
<b>GOALS</b>	<b>G1, G9</b>
<b>INPUT CONDITIONS</b>	The IU is in the Track4Run section of the application
<b>EVENT FLOW</b>	1. The IU opens the section of a run he is enrolled in 2. The IU clicks on "Unsubscribe" option
<b>OUTPUT CONDITIONS</b>	The IU is no longer a participant in the run
<b>EXCEPTIONS</b>	1. There is no run to unsubscribe from Exceptions cause the flow to go back to point 1.

### 3.2.3 Sequence diagram

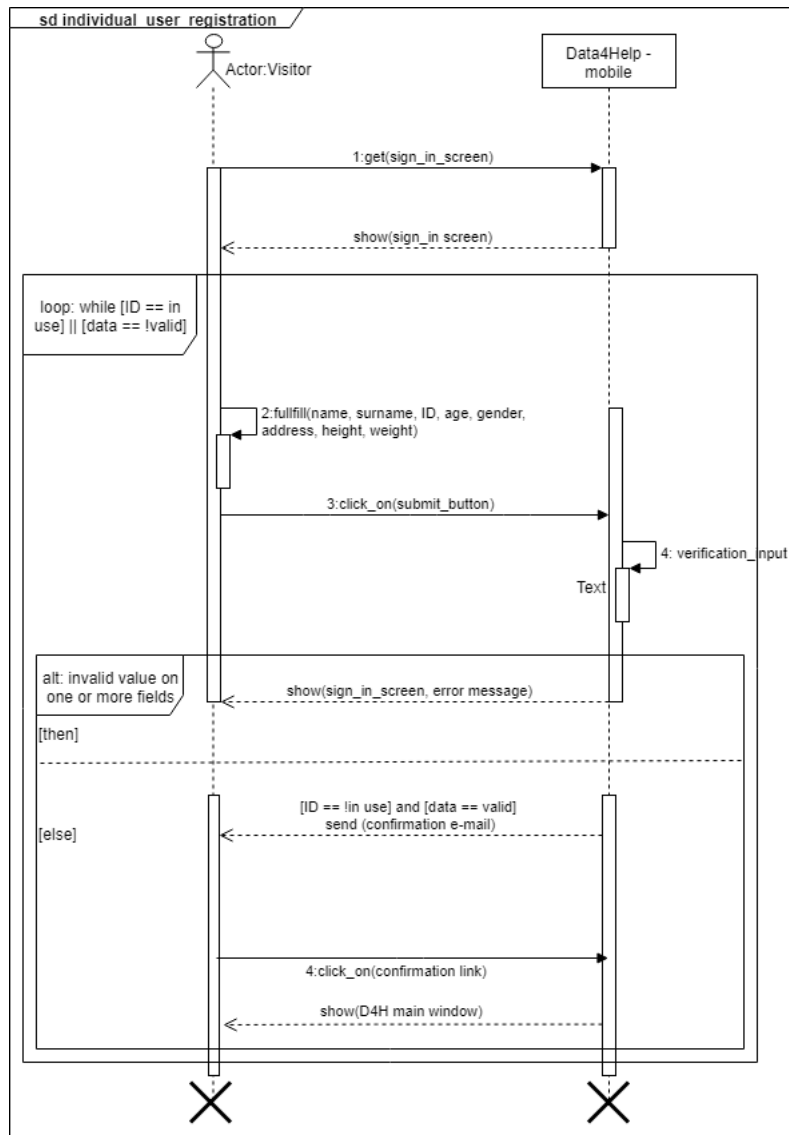
#### Ambulance call



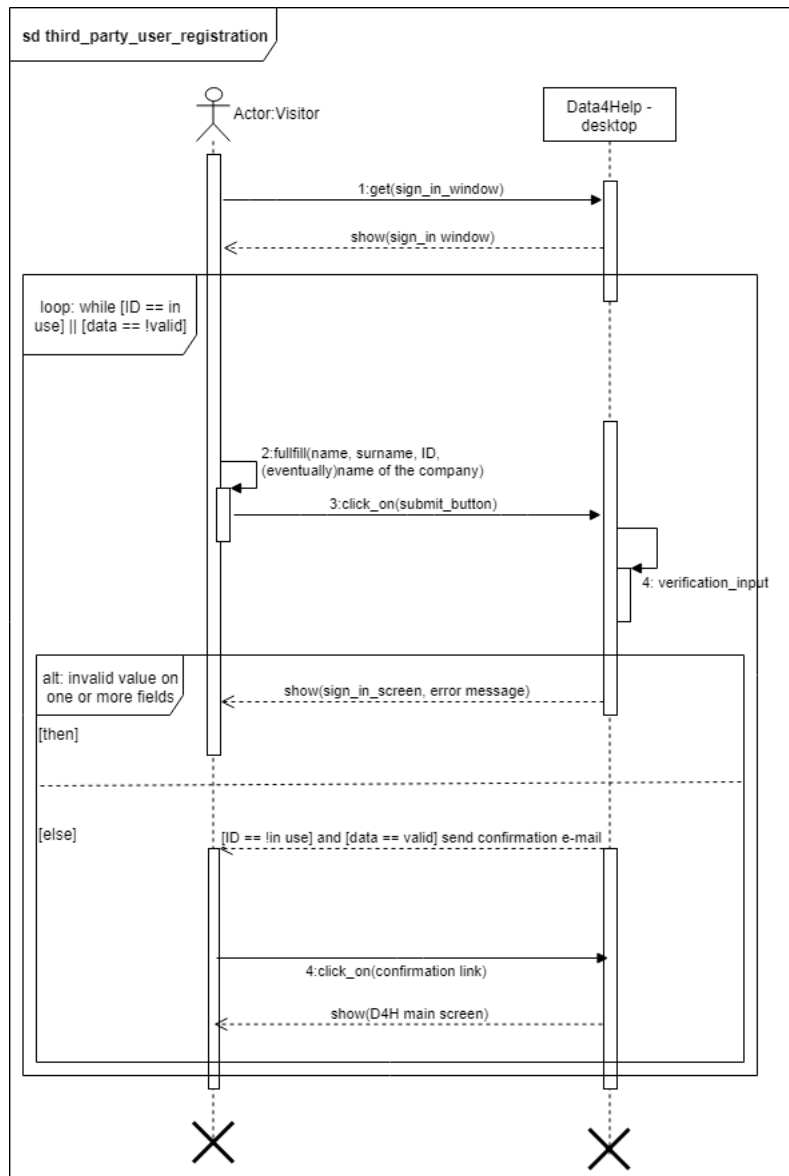
## Data request



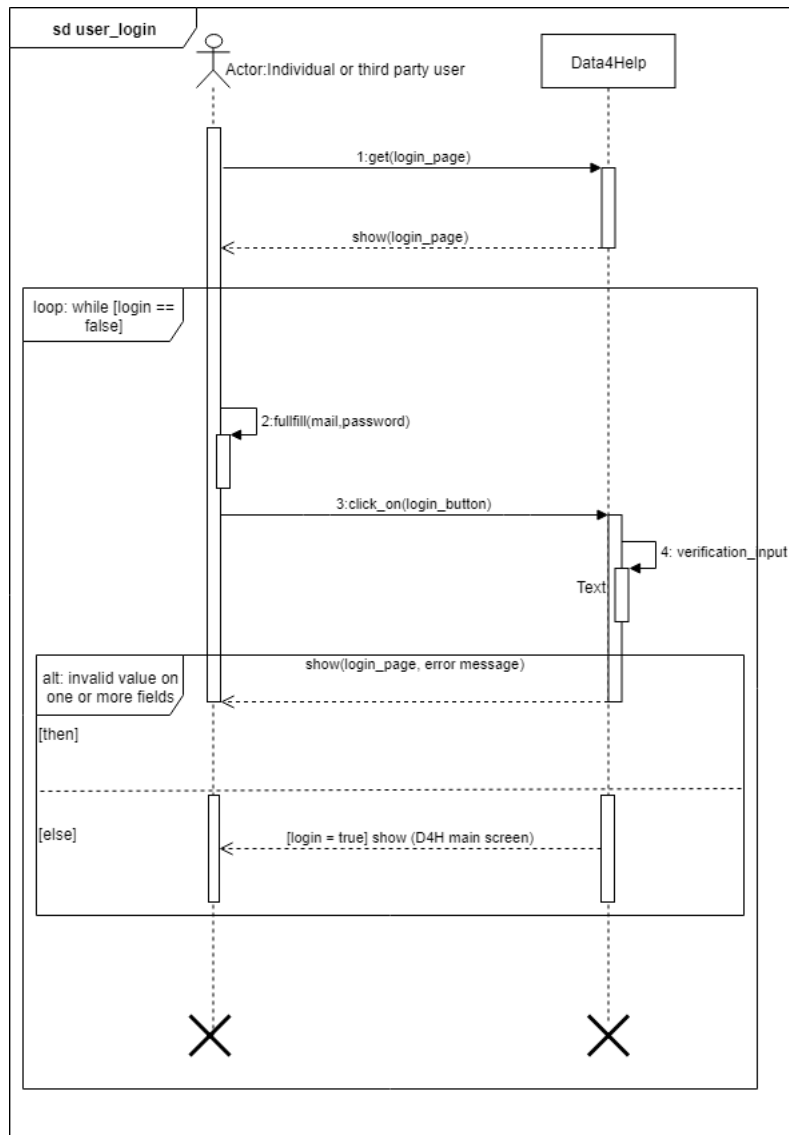
## IU registration



## TPU registration

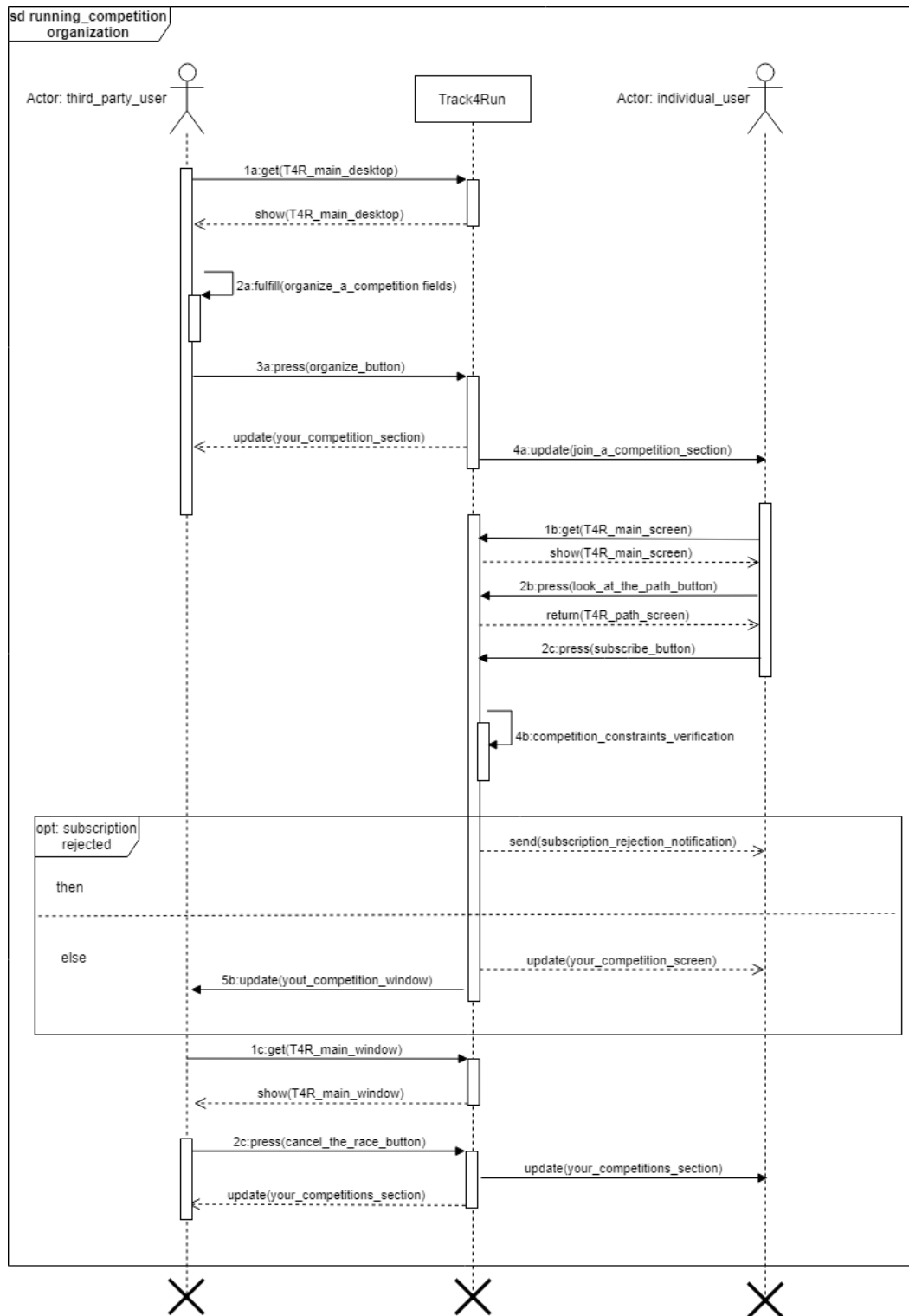


## Login





## Race organization



### **3.2.4 Requirements mapping**

**G1** Allow users to properly use the services they wish to employ

**R1** The User must be able to log in

**R2** An IU must be able to opt in and out of any additional service besides D4H

**G2** Allow visitor to register as individual or third-party user

**D1, D2, D3, D4**

**R4** A visitor must be able to register to the Data4Help System

**R4.1** A visitor must insert name, age, gender, height, weight, ID, address, email and password

**R4.2** A third party must insert name, ID, email and password

**G3** Allow individual users to monitor their location and health parameters

**D5, D6**

**R5** The system must provide to logged IU their current location and health parameters

**R6** The system must collect and keep track of the location and health data

**G4** Allow third-party users to request the data on specific users

**D2**

**R7** A TPU must be able to request the stored data or subscribe to newly produced data, after providing an ID of the user

**R7.1** A TPU must be warned whether the data they requested is available

**R8** A TPU must be able to obtain the data it has access to

**G5** Allow individual users to approve or deny the specific request for their data

**R9** The IU must be notified when a TPU request their data

**R10** An IU must be able to accept the request on their data

**R11.1** An IU must be able to deny access to their data

