Politecnico di Milano

AA 2018-2019

POLITECNICO
MILANO 1863

Computer Science and Engineering

Software Engineering 2

RASD

Requirement Analysis and Specification Document

Version 1.0 - 11.11.2018

Sankari Gopalakrishnan

David Brellmann

Louis Lesieur

Table of Contents

1.Introduction

1.1 Purpose

The purpose of this document is to provide a detailed description of the TrackMe system. This will be done by a presentation of the proposed solution and its purpose, listing its goals, requirements and assumptions through which they will be achieved. The document is meant to be used by the clients, users and by the parties designated with the task of creating the specified system, mainly the system and requirements analysts, the project managers, software developers and testers.

The TrackMe system is designed as a software application used for collecting data and providing access to it. Three services are managed by the application. The first one is called Data4Help. It can be useful for third parties such as health-interested organizations or insurance companies. The service collects the location and health data of the users and registered third party users of Data4Help can request user specific data and anonymous data of groups of individuals. The data is sent if the condition of anonymity is sufficient or if the user approves the request for his/her data. Also, the third parties can subscribe to data to receive it as soon as it is produced. The goal of the second service, AutomatedSOS , is to provide help to elderly people, by sending an ambulance as soon as possible when the data collected from them shows an immediate danger for their life. The application takes advantage of the device that sends non-stop data to offer a service that will save lives. The third service, Track4Run, is to support to all the actors of runs. The run organisers can set up the path of the run and set up an enrolment process for the run participants, who are users of Data4Help with their own device. The application then shows to the spectators, a real time map view with the location of all the participants.

1.2 Scope

1.2.1 Description of the given problem

As already mentioned, the Data4Help service is expected to give anonymous health data to third parties requiring it. The anonymity should be always considered for the privacy of the users. That is to say that the application should not provide data which could be misused by third parties, for instance if the size of specific category of persons whose data is required is too few.

The second service is meant to call help for elderly people if they need it. Thus, the application should monitor the data continuously, and not just retrieve the data of the device once a day. In order to know when one’s situation becomes dangerous, AutomatedSOS should allow elderly

users to set thresholds in their parameters to determine their limits. Moreover, the system should manage to contact an ambulance facility quickly enough to be helpful.

For the Track4Run service in particular, and also for the others, TrackMe has to ensure the interaction with the GPS integrated in the device, to monitor the location of the users.

1.2.3 Goals

[G1] – Collect data (location, health status) from all registered users.

[G2] – Registered third parties can access data from individual users with permission.

[G3] – Registered third parties can request for anonymized data of groups of individuals.

[G4] – Registered first parties can subscribe and receive data.

[G5] – Registered elderly users can subscribe for a personalized and non-intrusive SOS service.

[G6] – Call an ambulance for subscribed elderly people if needed.

[G7] – Run organisers can set up a path for the run.

[G8] – Run organisers can set up enrolment process.

[G9] – Run participants can enroll for a run.

[G10] – Showing the situation of the run to spectators.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Device/Wearable: any device owned by the user which is able to collect the data and sending it to the application
- Third party: association or business interested in collecting data
- Health status: cardiac rhythm, number of steps in a period of time

1.3.2 Acronyms

- RASD – Requirement Analysis and Specification Document
- API – Application Programming Interface
- GPS – Global Positioning System
- RO – Run organisers
- RP – Run participants

1.3.3 Abbreviations

- [Gn] : n-th goal
- [Rn] : n-th functional requirement
- [Dn] : n-th domain assumption

1.4 Document Structure

Chapter 1 introduces the problem and describes the purpose of the 3 services managed by the application TrackMe: Data4Help, AutomatedSOS, and Track4Run. The scope of application is defined by stating the goals and description of the 3 services.

Chapter 2 presents the overall description of the project. The product perspective includes details on the shared phenomena and the domain models. The class diagram describes the domain model used, and the state diagram analyses the processes of providing data, sending an ambulance and managing a run. Here most functions of the system are more precisely specified, with respect to the already mentioned goals of the system. In the user characteristics section, the types of actors that can use the application are described.

Chapter 3 contains the external interface requirements, including: user interfaces, hardware interfaces, software interfaces and communication interfaces. Few scenarios describing specific situations are listed here. Furthermore, the functional requirements are defined by using use case and sequence diagram. The non-functional requirements are defined through performance requirements, design constraints and software system attributes.

Chapter 4 includes the Alloy model. Also, a world generated by it is shown.

Chapter 5 shows the effort spent by each group member while working on this project.

Chapter 6 includes the reference documents.

2. Overall Description

2.1 Product Perspective

The system acts as a middle man between users and third parties by collecting data from registered users and forwarding requested data to third parties with due permission or anonymized. As added services, the system provides services for elderly people and marathon organizers. Elderly people can make use of a non-intrusive automatedSOS service by entering their health parameters which will be monitored by the system and ambulance will be made available when the values reach below threshold. Run organizers can make use of the system to setup run path and participants can enroll for the same. A spectator view is also provided with participant’s position on map.

The entities involved in the system is described with the help of a class diagram and the various control flows are described using an activity diagram

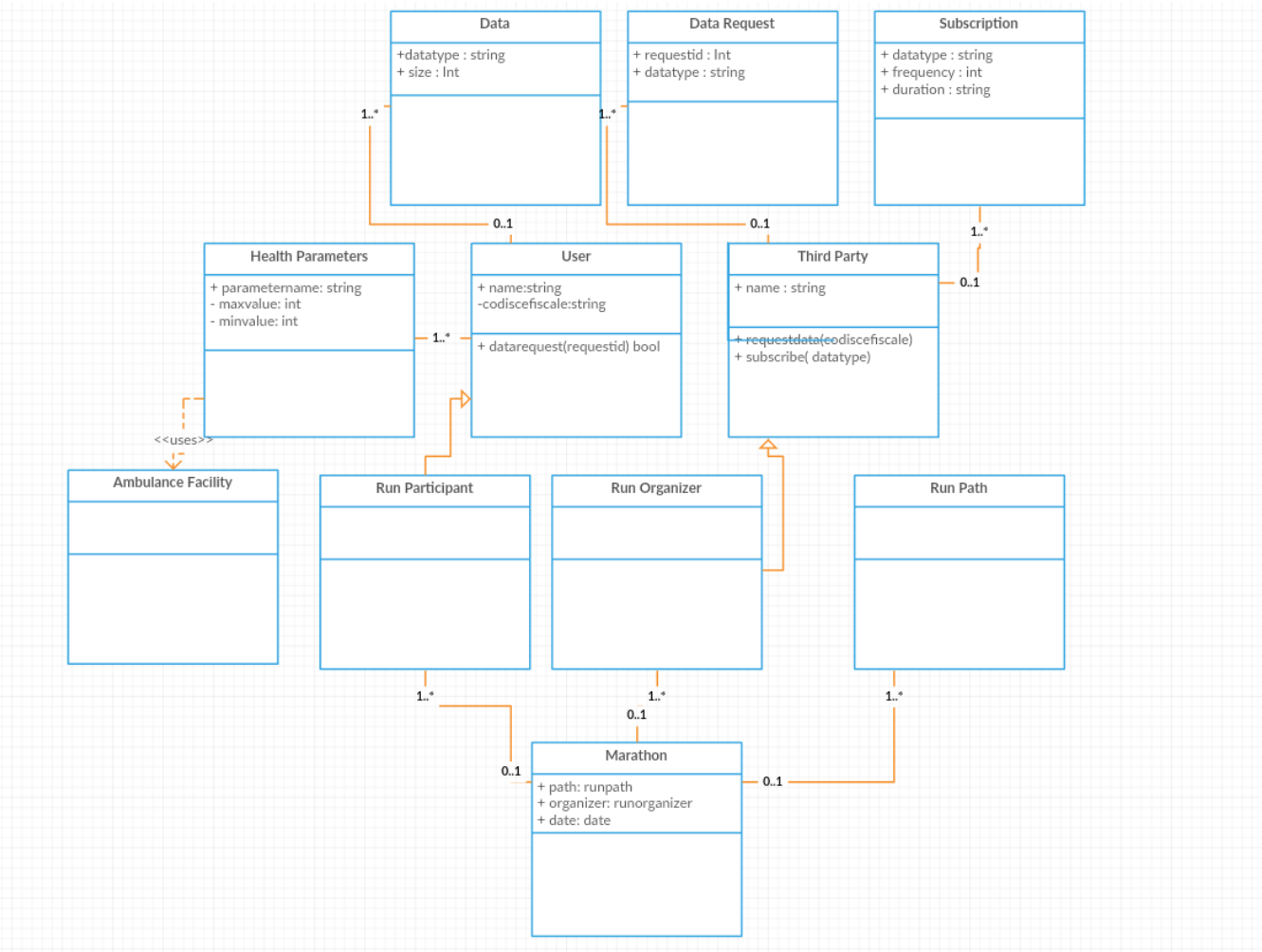


Figure 1 – Class Diagram

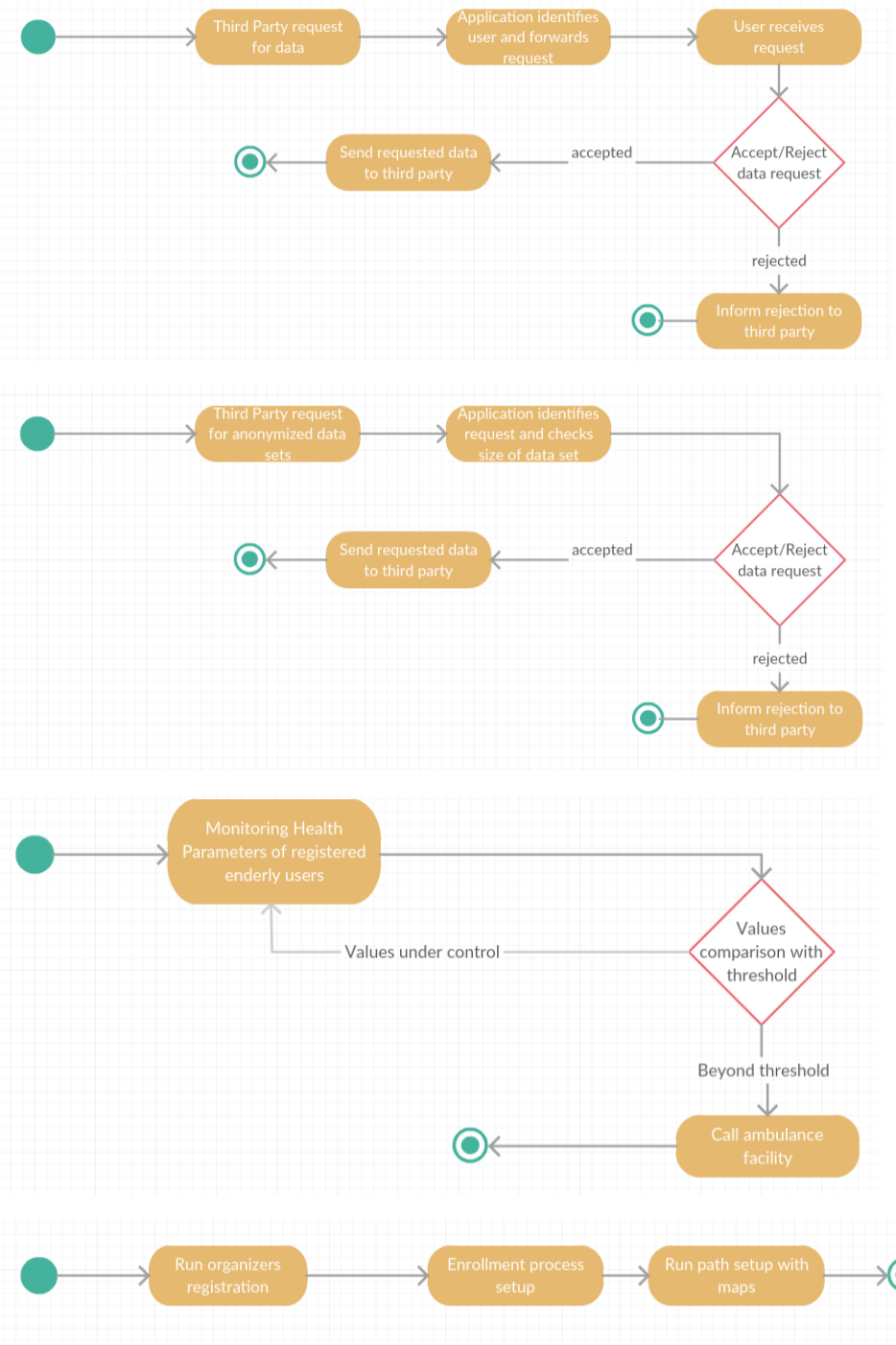


Figure 2 – Activity Diagram for all users

2.2 Product Functions

2.2.1 Data as a service

The main goal of this function is to collect data from registered users and make it available for third parties on request with due permission from the user. The app users and third parties need to register with the app via the registration portal to avail this service. Every time a third party requests for data from a specific individual, the request is forwarded to the user to confirm and the data is made available to the third party only when the user accepts the request.

2.2.2 Data set collection

This function allows the third party to request for anonymized data of group of individuals. These requests are handled by the app itself and the decision is based on the size of the data set requested in order to preserve anonymity. This requires the system to be able to process and retrieve data based on a set of given conditions. Moreover, as an added service, the third party can also subscribe to new data and the app will send it over as soon as they are available.

2.2.3 Automated SOS

As a value-added service for the user, the app provides a personalized and non-intrusive SOS service to elderly people. It collects health condition related threshold details from registered elderly users and monitors them continuously. In case the parameters go below threshold, an ambulance will be made available to the user's location in less than 5 seconds of reaction time. The app makes use of an existing ambulance facility for the same.

2.2.4 Track for run

This function helps run organizers to set up a path for a run. They can also setup an enrollment process using which the users who are willing to participate can register for the run. A map view of the run with positions of participants on the path will also be made available for run spectators to watch.

2.3 User characteristics

Basic User – A person who is registered to the app and allows his data to be collected which will be made available to third parties upon request.

Third party – Registered users who request for data from specific individuals or group of individuals using the app including subscribing for new data.

Elderly people – Registered users who subscribe for an automated SOS service by entering personal health conditions and threshold parameters.

Run organizers – Registered users who can set up a path for a run and get participants registered through an enrollment process.

Run participants – Registered users who can enroll for a run

Run spectators – Users who make use of the spectator service of the app that gives a map view of the run with the position of the participants in the run.

2.4 Assumptions, dependencies and constraints

2.4.1 Domain assumptions

D1: Users have valid codice fiscale

D2: Location of users is obtained from GPS of the user's device

D3: Users all always connected to internet

D4: Third party knows the codice fiscale of the user

D5: Facility to call an ambulance is available

D6: System uses existing map application

D7: Run participants should be registered users of Data4Help service

D8: Spectators have internet access

2.4.1 Constraints

C1: Size of data set requested for anonymized data is larger than 1000

C2: People aged above 50 are considered as elderly people

C3: Reaction time to call the ambulance facility is less than 5 seconds

2.4.3 Dependencies

The system is dependent on few external services in order to reuse existing infrastructure.

1. Ambulance facility – An existing ambulance calling facility that would allocate an ambulance to provided location based on proximity and availability.
2. Map – An existing map service that would be helpful for setting up a path for the run and show the participants on the map with their location data.

3. Specific Requirements

3.1 External interfaces requirements

3.1.1 User Interfaces

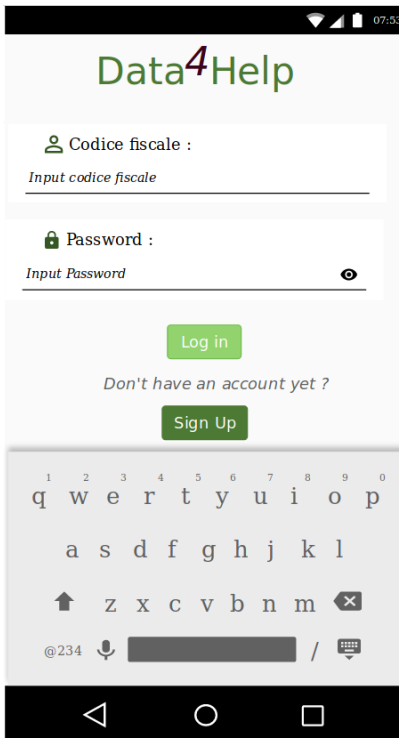


Fig 3: User registration

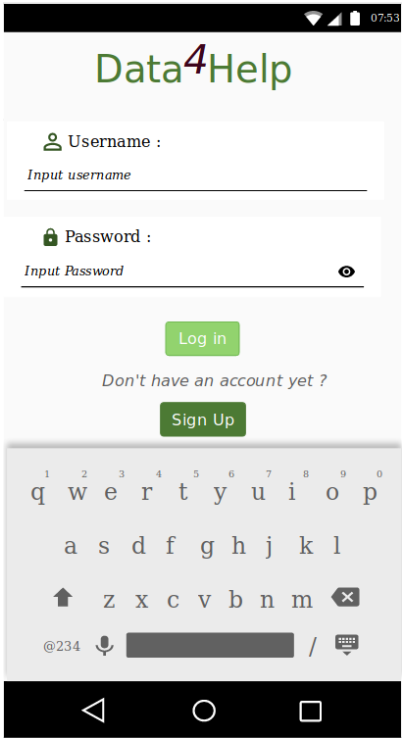


Fig 4: Third Party Registration

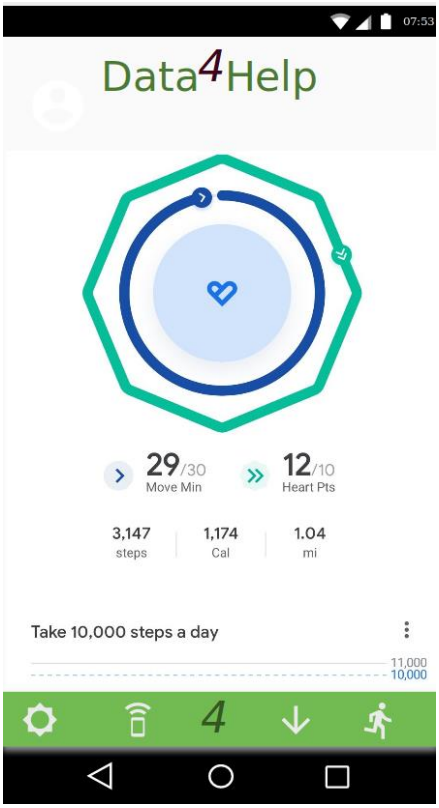


Fig 5 User Home Page

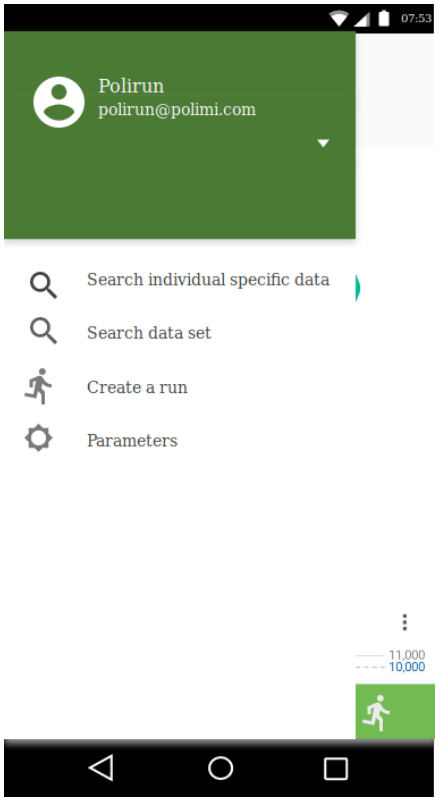


Fig 6 Third Party Menu

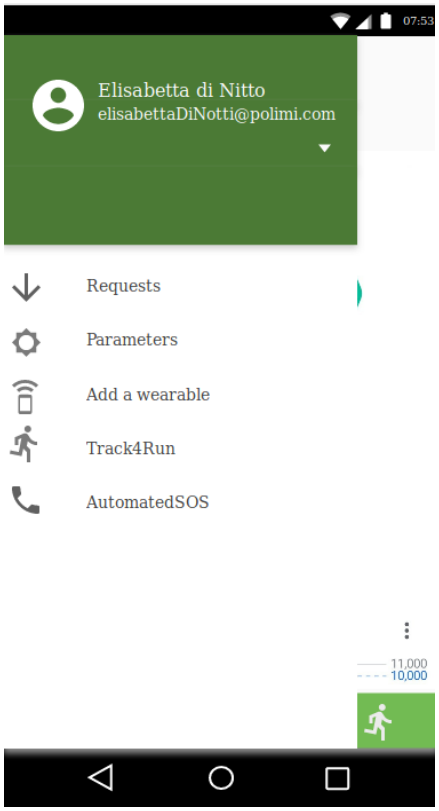


Fig 7 User Menu

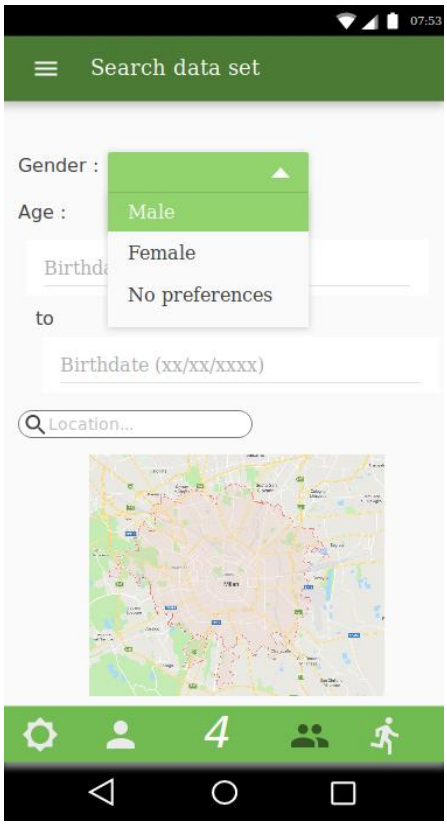


Fig 8 Search for Dataset

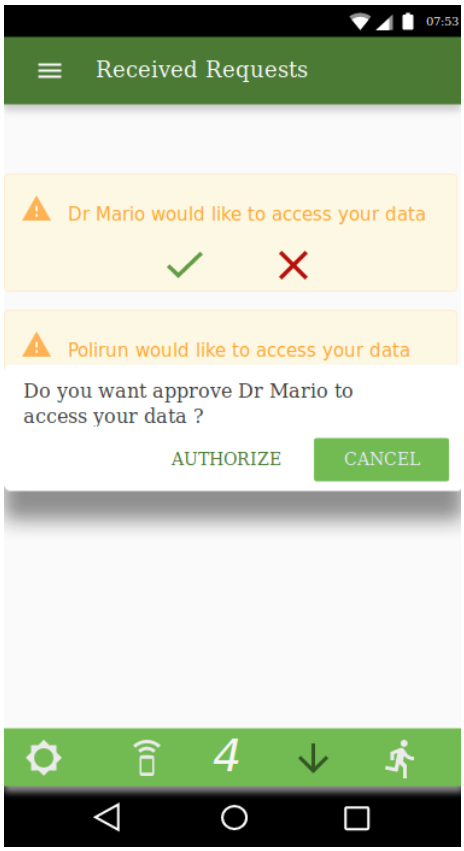


Fig 9: User approval

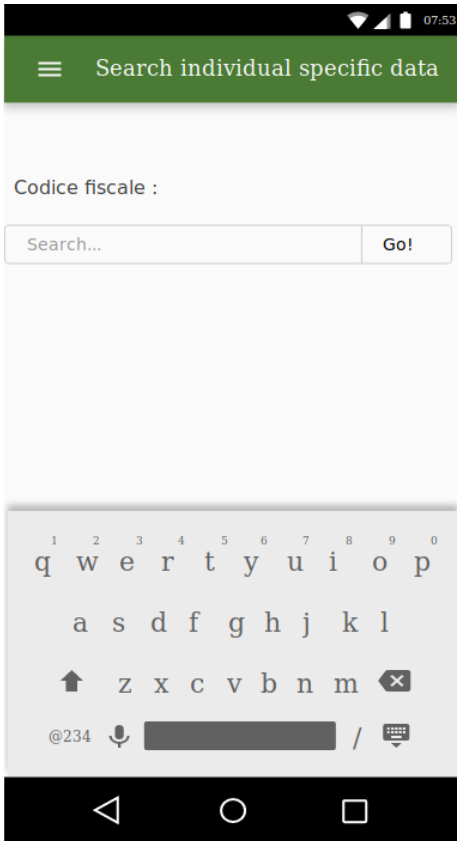


Fig 10: Search for individual specific data

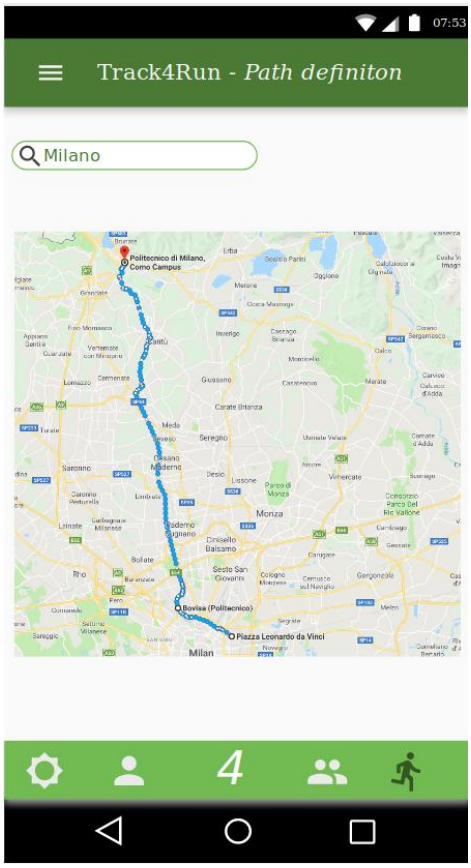


Fig 11: Run path definition

3.1.2 Hardware Interface

The software application needs at least a smartphone or a computer to be able to connect to Internet and that can use GPS Service and GSM Service.
Bluetooth and/or Wifi connections are also required to collect the health data generated by other devices (smartwatches, smart connected clothes and other wearables)

3.1.3 Software Interface

The application is using external services that are suitable for embedding in order to simplify the overall architecture and make it more lightweight.

- Collecting the health data

This application requires to collect some health data measured by one or several devices (smartphone, smartwatches, smart connected clothes etc.). There are several solutions to this requirement. One of them is using the Google Fit API that collects the data from different devices working with Android or Wear OS (Version of Google's Android operating system designed for smartwatches and other wearables).

- Geolocation

Data4Help needs to know at any time the GPS location of the users especially for the *AutomatedSOS*'s users. There exist several applications to collect the GPS location such as Google Android Device Manager API.

- Mapping and displaying of the run

Track4Run requires a map of the run planned. This map is useful for defining the path of the run and also for showing the situation of the run to spectators. An option would be to use the service Google Maps because it allows us to realize these functions in real time.

3.1.4 Communication Interface

The network-connected Android application uses HTTPS to send and receive data. The sending of HTTPS requests (encrypted requests with TLS) is important for the protection and the confidentiality of the data of the users. The Android platform includes the HttpURLConnection client, which supports TLS, streaming uploads and downloads, configurable timeouts, IPv6, and connection pooling and can be used for the development of the application.

One way to simplify the overall architecture and make it more lightweight is to use the Google APIs (see section 3.1.2). The communication with the different google services (such as Google Maps, Google Fit) is carried out by the using of GoogleApiClient ("Google API Client") object. The Google API Client provides a common entry point to Google Play services and manages the network connection between the user's device and each Google service.

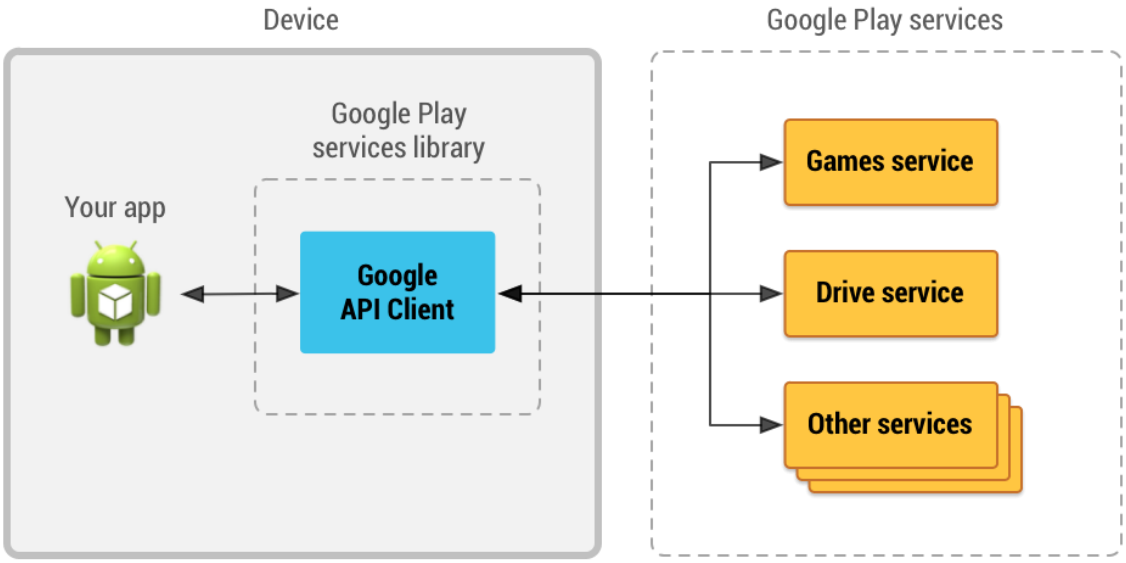


Figure 12: An illustration showing how the Google API Client provides an interface for connecting and making calls to any of the available Google Play services such as Google Play Games and Google Drive.

Thanks to the service Google Fit, we can use the service Sensors API. Sensors API provides access to raw sensor data streams from sensors available on the Android device and from sensors available in companion devices, such as wearables (smartwatches, smart clothes, ...).

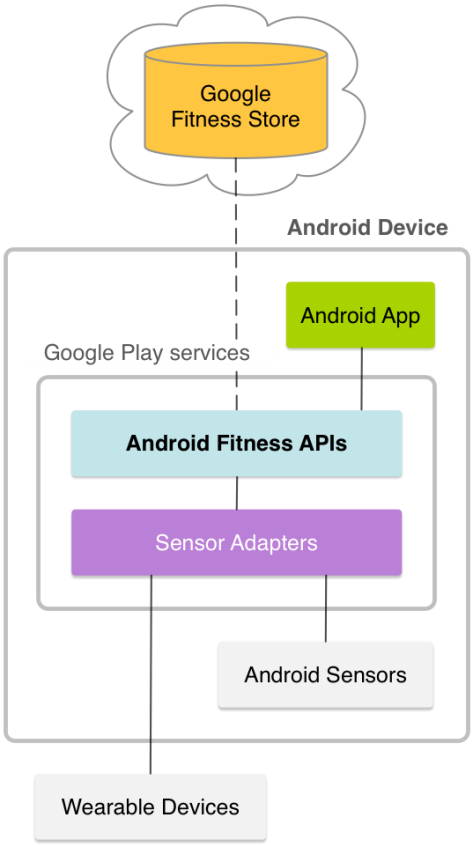


Figure 13: An illustration showing the communication between the application and the Google Fit Service.

In order to make an emergency call, *AutomatedSOS* uses the framework *Android Telecom.* The *Android Telecom* framework is responsible for managing calls on an Android device. This can include SIM-based calls using the Telephony framework, VOIP calls using SIP, or via a third-party VOIP.

3.2 Scenarios

3.2.1 Scenario 1

Luigi suffers a serious fever. However, Dr Mario, Luigi's doctor, still does not know the disease responsible of his fever and if the treatment given to Luigi will be efficient. Indeed, two treatments are possible: one if it is a food intoxication and the other one if it is viral disease. Both are subscribed to the service *Data4Help*. Luigi uses a smartwatch connected to his phone to monitor his temperature and Dr Mario can watch at any time on his phone the collected data because Luigi approved the requests of his doctor. If after some hours, the treatment given to Luigi is still not efficient, Dr Mario can give the other treatment to his patient.

3.2.2 Scenario 2

A Research Institute *4TheEnvironment,* based in Milano, would like to study the effects of pollution on the health of young adults. For the realization of this study, *4TheEnvironment* decides to use the service *Data4Help,* when the pollution is above a threshold defined. With this service *4TheEnvironment* can collect some health data such as the temperature, the cardiac rhythm, the blood pressure, the dehydration state measured by smartphones, smartwatches or other wearables on the young adults of 18-25 years old. *4TheEnvironment* collects and saves these data when they are above a threshold defined by the research institute. The number of people targeted is important enough (higher than 1000 in Milano) to guarantee the confidentiality of these people and therefore *4TheEnvironment* can realize its study.

3.2.3 Scenario 3

Chiara is 98 years old and still lives alone in her home. She suffers of some cardiac problems and at this age the chance of heart attacks is high. Her daughter, