

TrackMe project L. Barilani, W. Bonvini,
L. Carnaghi



POLITECNICO
MILANO 1863

Requirement Analysis and Specification Document

Deliverable:	RASD
Title:	Requirement Analysis and Verification Document
Authors:	Leonardo Barilani, William Bonvini, Lorenzo Carnaghi
Version:	1.0
Date:	8-November-2018
Download page:	https://github.com/Lockyard/BarilaniBonviniCarnaghi
Copyright:	Copyright © 2018, Leonardo Barilani, William Bonvini, Lorenzo Carnaghi – All rights reserved

Contents

Table of Contents	3
1 Introduction	5
1.1 Purpose	5
1.1.1 Goals	5
1.2 Scope	5
1.2.1 Project description	5
1.3 Definitions, Acronyms, Abbreviations	6
1.3.1 Definitions	6
1.3.2 Acronyms	6
1.3.3 Abbreviations	6
1.4 Revision history	7
2 Overall Description	8
2.1 Product perspective	8
2.2 Product functions	10
2.3 User characteristics	10
2.4 Assumptions, dependencies and constraints	11
3 Specific Requirements	12
3.1 External interface requirements	12
3.1.1 User Interfaces	12
3.1.2 Hardware Interfaces	16
3.1.3 Software Interfaces	16
3.1.4 Communication Interfaces	16
3.2 Functional requirements	17
3.2.1 Use case diagrams	17
3.2.2 Sequence diagrams	22
3.2.3 Mapping on requirements	26
3.3 Performance requirements	26
3.4 Design constraints	27
3.4.1 Standard compliance	27
3.4.2 Hardware limitations	27
3.4.3 Any other constraints	27
3.5 Software system attributes	27
3.5.1 Reliability	27
3.5.2 Availability	27
3.5.3 Security	27
3.5.4 Maintainability	27
3.5.5 Portability	27
3.6 Scenarios	28
3.6.1 Scenario 1 - Data4Help	28
3.6.2 Scenario 2 - Data4Help	28
3.6.3 Scenario 3 - Data4Help	28
3.6.4 Scenario 4 - Data4Help	28
3.6.5 Scenario 5 - Data4Help	28
3.6.6 Scenario 6 - AutomatedSOS	29
3.6.7 Scenario 7 - AutomatedSOS	29
3.6.8 Scenario 8 - Track4Run	29

3.6.9	Scenario 9 - Track4Run	29
3.6.10	Scenario 10 - Track4Run	30
3.6.11	Scenario 11 - Track4Run	30
4	Formal Analysis Using Alloy	31
4.1	Purpose	31
4.1.1	General Description	31
4.1.2	Simplifications	31
4.2	Code	33
4.2.1	Data4Help	33
4.2.2	AutomatedSOS	36
4.2.3	Track4Run	39
4.3	Worlds Generated	42
4.3.1	Data4Help	42
4.3.2	AutomatedSOS	43
4.3.3	Track4Run	44
4.4	Alloy results	45
5	Effort Spent	46
5.1	Leonardo Barilani	46
5.2	William Bonvini	46
5.3	Lorenzo Carnaghi	46

1 Introduction

1.1 Purpose

The purpose of this document is to give an overview of the Data4Help, AutomatedSOS and Track4Run services, and specify its requirements, functional and non functional and associated technical constraints. People who are interested in this service would need to download the relative app on their smartphone, and their data will be automatically collected by TrackMe. Data4Help is a service which aims to gather information about people's health status in order to share it with third-parties. AutomatedSOS aims to help elderly people by constantly monitoring their health status and automatically calling an ambulance in case of an emergency. Track4Run's purpose is to organize races among runners. It allows the creation of a run and its enrollment. Also, it provides a real-time map of the runners for the spectators.

1.1.1 Goals

- G1 - The Data4Help application gives constantly new data accordingly to users' activities.
- G2 - Registered TPs (Third Parties) can send requests to specific users if they know their SSN or CF in order to retrieve specific data. These users can accept or refuse the proposal.
- G3 - Registered TPs have access to make queries to Data4Help's database in order to get anonymized data by filtering by one or more parameters as they need.
- G4 - Results for TPs' queries are given only if the number of matches is greater or equal than 1000.
- G5 - Each user of the services offered by TrackMe is identifiable within the system in order to gather his data and communicate with him.
- G6 - AutomatedSOS sends automatically an ambulance request when and only when user's values are below a specific critical threshold, specifying the user's position.
- G7 - Users of Track4Run are able to create a run defining the precise path.
- G8 - Users of Track4Run are able to take part in a created run as runners.
- G9 - Users of Track4Run can see real-time runners' positions in runs such runners are subscribed in.

1.2 Scope

1.2.1 Project description

Data4Help is a service mainly thought for big companies who want to make better targeted choices for their business. Since the service acquires the health status of the users, most of the companies interested in the service are supposedly going to be related to lifestyle, fitness, and generally the sports field. The position of the users could be a very useful information for third parties related to the transportation system and tourism. Health status and position combined offer a good overview of the health situation of a population of a specific area selected by the third party. Such data can be used for pools and statistical purposes, so being attractive to health related realities: pharmaceutical industry, health organizations (state entities and non state ones). The final product will consist in 3 main softwares:

- The server-side software;
- The mobile application for standard-users;
- The desktop application for third-parties users.

There are quite a few shared phenomena that occur in the projects:

- A third party requests a query in Data4Help;
- One of the three apps gathers information;
- AutomatedSOS calls an ambulance in case of an emergency;
- A run gets created in Track4Run by an user.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- User: a user is a person who uses at least one of Track4Run's services by installing it on its smartphone/PC and signing up.
- Third-party: any organization/company/authority who registers to Data4Help in order to gather user's information.
- Query: A filtered search asked to the Data4Help's DB.
- Run: an event organized through Track4Run service, in a specific location.

1.3.2 Acronyms

- TP: Third Party (TPs for plural)
- SSN: Social Security Number
- CF: Codice Fiscale
- DB: Database
- ASOS: AutomatedSOS

1.3.3 Abbreviations

- [G_x]: x-th Goal
- [D_x]: x-th Domain assumption
- [R_x]: x-th Requirement
- [FR_x]: x-th Functional Requirement

1.4 Revision history

the version 2 of the RASD present the following changes:

- removed chat system reference in mockups and use cases
- Fixed english mistakes and typos
- Modified Alloy model for Track4Run: It no more shows users subscribed as participants or spectators but instead there is a new information on which participants are visible in a certain time. The predicates and assertions partially changed, to show a more coherent aspect.
- Added Data4Help's Third party's mockups
- Added the class TrackMe in the UML diagram

2 Overall Description

2.1 Product perspective

The following class diagram sums up conceptually the system behind the services Data4Help, AutomatedSOS and Track4Run.

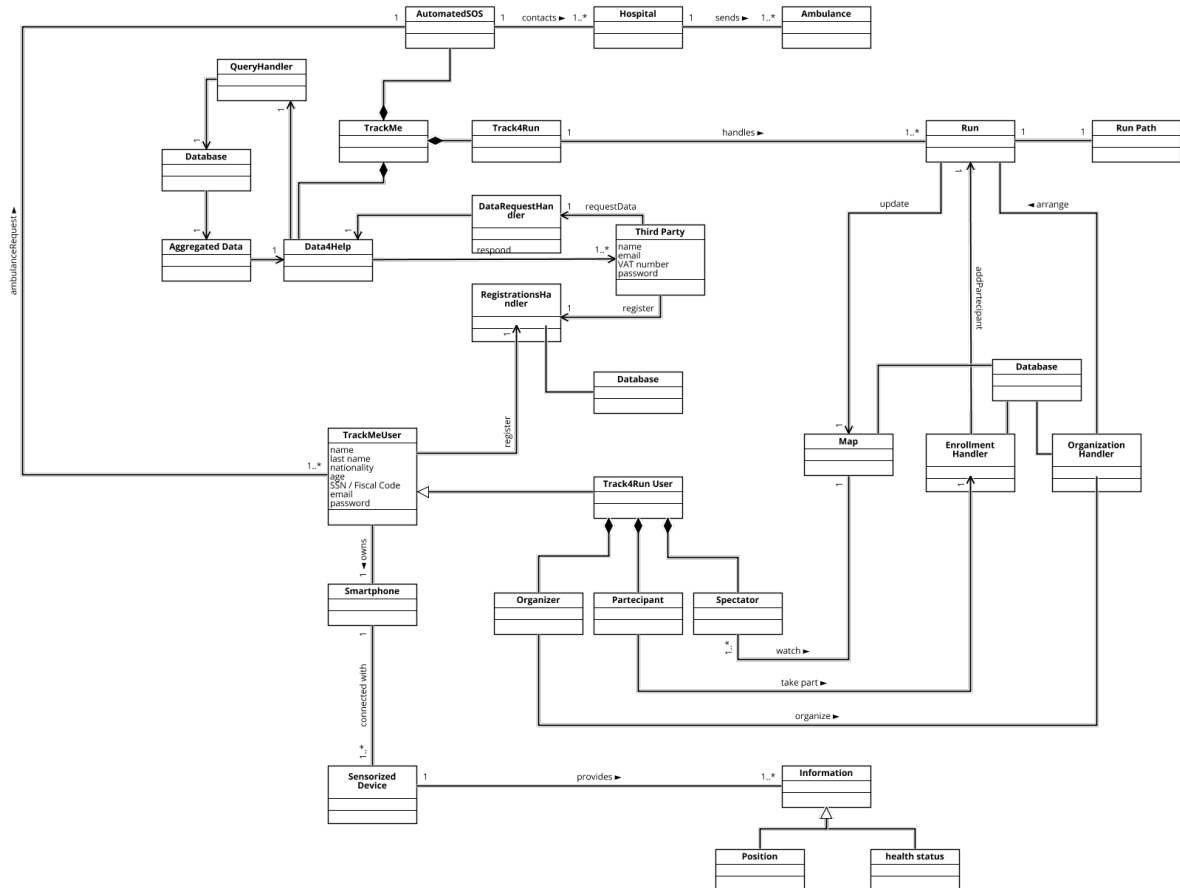


Figure 1: UML diagram

From this diagram it's visible that for the three services only one TrackMe account is needed to be created. The class TrackMe is mainly a container of the three services. Database is only one class, it gets repeated more than once only to make the diagram clearer to be looked at.

Here will follow the state diagrams of each of the services provided by TrackMe.

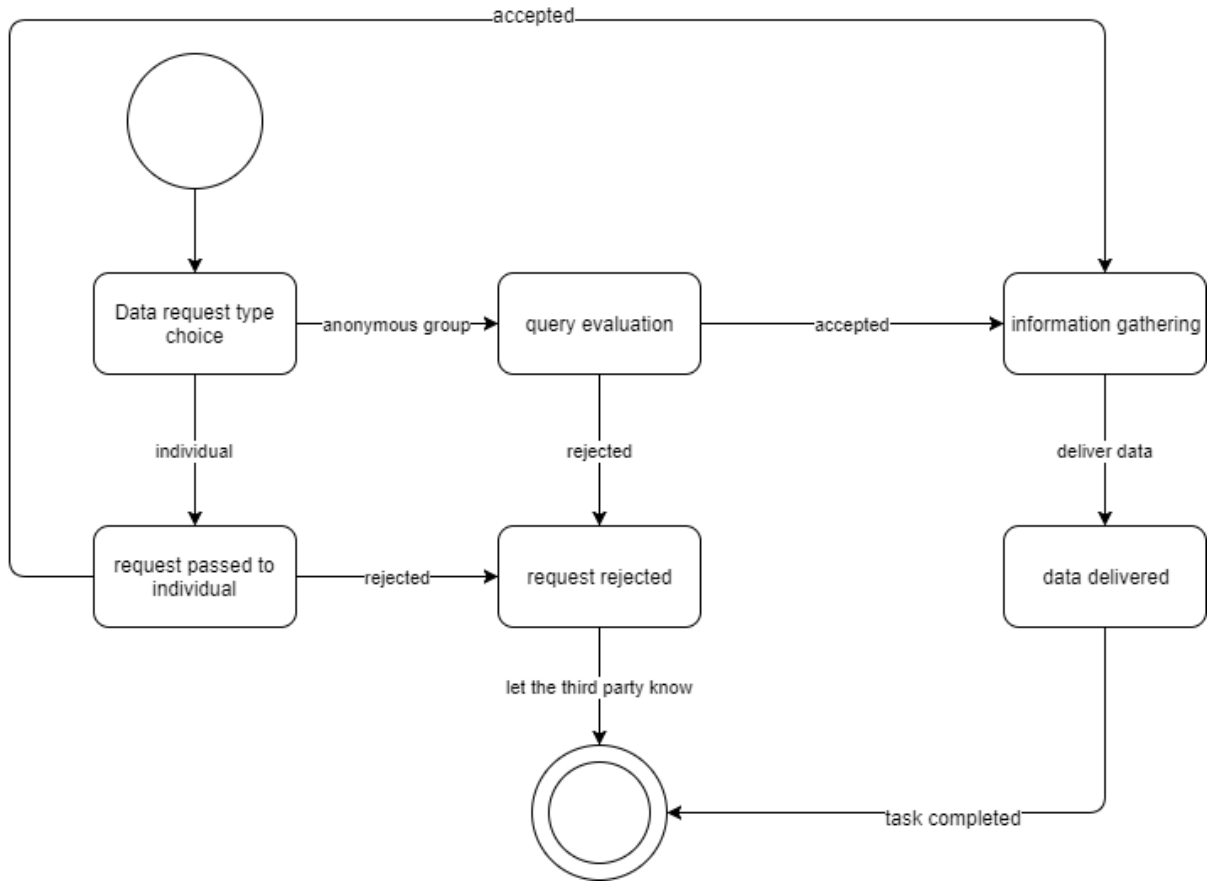


Figure 2: Data4Help state diagram

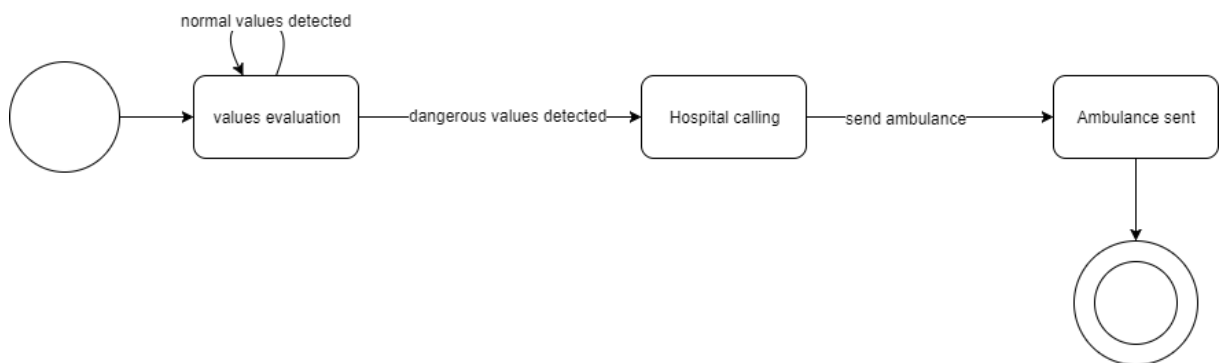


Figure 3: AutomatedSOS state diagram

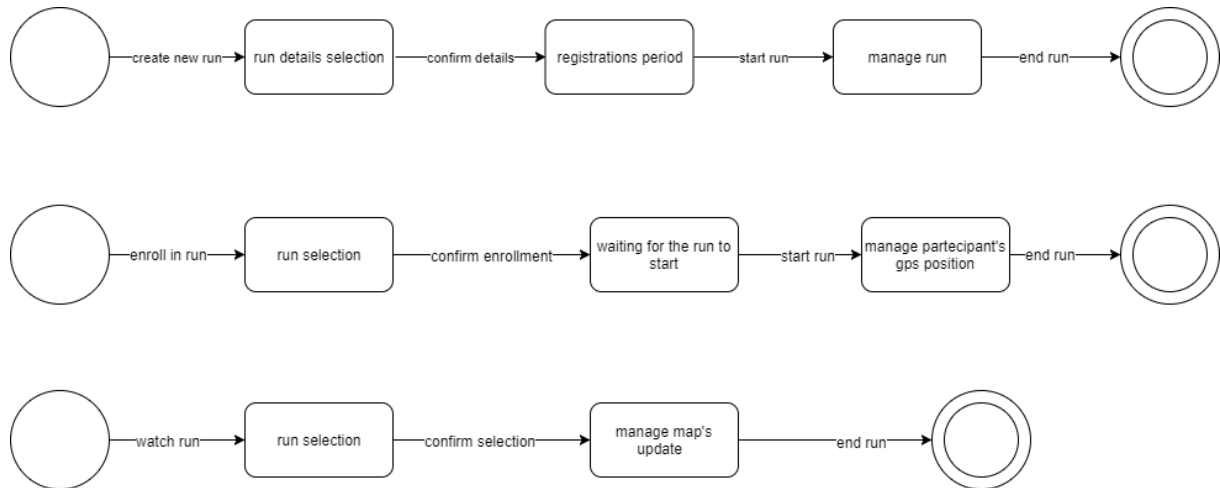


Figure 4: Track4Run state diagram

2.2 Product functions

- The first essential requirement is that all the data must be available remotely for all registered third parties;
- The second essential requirement is that all the data must be available in a anonymized way, so that no third parties can trace back a specific user data;
- The third essential requirement is high reliability for the AutomatedSOS service, because of its nature: the service must automatically detect an emergency and act accordingly without the user's interaction. It must be also active 24/7, must have low time response, and must be connected with an automated emergency call service;
- The fourth essential requirement for the Track4Run service is to be scalable enough for managing any number of active users simultaneously;
- The fifth essential requirement is that third parties users can have access to all the data from a specific user if that user allow them to do so;
- The sixth essential requirement is that third parties users can register to a query that was previously requested;

2.3 User characteristics

Relevant information for each actor:

- Data4Help Users: these users don't require specific needs, as they only passively contribute to the acquisition of general data for the use of third-parties. Only a smartphone and a smartwatch, or similar device, is required for these users.
- Registered Third-parties to Data4Help: their need is to be able to acquire various aggregated type of data, filtering their requests based on one or more parameter of the users, like location, personal informations etc. They must receive an answer if the number of results is not too low. Also all answers are anonymized.
- Users of AutomatedSOS: their need is an application always online which constantly checks their health status and in case of emergency automatically calls an ambulance for them.

- Users of Track4Run: these users must be able to create a specific track, to browse the available ones, and to join in one or more of them. This user must also be able to be tracked by their position during the run. Also, users that have created a run can join as a participant or a spectator as they like.

2.4 Assumptions, dependencies and constraints

- D1 - Every user of Data4Help has a smartwatch or a compatible wearable device.
- D2 - Data acquired through the users' devices is correct and not manipulated.
- D3 - An automatic ambulance calling service exists. Once the nearest hospital is localized by the system, such service gets called.
- D4 - The application has access to the ambulance calling system.
- D5 - The position acquired for tracking each Track4Run's runner is accurate enough, which means that the GPS system needs to be able to track users in a 5 meters diameter.

3 Specific Requirements

3.1 External interface requirements

3.1.1 User Interfaces

The following mockups offer an intuitive view of what the final product will look like.

The 'Sign up' screen for Data4Help features a red header with the app name and a hamburger menu. Below the header, there are input fields for 'name', 'last name', 'nationality', 'age', 'SSN / Fiscal Code', 'email', and 'password'. A link 'already signed up? click here' is positioned above a checkbox for 'agree on terms'. A yellow 'confirm' button is at the bottom.

(a) Data4Help - User - Sign Up

The 'Log in' screen for Data4Help has a red header with the app name and a hamburger menu. It includes input fields for 'email' and 'password'. Below these, there are links for 'forgot password? recover here' and 'not registered yet? click here'. A yellow 'confirm' button is at the bottom.

(b) Data4Help - User - Login

The 'Homepage' for Data4Help features a red header with a home icon, the app name, and a hamburger menu. Below the header, there are icons for 'smart watch', 'smart watch', and 'fitness watch'. A table displays health data: 'heart rate' (80 bpm), 'body pressure' (120/80 mmHg), and 'average steps' (3000 / day). A notification banner at the bottom reads 'New individual data request' with an 'Open' button.

(c) Data4Help - User - Homepage

The 'Sign Up' screen for Data4Help Third Party has a red header with the app name. It includes input fields for 'name', 'email', 'VAT number', and 'password'. A link 'already registered? click here' is above a yellow 'Confirm' button.

(a) Data4Help - Third Party - Sign Up

The 'Log in' screen for Data4Help Third Party has a red header with the app name. It includes input fields for 'VAT number' and 'password'. Below these, there are links for 'not registered? click here' and 'forgot password? click here'. A yellow 'Confirm' button is at the bottom.

(b) Data4Help - Third Party - Login

The 'Homepage' for Data4Help Third Party features a red header with the app name and a hamburger menu. It is divided into two main sections: 'notifications' on the left, showing a list of messages with a 'show messages' button, and 'data requested' on the right, showing a list of queries (Query 1 to Query 5) with a 'new request' button.

Figure 7: Data4Help - Third Party - Homepage

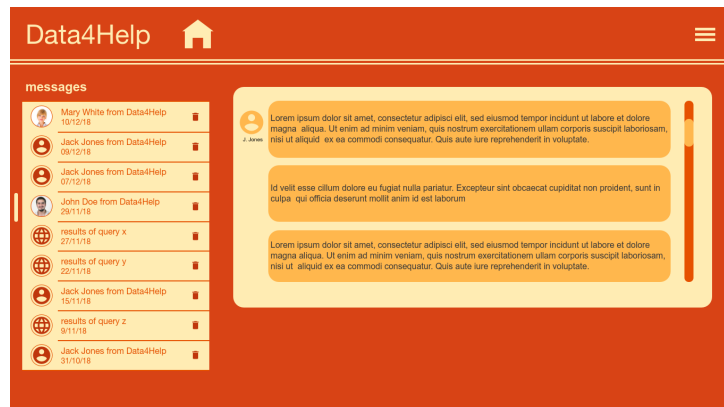


Figure 8: Data4Help - Third Party - Messages



Figure 9: Data4Help - Third Party - Messages - Query Result

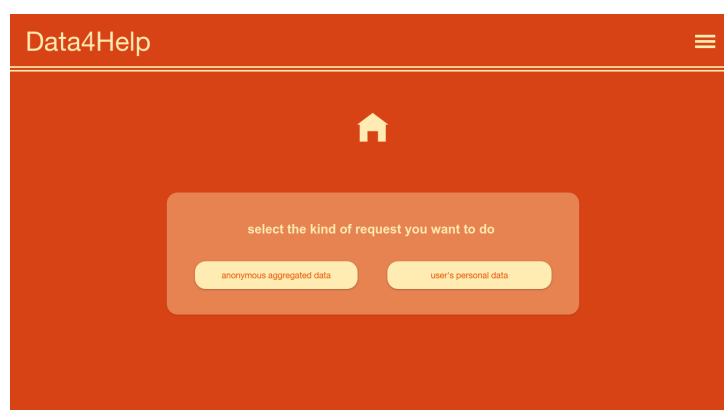
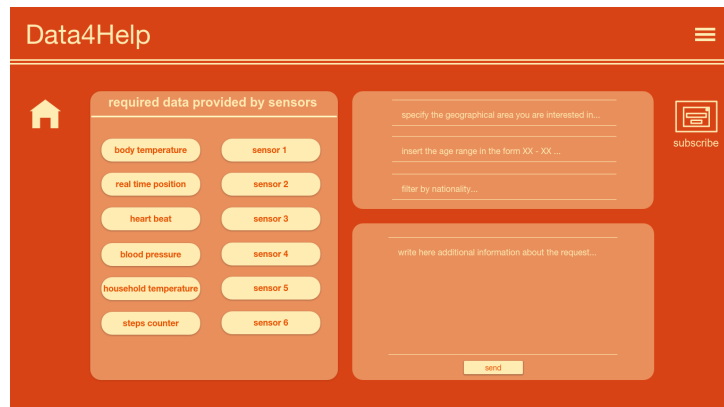
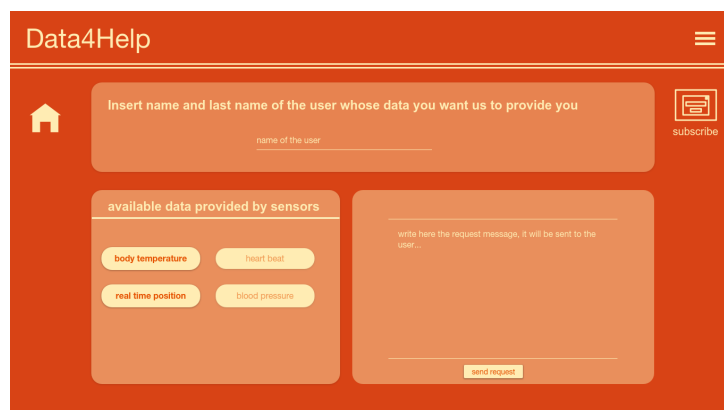


Figure 10: Data4Help - Third Party - New Query Request



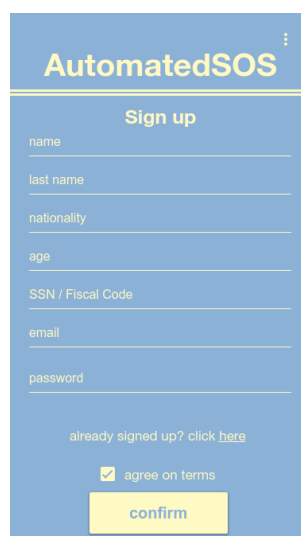
The screenshot shows a web interface for 'Data4Help' with an orange header. On the left is a home icon. The main content area is divided into two columns. The left column, titled 'required data provided by sensors', contains a grid of buttons: 'body temperature' (sensor 1), 'real time position' (sensor 2), 'heart beat' (sensor 3), 'blood pressure' (sensor 4), 'household temperature' (sensor 5), and 'steps counter' (sensor 6). The right column contains a form with three input fields: 'specify the geographical area you are interested in...', 'insert the age range in the form XX - XX ...', and 'filter by nationality...'. Below these is a larger text area for 'write here additional information about the request...' and a 'send' button. A 'subscribe' icon is in the top right corner.

Figure 11: Data4Help - Third Party - Aggregated Data Request



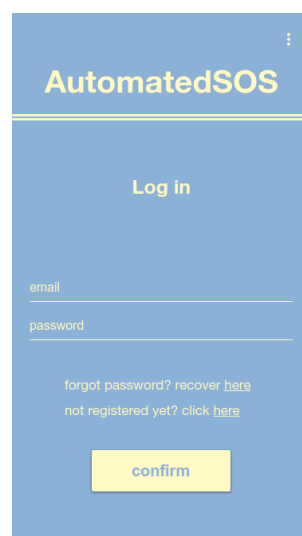
The screenshot shows a web interface for 'Data4Help' with an orange header. On the left is a home icon. The main content area has a top section with the text 'Insert name and last name of the user whose data you want us to provide you' and a 'name of the user' input field. Below this is a section titled 'available data provided by sensors' with buttons for 'body temperature', 'heart beat', 'real time position', and 'blood pressure'. To the right of this is a text area for 'write here the request message, it will be sent to the user...' and a 'send request' button. A 'subscribe' icon is in the top right corner.

Figure 12: Data4Help - Third Party - Individual Data Request



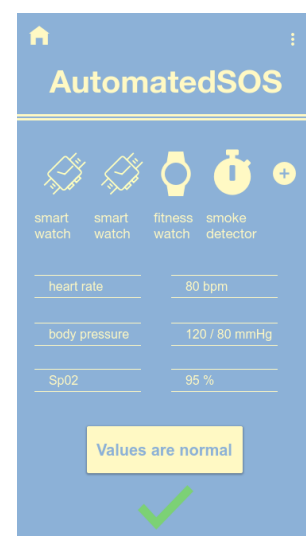
The screenshot shows a mobile app interface for 'AutomatedSOS' with a blue header. Below the header is a 'Sign up' section with input fields for 'name', 'last name', 'nationality', 'age', 'SSN / Fiscal Code', 'email', and 'password'. There is a link 'already signed up? click here' and a checkbox for 'agree on terms'. A yellow 'confirm' button is at the bottom.

(a) AutomatedSOS - Sign Up



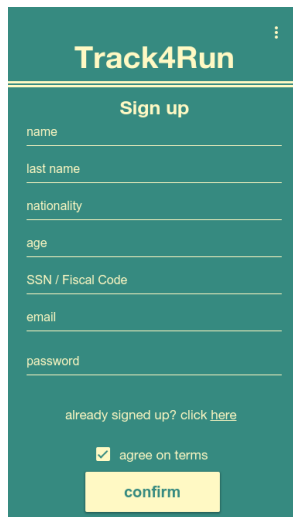
The screenshot shows a mobile app interface for 'AutomatedSOS' with a blue header. Below the header is a 'Log in' section with input fields for 'email' and 'password'. There are links for 'forgot password? recover here' and 'not registered yet? click here'. A yellow 'confirm' button is at the bottom.

(b) AutomatedSOS - Login



The screenshot shows a mobile app interface for 'AutomatedSOS' with a blue header. Below the header is a 'Homepage' section with icons for 'smart watch', 'smart watch', 'fitness watch', and 'smoke detector'. Below these are three rows of data: 'heart rate' (80 bpm), 'body pressure' (120 / 80 mmHg), and 'SpO2' (95 %). A yellow box at the bottom says 'Values are normal' with a green checkmark.

(c) AutomatedSOS - Homepage



Track4Run

Sign up

name

last name

nationality

age

SSN / Fiscal Code

email

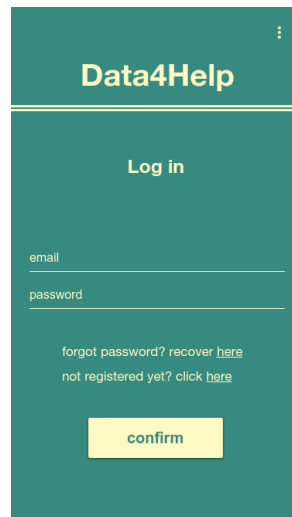
password

already signed up? click [here](#)

☒ agree on terms

confirm

(a) Track4Run - Sign Up



Data4Help

Log in

email

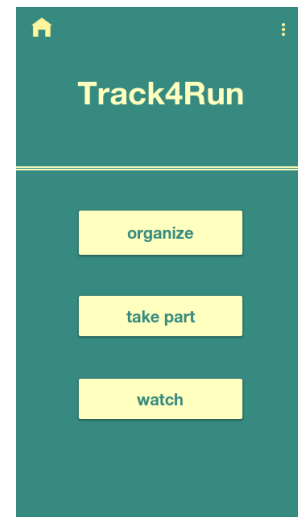
password

forgot password? recover [here](#)

not registered yet? click [here](#)

confirm

(b) Track4Run - Login



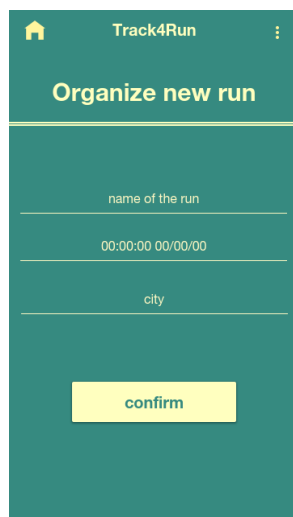
Track4Run

organize

take part

watch

(c) Track4Run - Homepage



Track4Run

Organize new run

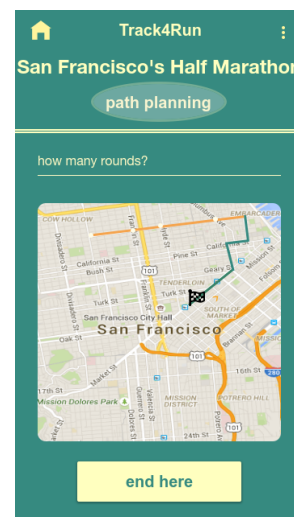
name of the run

00:00:00 00/00/00

city

confirm

(a) Track4Run - Organize

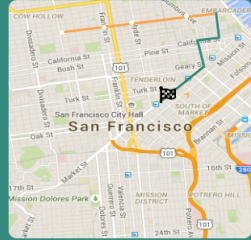


Track4Run

San Francisco's Half Marathon

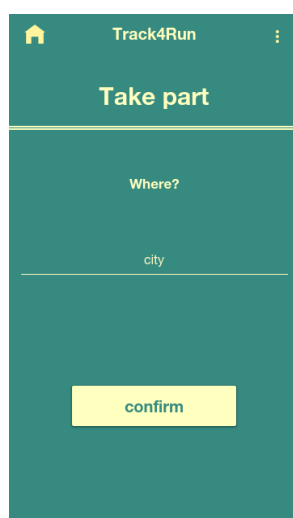
path planning

how many rounds?



end here

(b) Track4Run - Organize - Set Path



Track4Run

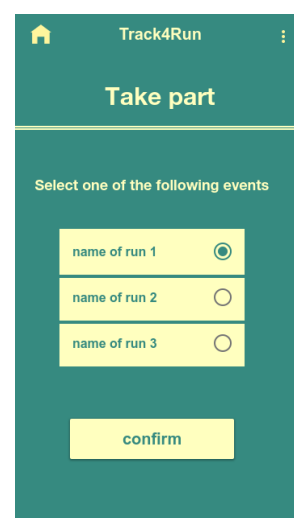
Take part

Where?

city

confirm

(a) Track4Run - Take Part - Step 1



Track4Run

Take part

Select one of the following events

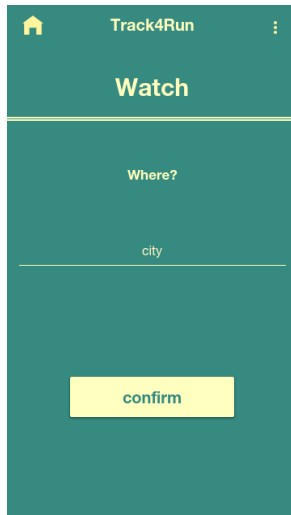
name of run 1 ☒

name of run 2 ☐

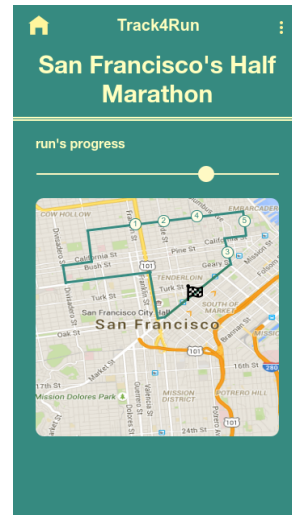
name of run 3 ☐

confirm

(b) Track4Run - Take Part - Step 2



(a) Track4Run - Watch



(b) Track4Run - Watch - Map

D4HTP - Homepage

3.1.2 Hardware Interfaces

[R1] - This application needs a data warehouse to store all the user data and to allow high performance researchs with large amount of data. This is done using the Amazon AWS services.

3.1.3 Software Interfaces

- [R2] - Amazon Redshift as a datawarehouse on top of Amazon S3 to store all the data in an efficient manner;
- [R3] - Google Maps for converting stored GPS data into human readable data for the third parties clients;
- [R4] - Google Maps also for the Track4Run real-time map service.
- [R5] - The API interface for the automatic message sent to the nearest hospital.

3.1.4 Communication Interfaces

[R6] - Both users and third parties will have to use HTTPS to communicate with the TrackMe server, because security and privacy is an important factor in both cases;

3.2 Functional requirements

3.2.1 Use case diagrams

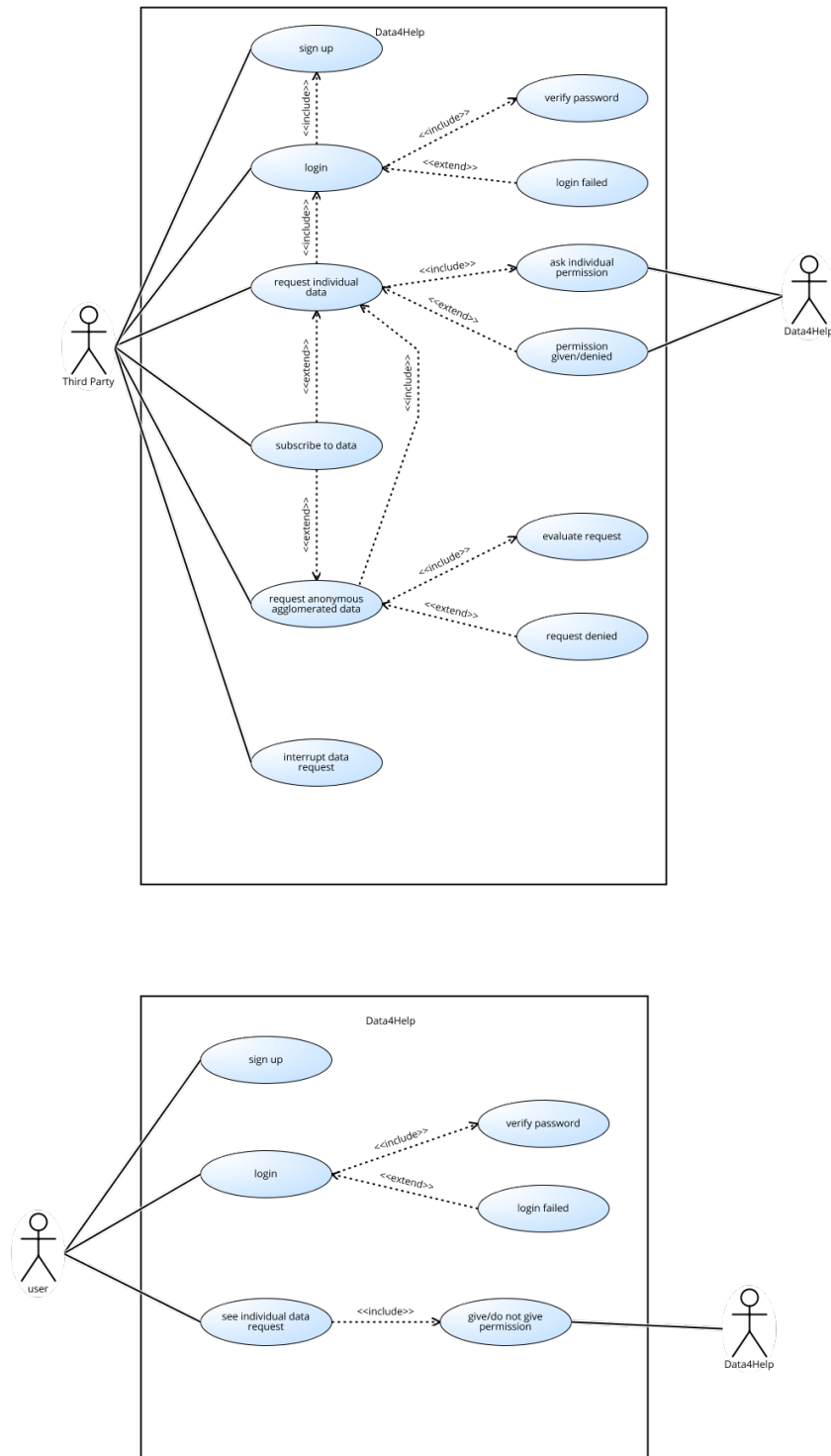


Figure 18: Data4Help

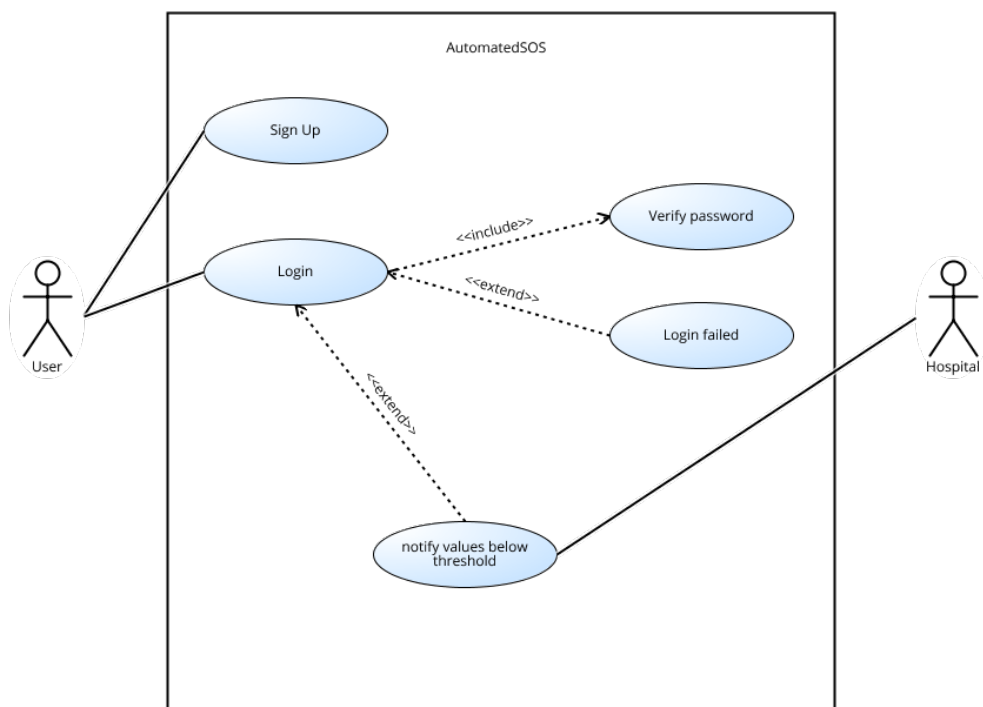


Figure 19: AutomatedSOS

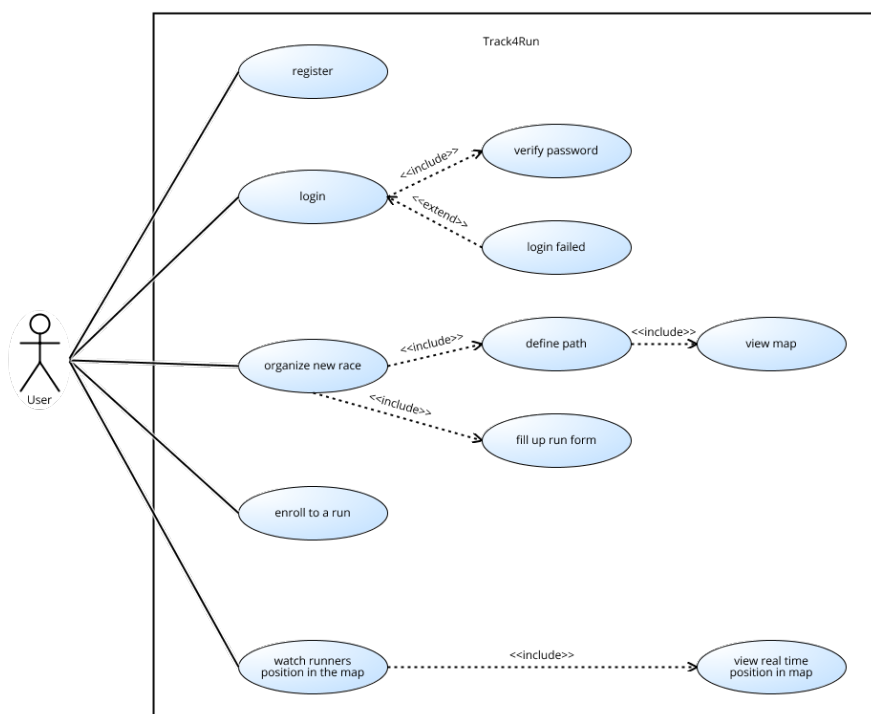


Figure 20: Track4Run

Name	Sign up - Data4Help
Actor	Third Party
Entry condition	the user has installed the application on his device
Event flow	<ol style="list-style-type: none"> 1. Open the application and click on the "sign in" button 2. Fill up the form with his personal information, including his fiscal code 3. Agree with the privacy policy 4. Confirm the registration 5. The system saves the data in its database
Exit condition	The user has successfully registered and is now able to use the service
Exceptions	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a refreshed "sign up" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The email the user provided has already been used 2. The fiscal code the user provided has already been used (he has already registered) 3. The user does not fill all the mandatory info and clicks on "confirm" 4. The user does not agree with the privacy policy and clicks on "confirm"

Name	Login - Data4Help
Actor	Third Party
Entry condition	the user has installed the application on his PC/Mac, registered and logged in
Event flow	<ol style="list-style-type: none"> 1. Open the application and click on the "login" button 2. Insert VAT identification number 3. Insert password 4. Check VAT and password match 5. Enter the desktop application
Exit condition	The user has successfully logged in and is now on the main page of the application
Exception	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a refreshed "login" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The VAT identification number used is unknown by the system 2. The VAT identification number and the password do not match 3. The user does not fill the "VAT identification number" field and clicks on "confirm" 4. The user does not fill the "password" field and clicks on "confirm"

Name	Request individual data - Data4Help
Actor	Third Party
Entry condition	the user has installed the application on his PC/Mac, registered and logged in
Event flow	<ol style="list-style-type: none"> 1. Click on the "new request" button from the homepage 2. Click on the "user's personal data" button 2. Fill form with the name of the individual, kind of data requested, and a message to be shown to the individual 3. Click on the "send request" button 4. Receive response from the service
Exit condition	<p>Two possibilities:</p> <ol style="list-style-type: none"> 1. The TP receives the approval for the request and the asked data. 2. The TP receives a denial for the requested data
Exceptions	<p>All the exceptions are handled by showing to the actor a log which explains the kind of exception. Moreover he is taken back to a refreshed "request individual data" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The name inserted in the form does not match with any name contained in the database

Name	Request anonymous aggregated data - Data4Help
Actor	Third Party
Entry condition	the third party has installed the application on his PC/Mac, registered and logged in
Event flow	<ol style="list-style-type: none"> 1. Click on the "new request" button from the homepage 2. Click on the "anonymous aggregated data" button 3. Specify the kind of data requested (position, age, average heart beat rate,...) 4. Click on the "send" button 5. Receive response from the service
Exit condition	<p>Two possibilities:</p> <ol style="list-style-type: none"> 1. The third party receives the approval for the request and the asked data. 2. The third party receives a denial for the requested data, because the number of individual registered in such area that satisfies the criteria chosen by the third party are less the 1000
Exceptions	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a refreshed "aggregated data request" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. No data is requested from the third party and the "send" button is clicked

Name	Sign up - AutomatedSOS & Track4Run
Actor	User
Entry condition	the user has installed the app on his device
Event flow	<ol style="list-style-type: none"> 1. Click on the "sign up" button 2. Fill up the form with personal info 3. Click on the "confirm" button
Exit condition	The user signs up and is ready to log in
Exceptions	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a refreshed "sign up" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The user does not fill all the mandatory information in the form and clicks on the "confirm" button 2. The user does not agree with the privacy policy and clicks on the confirm button

Name	Login - AutomatedSOS & Track4Run
Actor	User
Entry condition	the user has installed the app on his device and signed up
Event flow	<ol style="list-style-type: none"> 1. Click on the "login" button 2. Insert email 3. Insert password 4. Click on the "confirm" button
Exit condition	The user logs in and is ready to use the service
Exceptions	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a refreshed "login" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The user inserts an e-mail address unknown by the system's database 2. The user inserts a known e-mail address but the password he inserts does not match the provided e-mail

Name	Organize new run - Track4Run
Actor	User
Entry condition	the user has installed and logged in the application on his device
Event flow	<ol style="list-style-type: none"> 1. Click on the "organize" button 2. Fill up form with name of the event, date, location 3. Click on the "confirm" button 4. Select path from available paths by: <ol style="list-style-type: none"> 4a. choosing start point from available street 4b. choosing end point from such street 5. Choose between going back to 4a. or ending path 6. Click on the "confirm" button
Exit condition	The user has successfully organized the run
Exception	<p>All the exceptions are handled by showing to the user a log which explains the kind of exception. Moreover he is taken back to a new "Organize new run" page.</p> <p>The possible exceptions are:</p> <ol style="list-style-type: none"> 1. The user forgets to fill at least one of the field in the form and clicks on the "confirm" button. 2. The city selected to organize the run in is not recognized by the Track4Run systems, which means that Track4Run does not own a stylized map of such city.

3.2.2 Sequence diagrams

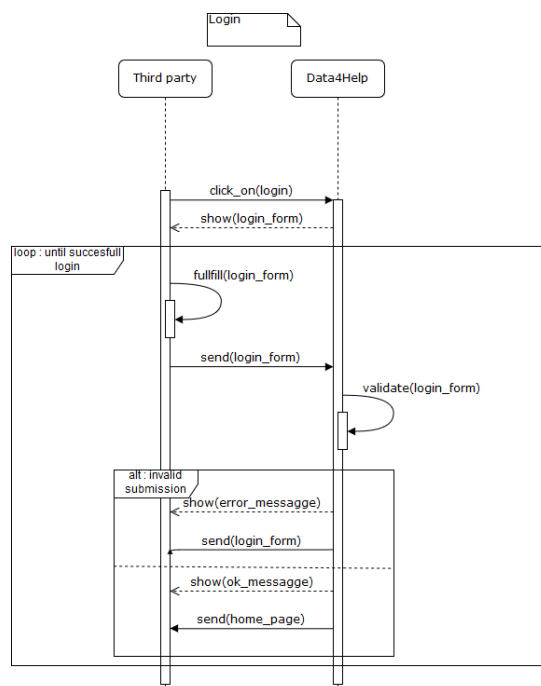


Figure 21: Login sequence diagram

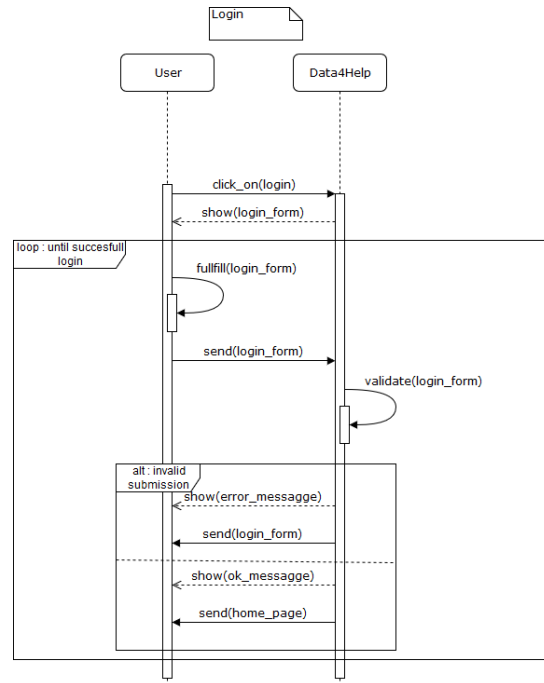


Figure 22: User's login sequence diagram

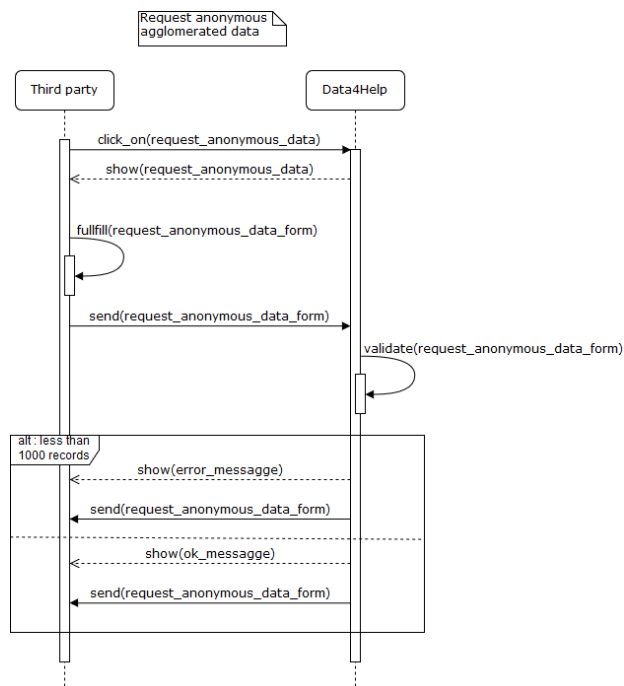


Figure 23: Request anonymous agglomerated data sequence diagram

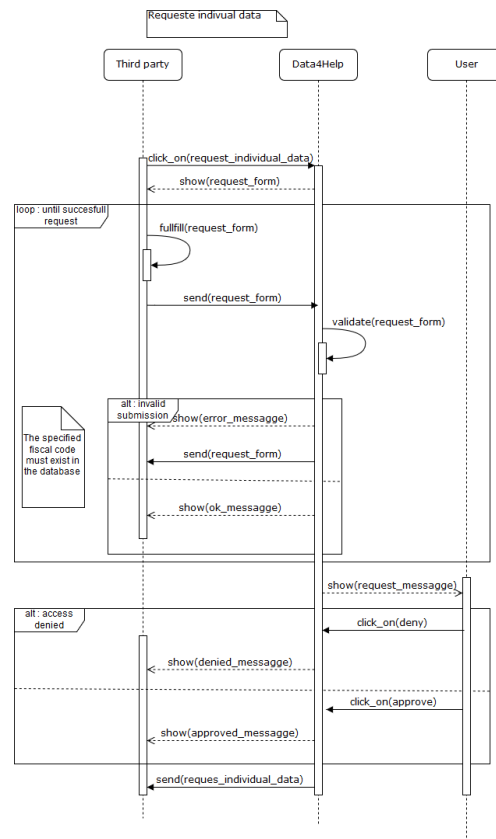


Figure 24: Request individual data sequence diagram

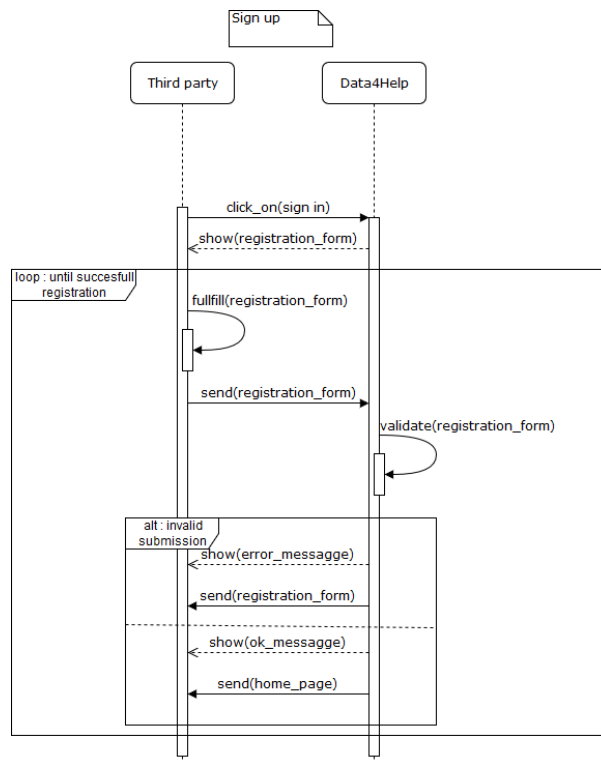


Figure 25: Sign up sequence diagram

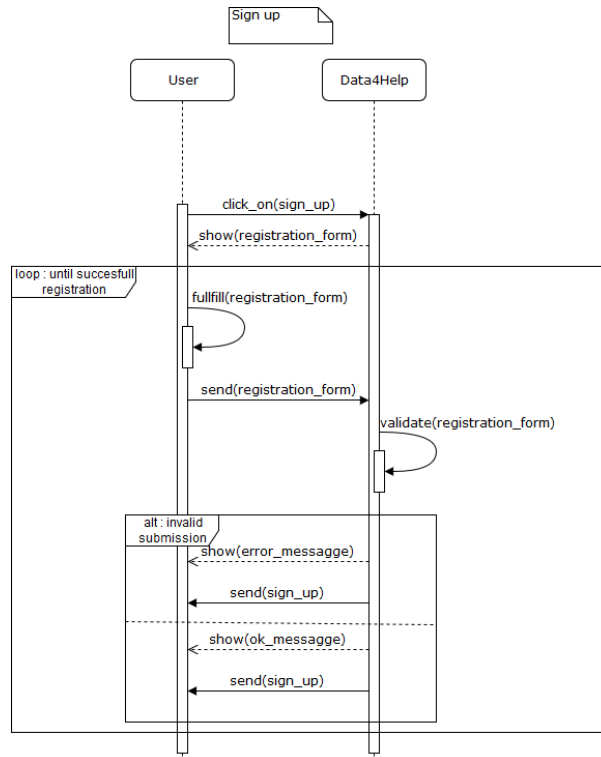


Figure 26: Sign up user sequence diagram

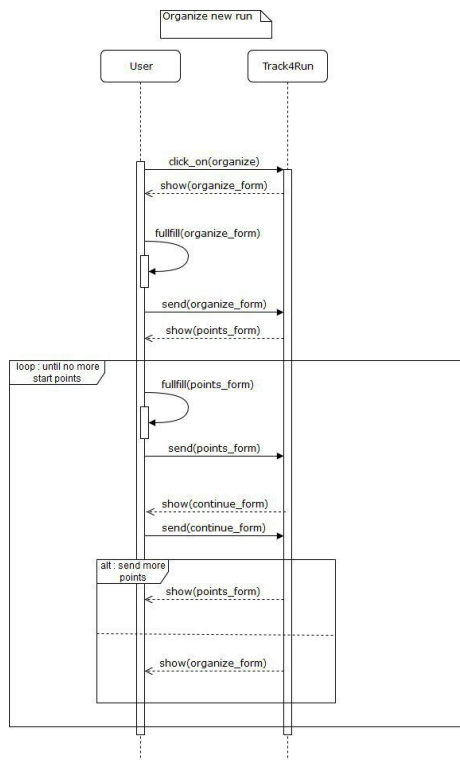


Figure 27: New run sequence diagram

3.2.3 Mapping on requirements

- G1 - The Data4Help application gives constantly new data accordingly to users' activities. This is achieved with [D1], [D2], [R8], [R9], [R10], [R11], [R13], [R14],
[FR1] - The users must be logged in
- G2 - Registered TP (Third Parties) can send request to specific users if they know their SSN or CF to retrieve their specific data. This is achieved with [R2], [R3], [R8], [R10], [FR1],
[FR2] - the users have the possibility of accepting or refusing a non-anonymous data requests.
[FR3] - The TP must be logged in.
- G3 - Registered TP have access to make queries to Data4Help's database to get specific data by filtering by one or more parameters as they need. This is achieved with [D2], [R1], [R2], [R3], [R7], [R8], [R10], [R11], [FR3]
- G4 - Results for TP's queries are given only if the number of matches is greater or equal than 1000. This is achieved with [R1], [R2], [R7].
- G5 - Each user of the services offered by TrackMe is identifiable in order to gather his data and to communicate with him. This is achieved with [R6], [FR1], [R12],
[FR4] - the users must register in the system in order to be identified.
- G6 AutomatedSOS sends automatically a request when and only when some user's values are below a specific critical threshold, specifying the user's position. This is achieved with [D1], [D2], [D3], [D4], [R5], [R6], [R9], [R10], [R11], [R14], [FR1],
[FR5] - The hospitals need to be always available in case of emergency call, this means that there is always an ambulance and some personnel available.
[FR6] - The users of AutomatedSOS need to have their sensors equipped devices connected to the system in order to be helped.
- G7 - Users of Track4Run are able to create a run defining the precise path. This is achieved with [R1], [R6], [R7], [R14], [FR1],
[FR7] - The users of Track4Run, whenever they want to organize a new run, need to insert reasonable gps coordinates in the definition of the path.
[FR8] - The users of Track4Run, whenever they want to organize a new run, need to have the permission of the city hall in case the path defined is public land.
- G8 - Users of Track4Run are able to take part in a created run as runners. This is achieved with [R7], [R14], [FR1],
[FR9] - The users of Track4Run, whenever they take part in a run, need to wear their gps equipped device in order to be tracked.
- G9 - Users of Track4Run can see real-time runners' positions of runs in which they are subscribed. This is achieved with [D5], [R4], [R7], [R11].

3.3 Performance requirements

[R7] - Performance plays a big role in all the 3 projects: regarding the Data4Help and Track4Run they both have to manage a large amount of people that can range from 0 (i.e.: an event organized through Track4Run) to 10000 (i.e.: a fair amount of user that we should expect from Data4Help) based on the

current users requests. Regarding the AutomatedSOS there isn't a specific need for performance, but for reliability, which is explained in the reliability section.

3.4 Design constraints

3.4.1 Standard compliance

[R8] - Internally the data that is collected through Data4Help can be stored in any way that is comfortable for the developers to work with, but when it is delivered to the third parties it must use a standard format. This means that:

1. - R8.1 - the TP can choose of receiving the results of its requests in the form of a summary in a PDF file.
2. - R8.2 - the TP can choose of receiving the results of its request in a database file, in such a way that the third parties can manipulate the data the way they like.
3. - R8.3 - The TP can request both the formats above.

3.4.2 Hardware limitations

The services to be developed would work on any of the smartphones available in the market because the technologies needed for such services in order to work are very basic (bluetooth connection with the sensors equipped devices, gps and internet connection).

3.4.3 Any other constraints

[R9] - Bandwidth may be a constraint, because developers need to pay attention to send only the essential data from the sensors equipped device to the server, because in an hypothetical situation the application may drain too much mobile data from the users' sensors equipped device. To achieve this is possible also to set a reasonable amount of time to be waited between each single user's update of his/her values to the server. On the contrary, AutomatedSOS doesn't have any bandwidth limitation.

3.5 Software system attributes

3.5.1 Reliability

[R10] - Reliability is the main concern for AutomatedSOS, because of the nature of the service itself, and surely is important in the development of the Data4Help module. Secondly, reliability is not the main concern in Track4Run. The whole system should at least be 99.999% reliable.

3.5.2 Availability

[R11] - AutomatedSOS needs a six 9s (99.9999%) availability, due to the high importance availability plays for this service, while Data4Help and AutomatedSOS need at least a five 9s (99.999%) availability.

3.5.3 Security

[R12] - Anonymization must be granted giving the fact that we are working with very sensible data. Since AutomatedSOS and Track4Run talk with Data4Help, this means that the communication between the modules must be protected as well.

3.5.4 Maintainability

[R13] - Maintainability is a big concern for all the three services. Given the fact that the application is mainly developed for Android and iOS, it must be easily maintainable, because of the frequent updates of both the operating systems. This translates in the decoupling of the components of the system.

3.5.5 Portability

[R14] - The application that is developed for Data4Help users must work for all iOS' and Android's updates by the date of launch of the app. The same can be applied to Track4Run and Automated SOS

because it's not easy to foresee how smartwatches' and smartphones' market will change in the future. The desktop application of Data4Help instead has not the same portability requirements, because, being developed for the Windows operating system, we can rely on his well established legacy of portability.

3.6 Scenarios

3.6.1 Scenario 1 - Data4Help

Data Request: A local newspaper of the city of Padova wants to write an article about the average age of the people visiting the city so one of the journalists opens the Data4Help account of his journal on his PC. He finds himself in the homepage of the application and clicks on the "New Request" button. Then He clicks on the "anonymous aggregated data" button. Afterwards a few text fields pop up and he fills them with requesting to the system the age of the people that in the last month spent at most one week in the city. Done so, he asks for a monthly update. He clicks on "send" and waits for the approval of the request. After a few seconds the system approves the request and provides him with a list of numbers corresponding to the ages of such people (such list is provided monthly as requested).

3.6.2 Scenario 2 - Data4Help

Data Request Denied: The touristic office of Santa Lucia, a small town in Sardegna wants to understand how many non italian tourists come visit the city in the summer, so they request data for all the foreigners spending at most 2 weeks in the city (doing the same procedure explained in the scenario above), asking for a 2 weeks update. After 2 weeks Data4Help tells the touristic office, through the in-app notification software, that they can't release the information collected because the number of people who visited the town in that period are less than 1000. In such message Data4Helps tells to the office that they can ask to collect data for the coming next two weeks.

3.6.3 Scenario 3 - Data4Help

Request Interruption: A company that sells school items is willing to open a new shop in Milan. They want to maximize the number of customers visiting their shop so they ask to Data4Help to provide them with the data of the people living in Milan who are between 6 and 20 years old. The service provides to the company the position of each of such individuals, updated hourly. After six months the company is satisfied with the data they have collected, and understands that the part of the city where there is the highest flux of students is Città Studi. So the CMO of the company goes to the homepage of the Data4Help app, clicks on the query he wants to interrupt, and then clicks on the query interruption button. The app responds with a log stating that the removal of the request was successful.

3.6.4 Scenario 4 - Data4Help

Company registration and login: A sportwear company called RunUp wants to start a series of running events in Rome in order to promote their products and their philosophy. The company asks to the chief marketing officer to organize the calendar of events that will be held in Rome. So he downloads the desktop application on his PC and he opens it. The application recognizes this is the first time that it's being opened, so a signing up page gets opened. He registers his company and a form pops up. Such form has 4 fields: company's name, email, VAT number and password. He fills such fields and clicks on the "confirm" button. The Data4Help Staff activates the company's account and the CMO is ready to log in, so he opens the application, clicks on the "Log in" button from the homepage and inserts the VAT number and password in the relative fields. He clicks on the "confirm" button and he's redirected to the his personal space's page. Now the CMO can use the application to make his request to the system.

3.6.5 Scenario 5 - Data4Help

User's Sign up and Login: John is a 25 years old man who wants to enroll in the gym nearby his house. He enters the gym and fills up the enrollment form. At the bottom of such form he is asked through a check box whether he wants to register in Data4Help. Doing so his phone could provide data to the gym, that would be used to help the trainers assigning targeted exercises to John. He checks the box and gives the filled form to the gym's secretary. Then he takes his iPhone, downloads the app from

the Apple Store, opens it and the "sign up" page of the app appears on the screen. Such form consists in seven fields: name, lastname, nationality, age, SSN (or fiscal code), email, and password. He fills all of such fields, checks the privacy policy box, and clicks on the "Confirm" button, so joining the users' community of Data4Help. He gets redirected to the "login" page which consists of two fields: email and password. He fills such fields and clicks on the "confirm" button, being redirected to the home of the app, where he can see the devices paired with his phone and a table containing all the data provided by the sensors paired with his phone.

3.6.6 Scenario 6 - AutomatedSOS

User's Sign up, Log in & Pairing: Maria is a 80 years old woman, which sometimes happens to have difficulties in breathing. This is why she bought a smart blood-oxygen levels sensor that she constantly wears. Since her doctor notices she bought such device, he suggests her to download the AutomatedSOS app. She goes to the Play Store with her Android smartphone and downloads it. Once she downloads it she opens it and the app shows the "sign up" page. Such page consists in a form with seven fields: name, last name, nationality, age, SSN (or Fiscal Code), email and password. She fills such fields, checks the privacy policy box and clicks on the "confirm" button. She gets redirected to the Log in page, where she fills the email and password fields and clicks on the "confirm" button. Now she gets to the homepage of AutomatedSOS, from where she can pair her sensor via bluetooth to the app. She manage to do it by clicking on the "Plus" icon, the smartphone finds the sensor and Maria agrees on pairing the devices.

3.6.7 Scenario 7 - AutomatedSOS

Help request: Walter is a 70 years old man which suffers from asthma, this is why he installed AutomatedSOS on his smartwatch. One evening he is at a friend's house and he has an asthma attack but he forgot his inhaler at home. His smartwatch notices an above-average hearth rate and AutomatedSOS calls an ambulance for him, to be sent at the location of his friend's house. The ambulance arrives in time, knowing he suffers from asthma, the nurses brought along an inhaler to be given to him. He uses the inhaler and his breath goes back to normal.

3.6.8 Scenario 8 - Track4Run

Participant signs up and logs in: Anna is an university student who studies in Zurich. One mon-day morning, after class, she notices that her university, the University of Zurich, is organizing a half-marathon by the lake of Zurich. On the bottom of the flyer is written that in order to participate, each runner has to register via Track4Run. Since she owns an Android smartphone, she downloads the app from the Play Store, and opens it. The "Sign up" page gets displayed. Such page consists in a form to be filled containing the following fields: name, last name, nationality, age, SSN / fiscal code, email, and password. She fills such fields, checks the privacy policy box and clicks on the "confirm" button. Right after she is told her account has been activated and opens the log in page. She fills the email and password fields and clicks on the "confirm" button. Now she can succesfully enroll in the run organized by her university.

3.6.9 Scenario 9 - Track4Run

Organizing run & registration: Mike is a loyal user of Track4Run. One day he decides to organize a run in the main park of his town, Windsor. He opens the Track4Run app on his smartphone and logs in. He gets redirected to the homepage of the app, where he can decide whether he wants to organize, take part or watch a run. He clicks on the "Organize" button and a new page opens where he needs to fill up a form inserting the name of the run, the date and time (up to minute precision), and the city where such run will be held. He clicks on the "confirm" button and a map gets displayed from which he can select among many available paths or combinations of such paths. The selection happens by simply clicking on one of the many streets displayed by the map, selecting a starting point and an ending point (such ending point must correspond to a crossroad or to the actual arrival point of the run). Such process will be repeated until an arrival point will be defined. Once the path is defined Mike clicks on the "end here" button and waits for the approval of Track4Run. Het gets told that the partecipant can take part to

such new run. The day of the competitions arrives, and the event reveals to be a success, due to the high amount of people taking part in it.

3.6.10 Scenario 10 - Track4Run

Watch run: Gabriel is a 50 years old man who has a 15 years old son, Mike. Mike is going to take part to a 10 km local run held in his hometown, Pavia, this sunday, which has been organized via Track4Run. Gabriel is a big fan of his son, unfortunately he will be abroad for work during the weekend, but he knows that he can log in Track4Run as a spectator during the run of his son. On sunday he opens the app and the homepage of the app gets displayed. Here he can choose to click one out of three buttons: "organize", "take part" or "watch". He clicks on the third one and a text field gets displayed. In such field he will write the city the run will be held in, namely Pavia. After doing so he clicks on the "confirm" button. Now a list with the run held in Pavia gets shown, he clicks on "Local Run" out of the list and on the "confirm" button. Now a map displaying a progress bar, and a map of the path gets shown (the runners are represented by a circle inside which there is their id). Now Gabriel can watch his son's performance from his hotel apartment.

3.6.11 Scenario 11 - Track4Run

Taking part in a Run: David is a professional runner. One day, while surfing the internet, he notices that a run will be held in his city, London, in a week. In order to register to the run he will need to enroll via Track4Run. Being a professional runner he already downloaded, signed up and logged in the app. He now just needs to click on the "take part" button in the homepage of the app. A text field gets shown to him, in which he writes the name of the city where such run will be held, namely London. Afterwards he clicks on the "confirm" button. Now a list of run held in London is shown to him, from which he selects "London's half-marathon 2018" and clicks on the "confirm" button. Right after he receives an email with all the run's information.

4 Formal Analysis Using Alloy

4.1 Purpose

4.1.1 General Description

The purpose of this section is to highlight some constraints in the model, through a simplified model of alloy. Three models are described here, one for each service to develop (Data4Help, AutomatedSOS and Track4Run), each with one or more aspect to show:

- **Data4Help:** here the model includes users, their data, third parties and query and requests, and the focus is on these last two. In particular two predicates are showed, one considering a world with queries but without requests and the opposite for the other. In the query example world the focus is on the validity of a query, meaning the number of results provided is high enough to return it to the TP which requested it. Also a global query is shown, meaning a query which returns all registered user's datas. Is present an assertion which checks that a global query actually returns all the userdatas present in the model.
- **AutomatedSOS:** AutomatedSOS consists in a simple model, composed by Devices (implicitly bound to a user, which is not included because not relevant here), Calls and ValuesLists, which represent all the values taken by a specific device in a precise moment in time. The focus is set on the time-critical operation of the call to the ospital, which must be made by the device at the first moment the values of the user reach a dangerous point, and not before or after that moment. Another case shown is the absence of calls in case of safe conditions. Is also made an assertion on the fact that at most one call per device is made in this model.
- **Track4Run:** this last model aims to show the interaction between elements of the application. here are modeled users, runs and registration to them, with a particular focus on users' possibilities and permissions. The two examples show different aspect of the Track4Run system. In the first one is shown a user who is both a creator and a participant, and in the second is shown a user which is visible in a certain time, because he's participating in a run at that time, and not visible in another time. Finally is made an assertion on the visibility of a user which is not a participant: such user cannot be visible at any time.

4.1.2 Simplifications

All of the following simplifications are meant to simplify the models and/or reduce complexity of generated example worlds, without loss of meaning relatively to the focus of each one. By doing simplifications, some integer values are not related strictly to the meaning of their variable, but should be seen instead as a symbolic partition of the real range of the variables.

Data4Help

- Position of a user is of type (x,y) and x,y are bounded [0,2]
- Age of a user is bounded [0,5]
- Queries can group by only equality (not inclusion in range)
- UserData includes only position, age and gender
- Queries are returned when there are at least 3 users in the result, instead of 1000
- Queries return user data not without filtering: in the model the privacy is not considered.

AutomatedSOS

- Values taken in ValuesLists are only pressure, temperature and heartbeat, to set an example.
- All the values in ValuesLists are bounded [0,5]
- Timestamps are bounded [0,2]
- The model shows always exactly 3 values lists per device, to simulate 3 consecutive points in time where values were taken
- The model shows a single case of an emergency situation, resulting in the fact that no more than one call can be made, because the subsequent calls would be pointless

Track4Run

- Time is represented generically with a Time signature. It stands for an abstract amount of time, depending on the run, in which a run can be held (e.g. a couple of hours). For simplicity and abstraction Time is not specified as a date or with a unit of measure
- Runs have at least 2 participants
- Tracks are defined by a set of generic MapPoints, containing at least 2 of them
- Real time maps have an internal date which indicates when the participants are seen by spectators on the map
- VisibleUsersInACertainTime are signatures which indicate which users can be seen through the app in a specific time. This information is not related with any user in particular (i.e. cannot happen that one user can see more participants than another at the same time) because any user can watch a run, so a participant position is accessible by any user during the run

4.2 Code

4.2.1 Data4Help

```

open util/integer
open util/boolean

/**-----SIGNATURES-----*/

//position of a user, represented with 2 simplified coordinates
sig Position {
  latitude: one Int,
  longitude: one Int
}
{latitude >= 0 and latitude <= 2 and longitude >= 0 and longitude <= 2}

//gender of a user
abstract sig Gender{}
one sig Male extends Gender{}
one sig Female extends Gender{}

//data of a user, retrievable from TPs through Data4Help services
sig UserData {
  position: one Position,
  age: one Int,
  gender: one Gender
}
{age >= 0 and age <= 5}

//User of Data4Help
sig User {
  data: one UserData
}

//Registered TP which can do global queries and requests to specific users.
sig ThirdParty {
  grantingUsers: set User
}

//a request made by a TP to a User to grant access to the TP to the user's data, if
  accepted from user.
sig Request {
  thirdParty: one ThirdParty,
  user: one User,
  grantedAccess: lone Bool
  //the Bool is lone because if the Bool is not set means the request is still pending
}

/*a Query made by a TP to the Data4Help database. The query is considered valid only
  if the number
  of results is greater or equal to 3 (for alloy model-simplification, instead of
  1000).
  It can be formulated on 0 or more variables, contained in UserDatas. With 0
  variables is a global
  query and returns all the Database.
  A query has no reference to a TP because there are no constraints depending on

```

```

        which TP formulates it
*/
sig Query {
  isValid: Bool,
  usersResult: set UserData,
  groupByPosition: Bool,
  position: lone Position,
  groupByAge: Bool,
  age: lone Int,
  groupByGender: Bool,
  gender: lone Gender
}

/**-----FACTS-----*/

//at most one request can exist for each couple (third party, user)
fact OnlyOneRequestForEachCouple{
  all disj r1, r2: Request |
    ((r1.thirdParty != r2.thirdParty) or
     (r1.user != r2.user))
}

//each User has its own distinct UserData
fact UniqueUserData {
  all disj u1, u2:User |
    u1.data != u2.data
}

//each UserData belongs to a User
fact UserDataBoundToUser {
  all ud:UserData |
    some u:User |
      u.data = ud
}

//each user in list of a third party's granted user list has agreed to one and only
//request with that specific third party
fact ThirdPartyKnownUsersMustHaveAgreed {
  all u: User, tp:ThirdParty |
    u in tp.grantingUsers
    implies
    (one r:Request |
     r.thirdParty = tp and r.user = u and r.grantedAccess = True)
}

//how a query works: its result is composed of users that have the same attribute as
//asked, if that attribute is asked.
fact QueryStructure{
  all ud: UserData, q: Query |
    (ud in q.usersResult
     iff
     ((q.groupByPosition = True) implies (q.position = ud.position)) and
     (q.groupByAge = True implies q.age = ud.age) and
     (q.groupByGender = True implies q.gender = ud.gender)
    )
}

```

```

//states when a query is returnable (valid) to the TP who asked it. It's returnable
  if the number of users is at least 3.
fact validQuery{
  all q: Query |
    q.isValid = True iff
      #q.usersResult >= 3
}

/**-----ASSERTIONS-----*/

//if a query has no constraint, must return all existing UserDatas
assert GlobalQuery {
  all q:Query |
    (q.groupByPosition = False and q.groupByAge = False and q.groupByGender = False)
    implies
      (all ud:UserData | ud in q.usersResult)
}

/**-----PREDICATES-----*/

//show that...
pred showRequest {
  //...at least one user has 2 or more requests but only a part of them accepted,
  and...
  some r1, r2:Request | r1.user = r2.user and
    r1.grantedAccess = True and r2.grantedAccess != True
  //...at the same time a TP has at least 2 requests, some accepted and some not
  some r3, r4:Request | r3.thirdParty = r4.thirdParty and
    r3.grantedAccess = True and r4.grantedAccess != True
}

//show at least a valid query, a non-valid one and a global query, all different ones
pred showQuery {
  some disj q1, q2, q3: Query |
    //q1 is valid
    (q1.isValid = True) and
    //q2 is not valid
    (q2.isValid = False) and
    //q3 is a global query
    (q3.groupByPosition = False and q3.groupByAge = False and q3.groupByGender =
      False)
}

check GlobalQuery

run showQuery for 4 but 3 Query, 0 Request, 0 ThirdParty

run showRequest for 4 but 0 Query

```

4.2.2 AutomatedSOS

```

open util/integer
open util/boolean

/**-----SIGNATURES-----*/

//List of values registered from the device (simplified).
//Also has a timestamp which indicates when the values were taken, and a boolean
    which is true if the values
//denote a dangerous condition for the user's health
sig ValuesList {
    pressure: Int,
    temperature: Int,
    heartbeat: Int,
    timestamp: Int,
    dangerousCondition: one Bool
} {pressure >= 0 and pressure <= 5 and
    temperature >= 0 and temperature <= 5 and
    heartbeat >= 0 and heartbeat <= 5 and
    timestamp >= 0 and timestamp <= 2
}

//the device of a user
sig Device {
    values: set ValuesList
} {#values = 3}

//a call sent to the nearest hospital. Contains the device which made it, with its
    data, and the time of the call.
sig Call {
    caller: one Device,
    timestamp: one Int
}

/**-----FACTS-----*/

//a valuesList belongs only to a device
fact ValuesListsBoundedToDevice {
    all v: ValuesList |
        one d: Device |
            v in d.values
}

//different ValuesLists have different timestamps if they belong to the same device
fact DevicesValuesListsHaveDistinctTimestamps {
    all disj v1, v2: ValuesList |
        (one d: Device |
            v1 in d.values and v2 in d.values)
    implies
        v1.timestamp != v2.timestamp
}

//definition of dangerous values. For simplicity, is 0 or 5 for temperature and
    pressure, 0 for heartbeat
fact WhenValuesAreDangerous {
    all v: ValuesList |

```

```

    v.dangerousCondition = True
    iff
      ((v.pressure = 0 or v.pressure = 5) or
       (v.temperature = 0 or v.temperature = 5) or
       (v.heartbeat = 0))
  }

//calls are made from a device that has at least one valuesList with dangerousValue
true
fact CallsFromEndangeredUsers {
  all c: Call |
    one v:ValuesList |
      v in c.caller.values and
      v.dangerousCondition = True and
      c.timestamp = v.timestamp
}

//if a user is in dangerous condition, a call is always made
fact CallsAreMandatory {
  all v:ValuesList |
    one d:Device|
      (v in d.values and
       v.dangerousCondition = True)
      implies
      some c: Call |
        c.caller = d
}

//2 calls with same caller and timestamp cannot exist
fact NoSameCall {
  no disj c1, c2: Call |
    (c1.caller = c2.caller and
     c1.timestamp = c2.timestamp)
}

//calls are made in the first moment (timestamp) the values are dangerous
fact CallsAtFirstDangerousCondition {
  all c: Call |
    no v1, v2: ValuesList |
      v1.dangerousCondition = True and
      v2.dangerousCondition = True and
      c.timestamp = v1.timestamp and
      v1 in c.caller.values and
      v2 in c.caller.values and
      v1.timestamp > v2.timestamp
}

/**-----ASSERTIONS-----*/

//at most there's only one call per device
assert MaxOneCallPerDevice {
  all d:Device |
    lone c:Call |
      c.caller = d
}

```

```
/**-----PREDICATES-----*/

//show a device without dangerous condition and that there are no calls in this case
pred showDeviceInSafeConditions {
  some d: Device |
    True not in d.values.dangerousCondition
}

//show a device in dangerous condition starting from timestamp 1, to show that the
//call is made exactly when the
//values are dangerous, and that after that no other calls are made
pred showDeviceInDangerousCondition {
  some d:Device |
    True in d.values.dangerousCondition
    and
    (some disj v1, v2: ValuesList |
      v1 in d.values and v2 in d.values and
      v1.timestamp = 0 and v1.dangerousCondition = False and
      v2.timestamp = 1 and v2.dangerousCondition = True
    )
}

check MaxOneCallPerDevice

run showDeviceInSafeConditions for 3

run showDeviceInDangerousCondition for 3
```

4.2.3 Track4Run

```

/**-----SIGNATURES-----*/
//The user of the Track4Run application. Has a set of Runs he/she created, a set of
  Runs to which he/she is
//subscribed and a set of accessible maps of runs.
sig User {
  createdRuns: set Run,
  subscribedRuns: set Run
}

//this represent a certain time in a day. It's not specified a precise amount of time
  for abstraction and simplicity
//of the model, but ideally represent the time in which a run is held.
sig Time{}

//list of visible users in a precise time. This is an information independent from
  any particular user, because
//each user can see any run, if the run is underway, and cannot if it's finished or
  has yet to start
sig VisibleUsersInACertainTime {
  timeOfVisibility: one Time,
  visibleUsers: set User
}

//this represents a point in the real world (simplified).
sig MapPoint {}

//A Run is an event created by one user (who can join as a participant or not join at
  all)
//and has a defined track, a map seen by spectators only and a time indicating when
  the run is held
sig Run {
  creator: one User,
  participants: set User,
  track: set MapPoint, //for simplicity a track is only defined by 2 or more MapPoint
  realTimeMap: one RealTimeMap,
  time: one Time
}
{#track >= 2 and
#participants >= 2} //for simplicity there are always at least 2 participants.

//a map for a run.
sig RealTimeMap {
  visibleUsers: set User,
  //this represent the time when users are visible to the spectators of the relative
    run
  visibilityTime: one Time
}

/**-----FACTS-----*/

//A user subscribed to a run must be in that run as a participant

```

```
fact UserInRunIfSubscribed {
  all u: User, r: Run |
    r in u.subscribedRuns implies
      u in r.participants
}

//the set of runs created is linked to the creator of the run
fact UserCreatorOfRuns {
  all u: User, r: Run |
    r in u.createdRuns iff
      u = r.creator
}

//the time of a run is the visibility time of its associated map
fact RunAndMapHaveSameTime {
  all r:Run |
    r.realTimeMap.visibilityTime = r.time
}

//all maps are associated with 1 and only run
fact MapsBoundedToOneRun{
  all m: RealTimeMap |
    one r:Run |
      m = r.realTimeMap
}

//in each run's map the users are the only users which are subscribed as participant
//of that run
fact usersOfAMap {
  all m: RealTimeMap, r:Run |
    m.visibleUsers = r.participants iff
      m = r.realTimeMap
}

//for each time is defined 1 and only 1 list of visibleUsersInACertainTime
fact VisibleUsersDefinedForEachTime {
  all t:Time |
    one v:VisibleUsersInACertainTime |
      v.timeOfVisibility = t
}

//for each time all the visible users are all and only the ones subscribed to runs
//which are held in that time
fact DefinitionOfVisibleUsers {
  all v:VisibleUsersInACertainTime, u:User |
    u in v.visibleUsers iff
      (some r:Run |
        r in u.subscribedRuns and
        r.time = v.timeOfVisibility)
}

/**-----ASSERTIONS-----*/

//check that users who are not subscribed to any run are never visible.
assert NoVisibleUserNotSubscribed {
  all v:VisibleUsersInACertainTime, u:User |
```



```
#u.subscribedRuns = 0 implies
u not in v.visibleUsers
}

/**-----PREDICATES-----*/

//show a user which is visible in a time but not in another
pred showUserVisibleForLimitedTime {
  some u:User, v1, v2: VisibleUsersInACertainTime |
    u in v1.visibleUsers and u not in v2.visibleUsers
}

//show that a creator can join as a participant
pred showUserCreatorAndSubscriber{
  some u:User |
    some r: Run |
      u = r.creator and
      u in r.participants
}

check NoVisibleUserNotSubscribed

run showUserCreatorAndSubscriber for 2 but 3 User

run showUserVisibleForLimitedTime for 2 but 3 Time
```

4.3 Worlds Generated

4.3.1 Data4Help

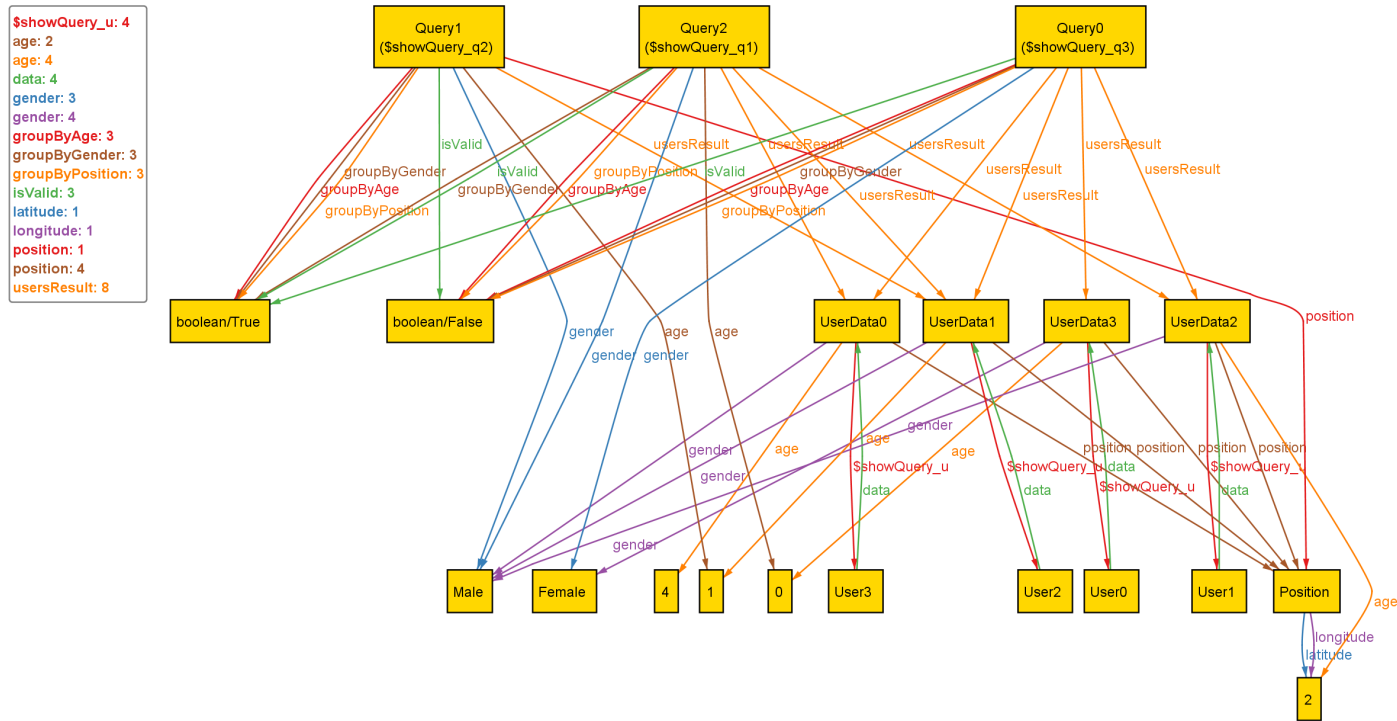


Figure 28: Data4Help case 1: q1 is a valid query, q2 is a non-valid query, q3 is a global query

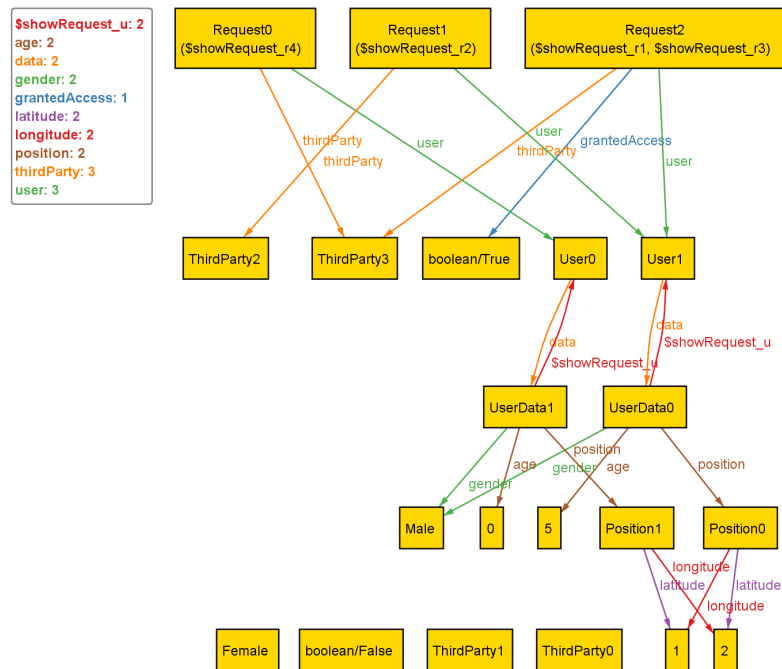


Figure 29: Data4Help case 2: r1 and r2 are requests with same user, but 1 is accepted and the other is not. r3 and r4 are requests made by the same TP, one accepted and the other not

4.3.2 AutomatedSOS

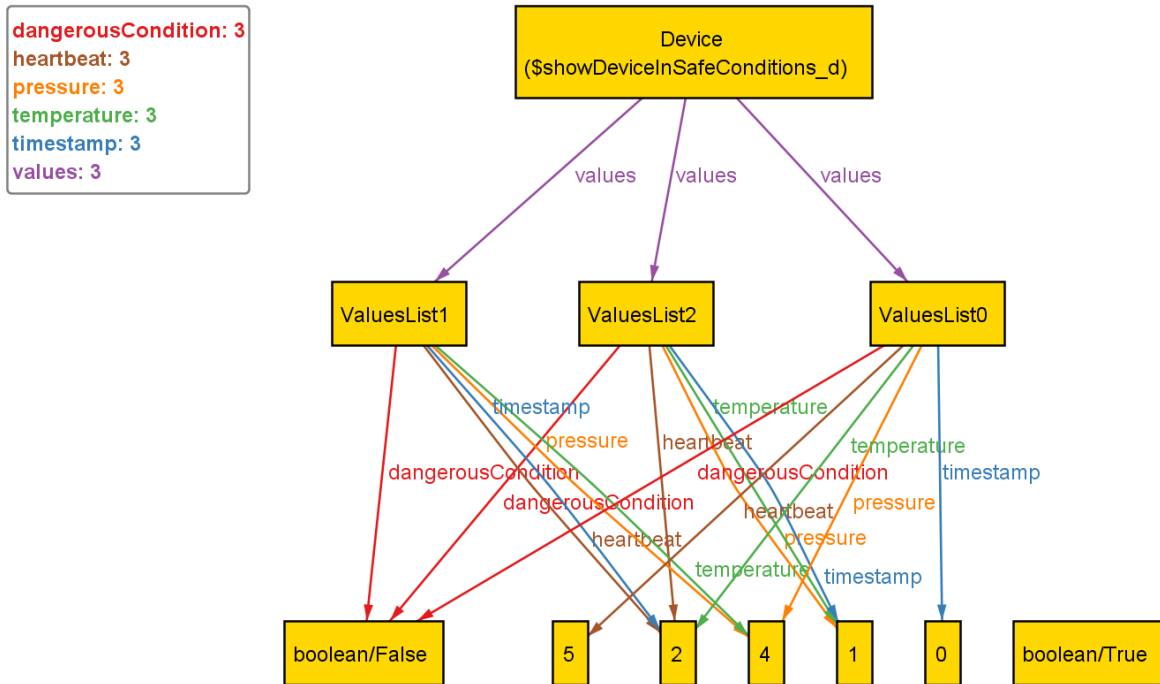


Figure 30: Automated SOS case 1: a device which never registers dangerous conditions

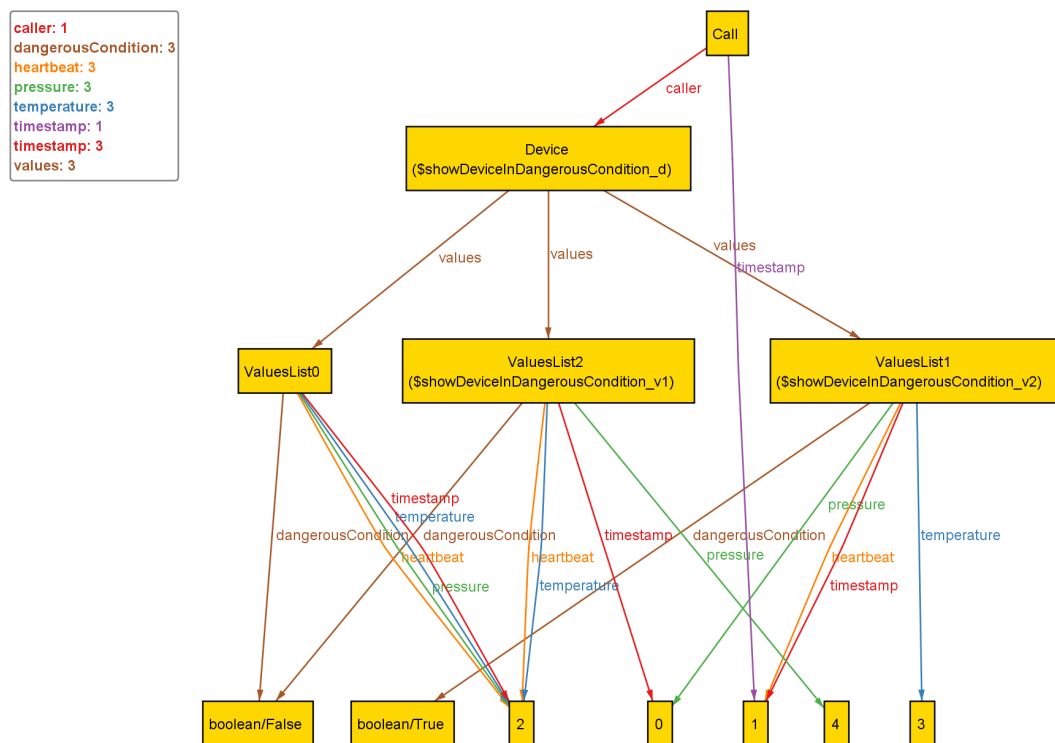


Figure 31: Automated SOS case 2: a device which registers a dangerous condition starting from timestamp 1, having so a call at the same timestamp

4.3.3 Track4Run

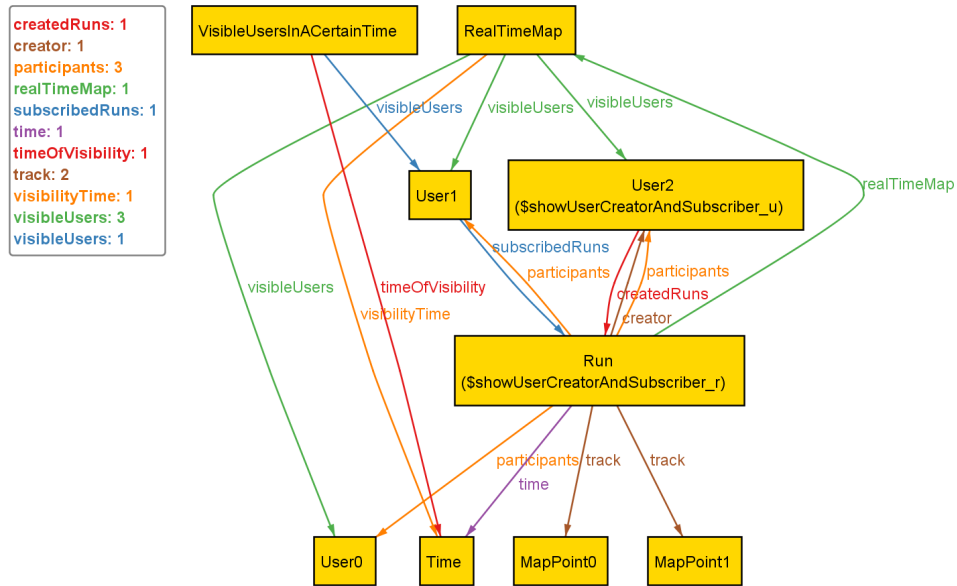


Figure 32: Track4Run case 1: user u (User2) is at the same time the creator and a participant in the same run

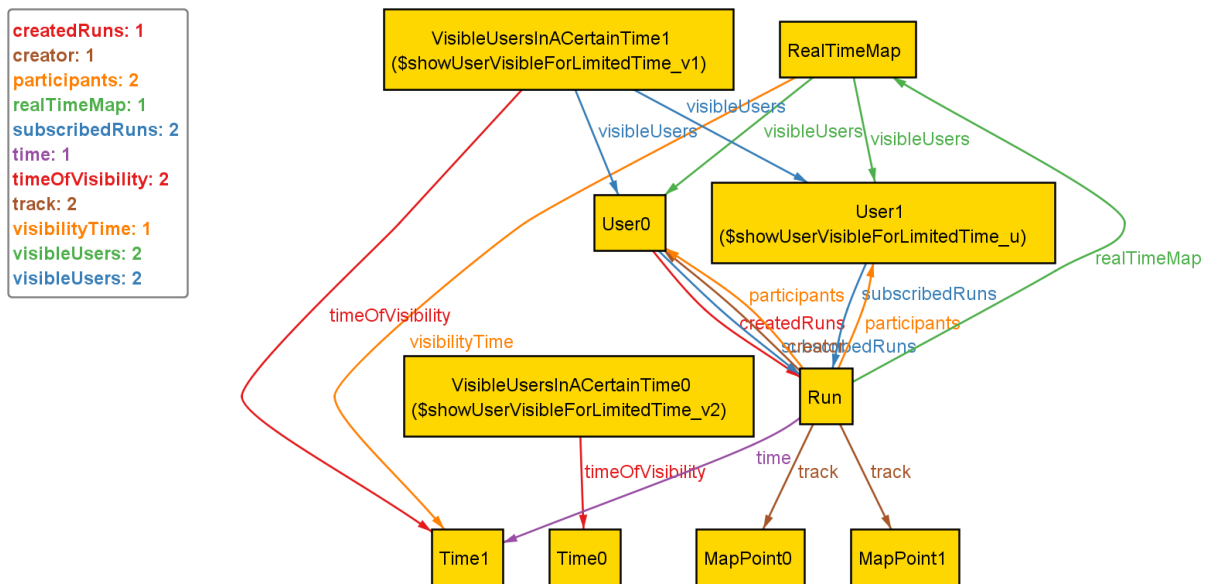


Figure 33: Track4Run case 2: user u (User1) is visible in a certain time (Time1) but not in another (Time 0)

4.4 Alloy results

```
Executing "Check GlobalQuery"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
4266 vars. 330 primary vars. 10898 clauses. 35ms.
No counterexample found. Assertion may be valid. 13ms.

Executing "Run showQuery for 4 but 3 Query, 0 Request, 0 ThirdParty"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
5027 vars. 374 primary vars. 13387 clauses. 29ms.
Instance found. Predicate is consistent. 29ms.

Executing "Run showRequest for 4 but 0 Query"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
4953 vars. 340 primary vars. 12429 clauses. 45ms.
Instance found. Predicate is consistent. 46ms.

3 commands were executed. The results are:
#1: No counterexample found. GlobalQuery may be valid.
#2: Instance found. showQuery is consistent.
#3: Instance found. showRequest is consistent.
```

Figure 34: Data4Help's alloy results

```
Executing "Check MaxOneCallPerDevice"
Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
4147 vars. 276 primary vars. 11874 clauses. 34ms.
No counterexample found. Assertion may be valid. 62ms.

Executing "Run showDeviceInSafeConditions for 3"
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
4132 vars. 276 primary vars. 11844 clauses. 37ms.
Instance found. Predicate is consistent. 37ms.

Executing "Run showDeviceInDangerousCondition for 3"
Solver=sat4j Bitwidth=4 MaxSeq=3 SkolemDepth=1 Symmetry=20
4343 vars. 282 primary vars. 12202 clauses. 37ms.
Instance found. Predicate is consistent. 30ms.

3 commands were executed. The results are:
#1: No counterexample found. MaxOneCallPerDevice may be valid.
#2: Instance found. showDeviceInSafeConditions is consistent.
#3: Instance found. showDeviceInDangerousCondition is consistent.
```

Figure 35: AutomatedSOS' alloy results

```
Executing "Check NoVisibleUserNotSubscribed"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
1656 vars. 123 primary vars. 3190 clauses. 36ms.
No counterexample found. Assertion may be valid. 10ms.

Executing "Run showUserCreatorAndSubscriber for 2 but 3 User"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
879 vars. 74 primary vars. 1549 clauses. 30ms.
Instance found. Predicate is consistent. 22ms.

Executing "Run showUserVisibleForLimitedTime for 2 but 3 Time"
Solver=sat4j Bitwidth=0 MaxSeq=0 SkolemDepth=1 Symmetry=20
798 vars. 69 primary vars. 1388 clauses. 23ms.
Instance found. Predicate is consistent. 31ms.

3 commands were executed. The results are:
#1: No counterexample found. NoVisibleUserNotSubscribed may be valid.
#2: Instance found. showUserCreatorAndSubscriber is consistent.
#3: Instance found. showUserVisibleForLimitedTime is consistent.
```

Figure 36: Track4Run's alloy results

5 Effort Spent

5.1 Leonardo Barilani

Description of the task	Hours
Introduction	4
Overall description	6
Specific requirements	14
Formal analysis using Alloy	2

5.2 William Bonvini

Description of the task	Hours
Introduction	4
Overall description	4
Specific requirements	22
Formal analysis using Alloy	2

5.3 Lorenzo Carnaghi

Description of the task	Hours
Introduction	4
Overall description	4
Specific requirements	4
Formal analysis using Alloy	16