**R11.2** An IU must be able to block a TPU to automatically deny their future requests for data

**R12** User must be able to cancel the TPU subscriptions to their data that they previously accepted

**G6** Allow third-party users to request data on anonymized groups of individual users

**R13.1** A logged in TPU must be able to request the stored data or subscribe to newly produced data, after providing the specifics of the group it is interested in

**R13.2** A TPU must be warned whether it is possible to properly anonymize the data requested

**R8** A TPU must be able to obtain the data it has access to

**G7** Call an ambulance if the system detects a critical health condition

**D5, D7, D8**

**R14** When a critical condition is registered, the system should contact the ES and provide it with the location and health status of the IU

**R15** The IU must be able to cancel the call

**G8** Allow third-party users to organize running competitions

**D9**

**R16.1** A TPU must be able to create a new run, specifying time, duration, path, restriction for enrolment

**R16.2** An organizer must be able to cancel a run

**R17** Run organizers, participants and spectators must be able to see the status of the ongoing run

**G9** Allow Individual users to enrol in existing running competitions as participants

**D9**

**R18.1** An IU must be able to enrol in a future run

**R18.2** A participant must be able to cancel their inscription to the run

**R17** Run organizers, participants and spectators must be able to see the status of the ongoing run

**G10** Allow Individual users to subscribe in existing running competition as spectators to monitor underway competitions

**D9**

**R17** Run organizers, participants and spectators must be able to see the status of the ongoing run

### **3.3 Performance Requirements**

The Data4Help Mobile Application must respond within 5 seconds from the reception of a datum that reports a parameter that is under the threshold of health range. The response indeed corresponds to take contact with Emergency Service that takes charge of the request and sends an ambulance.

Real time data available on server are necessary, for this reason the mobile application needs to have a worst-case latency under 30 ms that does not consider the time spent to reach the server over the network. This latency value only refers to the internal application process which starts from the reception of a datum and ends with its forwarding.

System availability must be at least 99.995% which means that no more than 26.30 minutes of downtime in a year is allowed.

Failure events cannot exceed three times over a year as reported in system reliability.

### **3.4 Design Constraints**

#### **3.4.1 Standards Compliance**

The web page will be developed taking in account Standards Compliance defined by the World Wide Web Consortium (W3C) with the purpose of enhancing interoperability among different browsers.

The web page needs its compliance to be tested, before the release, submitting the website URL on the Validator Tool ([validator.w3.org](http://validator.w3.org)) and consulting errors reported: fully compliance (meaning zero errors on Validator Tool) its really hard to achieve and in most of the cases its not necessary, moreover there are cases in which its needed to break some of W3C rules to make something work in a specific desired way on every browser.

The web page to release needs the result coming from Validator Tool to have at most 10 errors, an indefinite number of warnings are allowed.

## **3.5 Software System Attributes**

### **3.5.1 Reliability**

Reliability is a parameter to evaluate the quality of a software (or hardware) system that can be measured by different reliability metrics (hardware metrics are not suitable for softwares):

for intermittent demands its useful the POFOD (probability of failure on demand) that is the likelihood the system will fail when a request is made, while ROCOF (rate of occurrence of failures) is used when the system has to process a large number of similar requests in a short time. This metric permits to select time units (such as row execution time, calendar time, number of transactions). Reliability is defined indeed as the measure of how long the system performs its intended function.

The demands rate for the Data4Help System is likely to be unpredictable according to different options user can activate and the intensity of the interaction the user intends to have with the application or web page. For these reasons ROCOF metric is more appropriate to evaluate the Data4Help System reliability: the time unit selected is the year and the occurrence of failure over this period must not exceed two times.

### **3.5.2 Availability**

Availability is a parameter particularly relevant for continuously running systems: its a measure of the percentage of time the system is available for use and considers the restart and the repair time. Availability can be calculated as the Mean Time Between Failure (MTBF, uptime) divided by the sum of MTBF and the Mean Time To Repair (MTTR, downtime including restart).

Taking in account the fact that the Data4Help System operates in health services, it has been selected a four-and-a-half-nine availability that means a percentage of 99.995, a downtime per year of 26.30 minutes, 2.19 minutes per month.

### **3.5.3 Security**

Security of a system reflects the ability to guarantee confidentiality, integrity of data and authenticity over treats from both accidental and malicious events.

The Data4Help System authentication process (made from user identification and password) preserves confidentiality allowing users to access dedicated resources only: sensitive data such as passwords need to be encrypted implementing Hash-with-Salt or any stronger mechanism for encryption.

The Data4Help System uses TLS (transport layer security cryptographic protocol) to encrypt communications between the user and server(s).

Backups, kept both on cloud and on hard disks, need to be fully encrypted to prevent unauthorized access. When connections are established, HTTPS (HTTP + SSL certificate at application layer) is selected over HTTP.

Keeping data consistent over time is the concern behind Integrity: The Data4Help System avails itself of Web Application Firewalls (WAFs); to reduce human induced integrity errors, it should be implemented

a detecting algorithm such as Damm or Luhn algorithm, while to ensure logic integrity a run-time sanity checks needs to be present (such as SQL CHECK constraints). The use of ZFS (or equivalent) should be taken into account to include an extensive protection against data corruption.

### **3.5.4 Maintainability**

Maintainability is a measure of how easily and rapidly a software system can be corrected, improved and adapted to a new environment: its important for the system quality to be concerned about maintainability from the very beginning of the development, later adjustment can be very expensive.

To achieve this quality, the Data4Help System software will avoid high dependencies in the code and use specifics Patterns wherever its appropriate; the development should be iterative and incremental (such as Agile Methodology), Waterfall approach could compromise high maintainability and shouldnt be applied.

### **3.5.5 Portability**

The concern behind porting is to build a program executable and usable over different environments having effort to (re)adapt the software significantly inferior to the effort for new implementation.

The separation between the logic and the interface of the software, that needs to be done from the very beginning of development, is essential to ensure the Data4Help System portability.

Portability can also be affected by the choice of programming language and libraries: language portability its kind of a debated issue, java has a bunch of standard libraries and should be portable and distributed enough. Independence from hardware enhances the Data4Help System portability.

## 4. Formal Analysis Using Alloy

### 4.1 Data4Help

The first alloy model focuses on the lifecycle of data requests performed by TPUs on IUs, on the access from TPUs to approved data, blocked TPUs and group requests namely:

- a request must be created waiting for approval and must then go in a deny or approved state, after which it never changes state, unless it is a subscription request whose approval can be revoked, causing the request to go from an approved state to a denied state;
- a TPU has access only to data for which exists either a stored data request approved after the data was collected or a subscription request approved at the time the data was produced;
- a TPU cannot make a request for data of an individual user if it is blocked;
- a group request is approved only if it targets at least 1000 IUs (in the model is used 3 for a faster execution).

To do so we considered a time frame for which the requests have the set of the state they are in every instant.

The IUs have instead the set of blocked TPUs; this is considered already filled with the TPUs that the IU blocked and never change in the time frame.

The TPUs have a set of data it has access to, after the time frame.

The Group Request have a set of IUs that if the request is valid, it contains more than 3 IUs, else it is empty; for simplicity the info restrictions is modeled with a set of targeted info: if a IU has one of those info he is included in the request target.

#### 4.1.1 Alloy model

---

```
open util/time
```

```
/* Users
```

```
*****
```

```
abstract sig User{
```

```
  id: one Int,
```

```
}
```

```
sig Info{}
```

```
sig ThirdPartyUser extends User{
```

```
  accessibleData: set Data /*The data accessible from the TPUs is the data  
    is has access to after the time frame.*/
```

```
}
```

```
sig IndividualUser extends User{
```

```
  blockedTPUs: set ThirdPartyUser, /*The data accessible from the TPUs is  
    the data is has access to after the time frame.*/
```

```

    info: one Info
}
sig Data{
    user: one IndividualUser,
    time: one Time
}

/* Request
*****
abstract sig DataRequest{
    thirdPartyUser: one ThirdPartyUser,
    individualUser: one IndividualUser,
    stateSet: set RequestState -> Time //The requests contain the set of
        state they were in the time frame considered.
}
sig StoredDataRequest extends DataRequest{}
sig SubscriptionRequest extends DataRequest{}
enum RequestState{W, A, D}
//WAITING, APPROVED, DENYED

/*A group request is either approved or not, independently from time*/
sig GroupRequest{
    thirdPartyUser: one ThirdPartyUser,
    infoSet: set Info, /*restriction on info, for simplicity we model them
        as a set of info a IU is required to have to be part of the target of
        the request */
    iUserSet: set IndividualUser, /*set of anonym IUs compatible with the
        request*/
}{
    #(infoSet) > 0
}

/* Unicity facts
*****
fact idUnique{
    no disjoint u1, u2: User | u1.id = u2.id
}

/*Data requests cycle facts
*****
fact dataRequestState{
    /*A state is defined for every time instant*/
    all dr: DataRequest | all t: Time | one state: RequestState |
        state in dr.stateSet.t
    /*Created in "Waiting"*/
    all dr: DataRequest | all t: Time | some t0: Time |
        gte[t, t0] && dr.stateSet.t0 = W
    /*When the request is "Waiting", it has never changed state*/
    all dr: DataRequest | all t1, t2: Time |
        (gte[t2, t1] && dr.stateSet.t2 = W) => (dr.stateSet.t1 = W)
    /*Once the request is "Denied" it can't change state again*/
    all dr: DataRequest | all t1, t2: Time |
        (gte[t2, t1] && dr.stateSet.t1 = D) => (dr.stateSet.t2 = D)
}/*Allowed state sequences: {W -> D}{W -> A}{W -> A -> D}*/

fact storedDataRequest{

```

```

    /*Once the request is "Accepted" it can't change state again*/
    all sdr: StoredDataRequest | all t1, t2: Time |
        (gte[t2, t1] && sdr.stateSet.t1 = A) => (sdr.stateSet.t2 = A)
}/*Restricted allowed state sequences to: {W -> D}{W -> A}*/

/*Data access facts
*****
fact accessibleData{
    all tpu: ThirdPartyUser | all data: Data |
        /*Data regarding time t is accessible if*/
        (data in tpu.accessibleData) <=>
        /*exist a stored data request approved after time t*/
        ((some sdr: StoredDataRequest | some t: Time |
            (sdr.individualUser = data.user) &&
            (sdr.thirdPartyUser = tpu) &&
            (sdr.stateSet.t = A) &&
            (sdr.stateSet.(data.time) = W)
        )
        ||
        /*exist a subscription request approved at time t*/
        ((some sr: SubscriptionRequest |
            (sr.individualUser = data.user) &&
            (sr.thirdPartyUser = tpu) &&
            (sr.stateSet.(data.time) = A)
        )
        ))
}

/*For each individual user all data exists*/
fact completeData{
    all us: IndividualUser | all t: Time | one data: Data | data.user = us
    && data.time = t
}

/*Group request facts
*****
fact anonymizedUsers{
    all gr: GroupRequest | all iu: IndividualUser |
        /*If there are enough users, all the compatible IUs are in the iUserSet*/
        (((iu.info in gr.infoSet <=> iu in gr.iUserSet) => (#(gr.iUserSet) > 2))
        => (iu.info in gr.infoSet <=> iu in gr.iUserSet)) &&
        /*Else iUserSet is empty*/
        ((not ((iu.info in gr.infoSet <=> iu in gr.iUserSet) => (#(gr.iUserSet)
        > 2))) => (#(gr.iUserSet) = 0))
}

/*Data request cycle pred
*****
pred makeDataRequest[iu: IndividualUser, tpu: ThirdPartyUser, t: Time]{
    //pre-conditions
    not isBlocked[iu, tpu]
    //post-conditions
    one dr: DataRequest | dr.thirdPartyUser = tpu && dr.individualUser = iu
    && dr.stateSet.(t.next) = W
}
pred isBlocked[iu: IndividualUser, tpu: ThirdPartyUser]{
    tpu in iu.blockedTPUs
}

```



```

}
pred approveDataRequest[tpu: ThirdPartyUser, dr: DataRequest, t: Time]{
  //pre-conditions
  dr.stateSet.t = W
  //post-conditions
  dr.stateSet.(t.next) = A
}
pred denyStoredDataRequest[tpu: ThirdPartyUser, sdr: StoredDataRequest, t:
  Time]{
  //pre-conditions
  sdr.stateSet.t = W
  //post-conditions
  sdr.stateSet.(t.next) = D
}
pred denySubscriptionApproval[tpu: ThirdPartyUser, sr:
  SubscriptionRequest, t: Time]{
  //pre-conditions
  not sr.stateSet.t = D
  //post-conditions
  sr.stateSet.(t.next) = D
}

/*Group request pred
*****
pred isValid[gr: GroupRequest]{
  #(gr.iUserSet) > 0
}

/*Show
*****

pred show1{
  #(ThirdPartyUser) = 1
  #(IndividualUser) = 1
  #(StoredDataRequest) = 1
  #(SubscriptionRequest) = 1
  #(GroupRequest) = 0
  /*At least one tpu has access to some data*/
  (some tpu: ThirdPartyUser | #(tpu.accessibleData) > 0)
  (some iu: IndividualUser | some tpu: ThirdPartyUser | some t: Time |
    makeDataRequest[iu, tpu, t])
  (some dr: DataRequest | some tpu: ThirdPartyUser | some t: Time |
    approveDataRequest[tpu, dr, t])
  (some sdr: StoredDataRequest | some tpu: ThirdPartyUser | some t: Time |
    denyStoredDataRequest[tpu, sdr, t])
  (some sr: SubscriptionRequest | some tpu: ThirdPartyUser | some t: Time
    | denySubscriptionApproval[tpu, sr, t])
}

pred show2{
  #(ThirdPartyUser) = 1
  #(IndividualUser) = 4
  #(GroupRequest) = 2
  (some gr: GroupRequest | isValid[gr])
  (some gr: GroupRequest | not isValid[gr])
}

```

```
/*Run
```

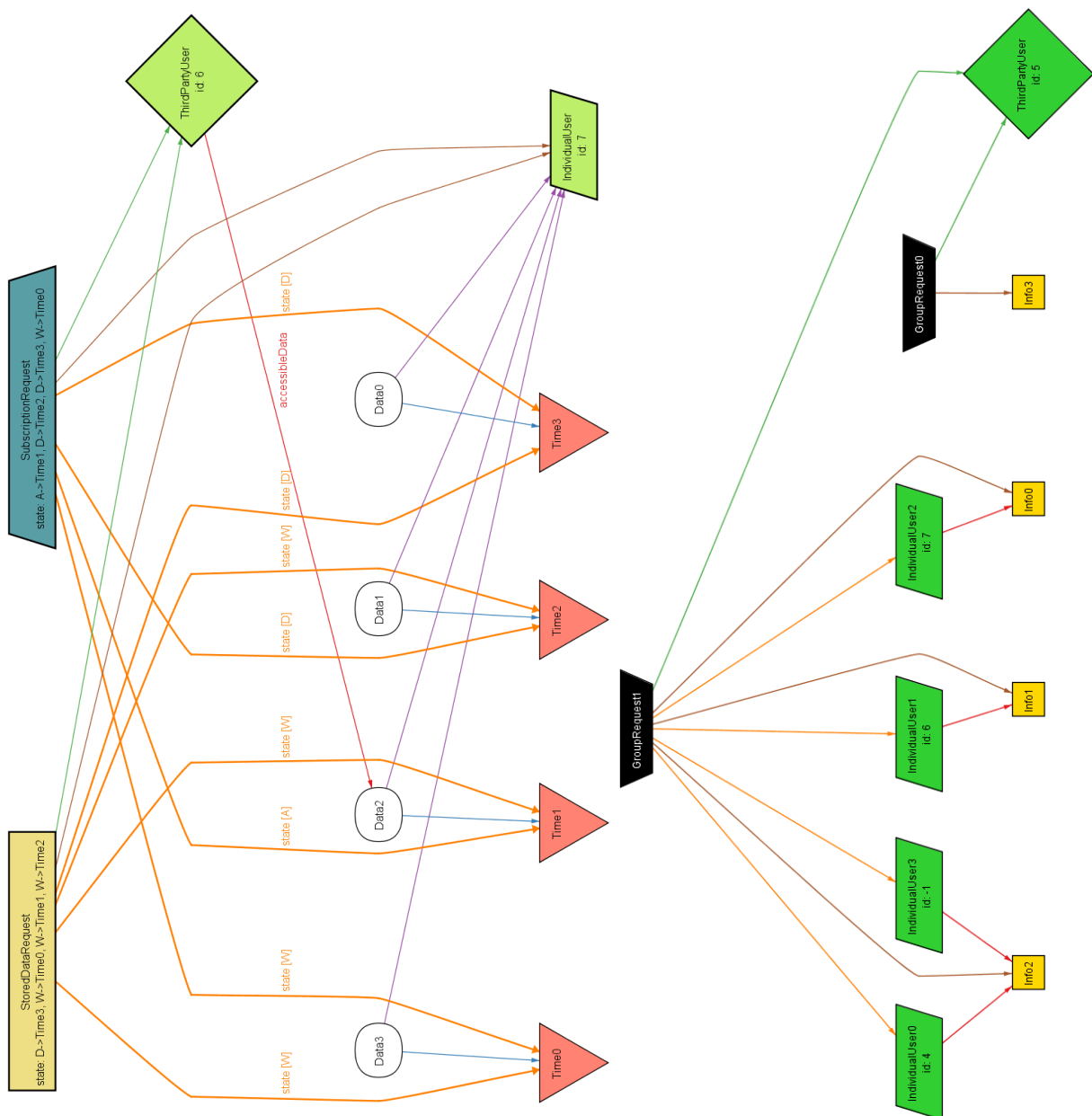
```
*****
```

```
run makeDataRequest for 10  
run approveDataRequest for 10  
run denyStoredDataRequest for 10  
run denySubscriptionApproval for 10
```

```
run show1 for 8 but exactly 4 Time  
run show2 for 5 but exactly 1 Time
```

---

### 4.1.2 Generated world



### 4.1.3 Alloy Result

Option **Font size** changed to **20**

#### Executing "Run makeDataRequest for 10"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
44844 vars. 1560 primary vars. 135226 clauses. 881ms.

**Instance** found. Predicate is consistent. 692ms.

#### Executing "Run approveDataRequest for 10"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
44291 vars. 1560 primary vars. 133104 clauses. 485ms.

**Instance** found. Predicate is consistent. 169ms.

#### Executing "Run denyStoredDataRequest for 10"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
44291 vars. 1560 primary vars. 133104 clauses. 456ms.

**Instance** found. Predicate is consistent. 332ms.

#### Executing "Run denySubscriptionApproval for 10"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
44291 vars. 1560 primary vars. 133293 clauses. 451ms.

**Instance** found. Predicate is consistent. 270ms.

#### Executing "Run show1 for 8 but exactly 4 Time"

Solver=sat4j Bitwidth=4 MaxSeq=7 SkolemDepth=1 Symmetry=20  
20460 vars. 976 primary vars. 55123 clauses. 142ms.

**Instance** found. Predicate is consistent. 102ms.

#### Executing "Run show2 for 5 but exactly 1 Time"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
5386 vars. 370 primary vars. 12620 clauses. 37ms.

**Instance** found. Predicate is consistent. 25ms.

#### 6 commands were executed. The results are:

- #1: **Instance found.** makeDataRequest is consistent.
- #2: **Instance found.** approveDataRequest is consistent.
- #3: **Instance found.** denyStoredDataRequest is consistent.
- #4: **Instance found.** denySubscriptionApproval is consistent.
- #5: **Instance found.** show1 is consistent.
- #6: **Instance found.** show2 is consistent.

## 4.2 AutomatedSOS

The second alloy model focus on the lifecycle of an emergency call:

- a call must end after it has contacted the ES or after it was blocked by the IU
- for every time interval in which the health was critical exactly one call must have been made

For simplicity sake, this model considers that an ambulance is called every time the status drops to critical.

### 4.2.1 Alloy model

---

`open util/time`

```
/* Signatures
*****/
abstract sig User{
  id: one Int,
  locationSet: set Location -> Time,
  dataSet: set Data -> Time,
  referenceData: set Data /*Set of data for critical health condition*/
}{
  #(referenceData) > 0
  all t: Time | one data: Data | dataSet.t = data
}
sig Data{}
sig Location{}

sig EmergencyCall{
  user: one User,
  location: one Location,
  callTime: one Time,
  stateSet: set CallState one -> Time
}{
  location = user.locationSet.callTime
  all t: Time | one cs: CallState | stateSet.t = cs
}
enum CallState{N, C, B, D}
//NOT_STARTED, CALLING, BLOCKED, DONE

/* Facts
*****/
fact idUnique{
  no disjoint u1, u2: User | u1.id = u2.id
}

fact callStates{
  /*A state is defined for every time instant after callTime*/
  all ec: EmergencyCall | all t: Time | one state: CallState |
    state in ec.stateSet.t
  /*Created in "Calling"*/
  all ec: EmergencyCall |
    (ec.stateSet.(ec.callTime) = C) && (no t: Time | (lt[t, ec.callTime])
    && (ec.stateSet.t = C))
  /*When the call is "not started", it has never changed state*/
}
```

```

all ec: EmergencyCall | all t1, t2: Time |
  (gte[t2, t1] && ec.stateSet.t2 = N) => (ec.stateSet.t1 = N)
/*Once the call is "Blocked" it can't change state again*/
all ec: EmergencyCall | all t1, t2: Time |
  (gte[t2, t1] && ec.stateSet.t1 = B) => (ec.stateSet.t2 = B)
/*Once the call is "Done" it can't change state again*/
all ec: EmergencyCall | all t1, t2: Time |
  (gte[t2, t1] && ec.stateSet.t1 = D) => (ec.stateSet.t2 = D)

/*Allowed state sequences: {N -> C -> B}{N -> C -> D}*/

/*For every instant in wich the user health went critical a call has been
made when health dropped*/
fact whenCriticalThereIsACall{
  all us: User | all t: Time |
    (us.dataSet.t in us.referenceData) => (
  one ec: EmergencyCall | all t1: Time |
    (ec.user = us) &&
    (lte[ec.callTime, t]) &&
    ((gte[t1, ec.callTime] && lte[t1, t]) => (us.dataSet.t1 in
      us.referenceData))
    )
}

/*No calls are made when health is not critical*/
fact {
  all ec: EmergencyCall | one us: User |
    (ec.user = us) &&
    (us.dataSet.(ec.callTime) in us.referenceData)
}

/* Predicates
******/
pred callAmbulance[us: User, t: Time]{
  //pre-conditions
  //post-conditions
  one ec: EmergencyCall |
    (ec.user = us) &&
    (ec.callTime = t.next) &&
    (ec.stateSet.(t.next) = C)
}

pred completeCall[ec: EmergencyCall, t: Time]{
  //pre-conditions
  ec.stateSet.t = C
  //post-conditions
  ec.stateSet.(t.next) = D
}

pred blockCall[ec: EmergencyCall, t: Time]{
  //pre-conditions
  ec.stateSet.t = C
  //post-conditions
  ec.stateSet.(t.next) = B
}

pred show {

```

```

    #(User) = 2
    #(EmergencyCall) = 2
    (all us: User | some t: Time | callAmbulance[us, t])
    (some ec: EmergencyCall | some t: Time | completeCall[ec, t])
    (some ec: EmergencyCall | some t: Time | blockCall[ec, t])
}

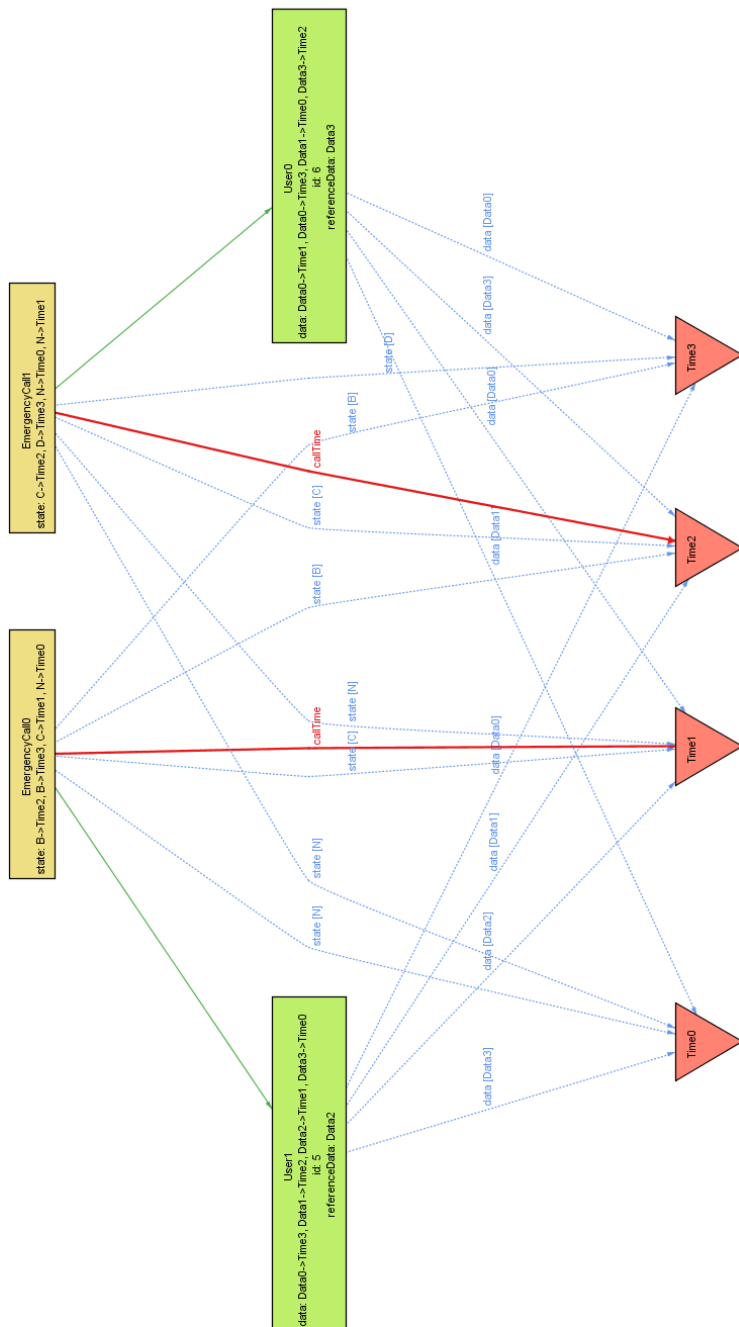
/* Run
*****/
run callAmbulance for 5
run completeCall for 5
run blockCall for 5

run show for 5

```

---

### 4.2.2 Generated world





#### 4.2.3 Alloy Result

##### Executing "Run callAmbulance for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
8959 vars. 560 primary vars. 22458 clauses. 61ms.  
**Instance** found. Predicate is consistent. 63ms.

##### Executing "Run completeCall for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
8874 vars. 560 primary vars. 21979 clauses. 78ms.  
**Instance** found. Predicate is consistent. 131ms.

##### Executing "Run blockCall for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
8874 vars. 560 primary vars. 21979 clauses. 82ms.  
**Instance** found. Predicate is consistent. 57ms.

##### Executing "Run show for 4"

Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20  
5721 vars. 368 primary vars. 13626 clauses. 41ms.  
**Instance** found. Predicate is consistent. 36ms.

##### 4 commands were executed. The results are:

- #1: **Instance found.** callAmbulance is consistent.
- #2: **Instance found.** completeCall is consistent.
- #3: **Instance found.** blockCall is consistent.
- #4: **Instance found.** show is consistent.

## 4.3 Track4Run

The third alloy model focus on the lifecycle of a run and the restriction on participants:

- every organized run must either be canceled before they start or successfully end;
- IUs can enroll to participate or spectate only during specific state of the run, respectively when the run has been organized but not started and when the run has started

### Alloy model

---

```
open util/time

/* Users
*****/
abstract sig User{
  id: one Int
}
abstract sig ThirdPartyUser extends User{}
abstract sig IndividualUser extends User{
  info: one Info
}
sig Info{}

sig Organizer extends ThirdPartyUser{}

sig Participant extends IndividualUser{}
sig Spectator extends IndividualUser{}

sig Run{
  organizer: one Organizer,
  stateSet: RunState one -> Time, /*state of the run for every instant in
    the time frame considered*/
  /*Users who spectated the run*/
  spectatorSet: set Spectator,
  /*Users who are participants at the end of the time frame considered*/
  participantSet: set Participant,
  /*These two set represent the number of participants and spactators for
    every instant in the time frame considered*/
  participantNum: set Int one -> Time,
  spectatorNum: set Int one -> Time,
  restrictionSet: set Info //restriction are modeled as a set of not
    allowed info
}{
  all t: Time | one i: Int | participantNum.t = i
  all t: Time | one i: Int | spectatorNum.t = i
  /*at the end of the time frame the number of participant is equal to
    participantNum*/
  all t: Time | (no t0: Time | gt[t0, t]) => participantNum.t =
    #(participantSet)
}
enum RunState{N, O, S, E, C}
//NOT_CREATED, ORGANIZED, STARTED, ENDED, CANCELED

/* Unicity facts
*****/
```

```

fact idUnique{
  no disjoint u1, u2: User | u1.id = u2.id
}

/* Run states facts
*****/
fact runStates{
  /*A state is defined for every time instant after callTime*/
  all r: Run | all t: Time | one state: RunState |
    state in r.stateSet.t
  /*A run must have been in the "Non created" state at least once*/
  all r: Run | some t: Time | r.stateSet.t = N
  /*When the run is "Not created" yet, it has never changed state */
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t2 = N) => (r.stateSet.t1 = N)
  /*If "Organized" has always been "Not started" or "Organized"*/
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t2 = O) => (r.stateSet.t1 = N ||
      r.stateSet.t1 = O)
  /*If "Started" will "End"*/
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t1 = S) => (r.stateSet.t1 = S ||
      r.stateSet.t1 = E)
  /*Once the run is "Ended" it can't change state again*/
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t1 = E) => (r.stateSet.t2 = E)
  /*If "Canceled" has never "Started"*/
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t2 = C) => (not r.stateSet.t1 = S)
  /*Once the run is "Canceled" it can't change state again*/
  all r: Run | all t1, t2: Time |
    (gte[t2, t1] && r.stateSet.t1 = C) => (r.stateSet.t2 = C)

}/*Allowed state sequences: {N -> O -> S -> E}{N -> O -> C}*/

/* Joining facts
*****/
fact openEnroll{
  /*If run is not in the "Organize" state none can enroll*/
  all r: Run | all t1, t2: Time |
    (not(r.stateSet.t1 = N) && not (r.stateSet.t1 = O) && not(r.stateSet.t2
      = N) && not (r.stateSet.t2 = O))
    =>
    (r.participantNum.t1 = r.participantNum.t2)

  /*If run is in the "Not Started" nobody enrollled*/
  all r: Run | all t: Time |
    (r.stateSet.t = N)
    =>
    (r.participantNum.t = 0)

  /*If run is in not in the "Started" nobody is spectating*/
  all r: Run | all t: Time |
    (not (r.stateSet.t = S))
    =>
    (r.spectatorNum.t = 0)

```

```

}

/* Run state pred
*****/
pred organize[o: Organizer, t: Time]{
    one r: Run |
        (r.stateSet.t = N) &&
        (r.organizer = o) &&

        (r.stateSet.(t.next) = O) &&
        (r.participantNum.(t.next) = 0) &&
        (r.spectatorNum.(t.next) = 0)
}

pred start[r: Run, t: Time]{
    //pre-conditions
    (r.stateSet.t = O)
    //post-conditions
    (r.stateSet.(t.next) = S)
}

pred end[r: Run, t: Time]{
    //pre-conditions
    (r.stateSet.t = S)
    //post-conditions
    (r.stateSet.(t.next) = E)
}

pred cancel[r: Run, t: Time]{
    //pre-conditions
    (r.stateSet.t = O)
    //post-conditions
    (r.stateSet.(t.next) = C) &&
    (r.spectatorNum.(t.next) = 0)
}

/* Enroll Run pred
*****/
pred enroll[p: Participant, r: Run, t: Time]{
    not (p.info in r.restrictionSet)
    r.participantNum.(t.next) = r.participantNum.t + 1
    p in r.participantSet
}

pred spectate[s: Spectator, r: Run, t: Time]{
    r.spectatorNum.(t.next) = r.spectatorNum.t + 1
    s in r.spectatorSet
}

/* Show
*****/
pred show{
    #(ThirdPartyUser) = 1
    #(IndividualUser) = 2
    #(Info) = 3
    #(Run) = 2
    (some o: Organizer | some t: Time | organize[o, t])
}

```

```

    (some r: Run | some t: Time | start[r, t])
    (some r: Run | some t: Time | end[r, t])
    (some r: Run | some t: Time | cancel[r, t])
    (some r: Run | some t: Time | some p: Participant | enroll[p, r, t])
    (some r: Run | some t: Time | some s: Spectator | spectate[s, r, t])
}

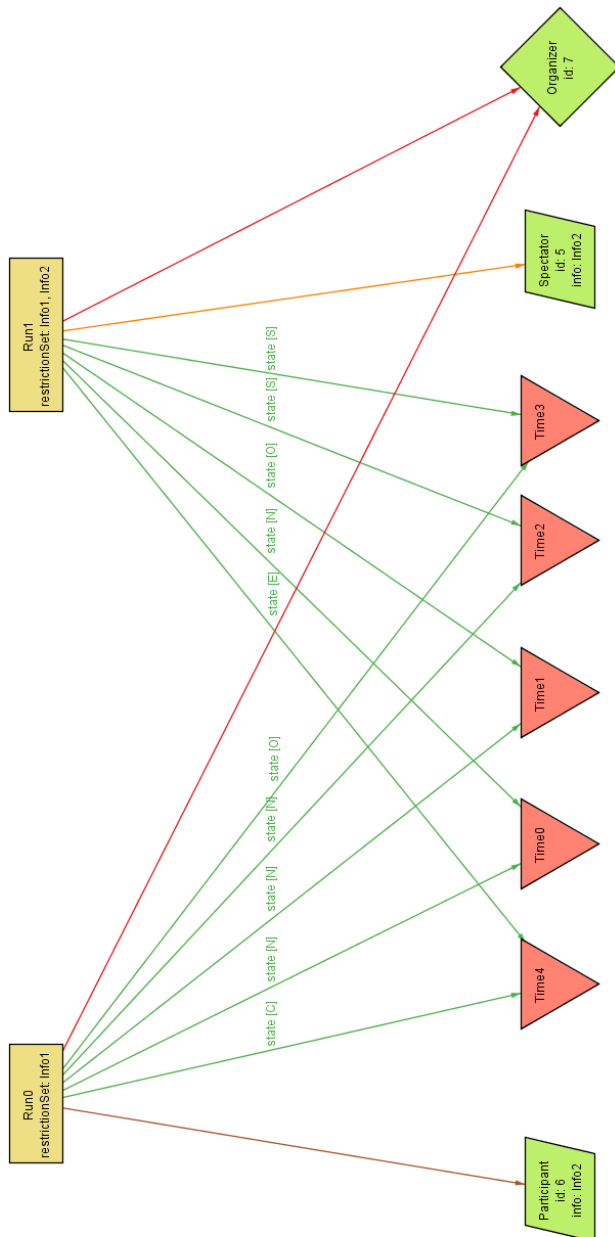
/* Run
*****/
run organize for 5
run start for 5
run end for 5
run cancel for 5

run show for 5

```

---

### 4.3.1 Generated world



#### 4.3.2 Alloy Result

##### Executing "Run organize for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
13069 vars. 1190 primary vars. 38595 clauses. 115ms.

**Instance found.** Predicate is consistent. 52ms.

##### Executing "Run start for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
12065 vars. 1190 primary vars. 34781 clauses. 85ms.

**Instance found.** Predicate is consistent. 31ms.

##### Executing "Run end for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
12065 vars. 1190 primary vars. 34781 clauses. 81ms.

**Instance found.** Predicate is consistent. 36ms.

##### Executing "Run cancel for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
12628 vars. 1190 primary vars. 35573 clauses. 125ms.

**Instance found.** Predicate is consistent. 81ms.

##### Executing "Run show for 5"

Solver=sat4j Bitwidth=4 MaxSeq=5 SkolemDepth=1 Symmetry=20  
16040 vars. 1250 primary vars. 46040 clauses. 127ms.

**Instance found.** Predicate is consistent. 59ms.

##### 5 commands were executed. The results are:

#1: **Instance found.** organize is consistent.

#2: **Instance found.** start is consistent.

#3: **Instance found.** end is consistent.

#4: **Instance found.** cancel is consistent.

#5: **Instance found.** show is consistent.

## 5. Effort Spent

### 5.1 ARGIRO' ANNA SOFIA

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
14/10/18	analysis and definition of the given problem	3
19/10/18	description of the scope, definition of the given problem	2
21/10/18	definition of the scope - AutomatedSOS, definition, acronymism abbreviation about it	1.5
22/10/18	product perspective, product function - AutomatedSOS	2
23/10/18	user characteristics, assumption, dependencies and constraints - Automated SOS	3
26/10/18	software interfaces	2
29/10/18	software interfaces review, communication interfaces	1
30/10/18	communication interfaces, assumptions review	1
2/11/18	reliability, performance requirements	3
3/11/18	availability	2
5/11/18	availability review, security	2.5
7/11/18	security	1.5
8/11/18	maintainability	1
10/11/18	portability, introduction, general group revision	7
11/11/18	general group revision	5
<b>SUM</b>		<b>37,5</b>



## 5.2 BATTAGLIA GABRIELE

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
14/10/18	analysis and definition of the given problem	3
19/10/18	description of the scope and definition of the given problem	2
23/10/18	definition of the scope - Data4Help	2.5
24/10/18	product prespective - Data4Help	1
25/10/18	product prespective review, product function, user characteristics, assumption, dependencies and constraints - Data4Help	2
26/10/18	overall desctription review - Data4Help	1
29/10/18	class diagrams and statecharts review	2
1/11/18	functional requirements - use case definition	2
2/11/18	functional requirements - use case definition	1
3/11/18	performance requirements, formal analysis using alloy	3
4/11/18	formal analysis using alloy	4
3/11/18	formal analysis using alloy	4
9/11/18	formal analysis using alloy	2
10/11/18	general group revision, introduction	6
11/11/18	general group revision	4
<b>SUM</b>		<b>39,5</b>

### 5.3 CASASOLE BERNARDO

DATE	DESCRIPTION OF THE TASK	HOURS SPENT
14/10/18	analysis and definition of the given problem	3
19/10/18	description of the scope and definition of the given problem	2
21/10/18	product prespective - Track4Run	1.5
25/10/18	product functions - Track4Run	1
26/10/18	user characteristics and assumption, dependencies and constraints - Track4Run	2
29/10/18	User interfaces	3
30/10/18	User interfaces	3.5
1/11/18	User interfaces, hardware interfaces	3.5
2/11/18	Design constraints - standard compliance, User interfaces review	1
3/11/18	Design constrains review, sequence diagrams	1.5
6/11/18	sequence diagrams review, functional requirements review	2
10/11/18	general group revision, introduction, effort spent	7
11/11/18	general group revision	4
<b>SUM</b>		<b>35</b>

## 6. References

### 6.1 Reference Documents

#### Past courses material

- RASD Sample from A.Y. 2015-2016.pdf
- RASD Sample from A.Y. 2016-2017.pdf

#### Standards compliance

- <https://www.w3.org/>
- <https://en.wikipedia.org/wiki/Standards-compliant>
- <https://validator.w3.org/>
- [http://www.thepollenshop.co.uk/accessible\\_web\\_services/standards\\_compliance/](http://www.thepollenshop.co.uk/accessible_web_services/standards_compliance/)
- <https://www.polemicdigital.com/w3c-compliance-is-it-a-requirement/>

#### GPS Hardware components

- [www.eisplc.com](http://www.eisplc.com)

#### Hardware constraints brief

- <https://github.com/marcobuschini/Software-Requirements-Specification/blob/master/External%20Interface%20Re>

#### Real time data

- <https://www.techadvisor.co.uk/how-to/game/how-to-lower-ping-3609346/>
- <http://www.plugthingsin.com/internet/speed/latency/>
- <https://www.techadvisor.co.uk/how-to/game/how-to-lower-ping-3609346/>

#### RT application

- <https://searchunifiedcommunications.techtarget.com/definition/real-time-application-RTA>

#### SSAttributes

- <https://sites.google.com/site/misresearch000/home/software-architecture-quality-attributes>
- Course slides on Modeling Requirements.pdf

## **Portability**

- <https://stackoverflow.com/questions/3925947/what-is-portability-how-is-java-more-portable-than-other-languages>
- <https://softwareengineering.stackexchange.com/questions/111080/why-is-java-considered-more-portable-than-other-languages-like-c>
- [https://en.wikipedia.org/wiki/Software\\_portability](https://en.wikipedia.org/wiki/Software_portability)
- <http://www.c4learn.com/java/why-java-is-platform-independent-and-portable/>
- [https://link.springer.com/content/pdf/10.1007%2F1-4020-8159-6\\_3.pdf](https://link.springer.com/content/pdf/10.1007%2F1-4020-8159-6_3.pdf)
- <https://en.wikipedia.org/wiki/Porting>

## **Maintain and develop**

- <https://blog.hyperiondev.com/index.php/2017/09/26/types-of-software-development/>
- <http://www.itinfo.am/eng/software-development-methodologies/>
- <https://software.ac.uk/resources/guides/developing-maintainable-software>
- [https://en.wikipedia.org/wiki/Software\\_maintenance](https://en.wikipedia.org/wiki/Software_maintenance)
- <https://en.wikipedia.org/wiki/Maintainability>
- <https://ieeexplore.ieee.org/document/5221065>
- <https://bobbelderbos.com/2016/03/building-maintainable-software/>
- [http://www.testingstandards.co.uk/maintainability\\_guidelines.htm](http://www.testingstandards.co.uk/maintainability_guidelines.htm)

## **Data integrity**

- <https://www.globalvisioninc.com/blog/12-ways-to-reduce-data-integrity-risk/>
- [https://en.wikipedia.org/wiki/Data\\_integrity](https://en.wikipedia.org/wiki/Data_integrity)
- <https://explorance.com/blog/3-strategies-to-maintain-data-integrity/>
- [https://en.wikipedia.org/wiki/Data\\_corruption#END-TO-END-PROTECTION](https://en.wikipedia.org/wiki/Data_corruption#END-TO-END-PROTECTION)
- <https://en.wikipedia.org/wiki/ZFS>
- <https://en.wikipedia.org/wiki/RAID>
- [https://en.wikipedia.org/wiki/Check\\_constraint](https://en.wikipedia.org/wiki/Check_constraint)
- [https://en.wikipedia.org/wiki/ECC\\_memory](https://en.wikipedia.org/wiki/ECC_memory)

## **Software interfaces**

- <https://developers.google.com/maps/documentation/javascript/datalayer>
- <https://www.postgresql.org>

## **Security**

- [https://en.wikipedia.org/wiki/Transport\\_Layer\\_Security](https://en.wikipedia.org/wiki/Transport_Layer_Security)
- <https://f5.com/we-make-apps-go/safer/how-to-ensure-the-availability-integrity-and-confidentiality-of-your-apps>

## **Confidentiality**

- [https://developer.mozilla.org/en-US/docs/Web/Security/Information\\_Security\\_Basics/Confidentiality,\\_Integrity,\\_and\\_Availability](https://developer.mozilla.org/en-US/docs/Web/Security/Information_Security_Basics/Confidentiality,_Integrity,_and_Availability)
- <https://www.techopedia.com/definition/10254/confidentiality>

## **Availability**

- [https://en.wikipedia.org/wiki/Mean\\_time\\_between\\_failures](https://en.wikipedia.org/wiki/Mean_time_between_failures)
- <https://www.techopedia.com/definition/8281/mean-time-to-failure-mttf>
- <http://sai.syr.edu/~chapin/cis583/RelSpecs.pdf>

## **Reliability**

- [https://en.wikipedia.org/wiki/Software\\_reliability\\_testing](https://en.wikipedia.org/wiki/Software_reliability_testing)
- <https://www.unf.edu/~ncoulter/cen6070/handouts/minorreport/Rosenberg.html>
- [https://www.tutorialspoint.com/software\\_quality\\_management/software\\_quality\\_management\\_metrics.htm](https://www.tutorialspoint.com/software_quality_management/software_quality_management_metrics.htm)
- [https://www.researchgate.net/publication/236845436\\_Metrics\\_models\\_and\\_measurements\\_in\\_software\\_reliability](https://www.researchgate.net/publication/236845436_Metrics_models_and_measurements_in_software_reliability)
- [https://en.wikipedia.org/wiki/List\\_of\\_software\\_reliability\\_models](https://en.wikipedia.org/wiki/List_of_software_reliability_models)

## **Communication Interface**

- [https://en.wikipedia.org/wiki/Client%E2%80%93server\\_model](https://en.wikipedia.org/wiki/Client%E2%80%93server_model)
- <https://www.quora.com/How-does-an-Android-app-communicate-with-its-servers>
- <https://security.stackexchange.com/questions/76434/web-server-interaction-with-a-database-server>

## **6.2 Software**

- TeXWorks v0.6.2
- AlloyAnalyzer v4.2
- Umlet v14.2
- Draw.io v9.4.1
- proto.io v6.3.2.3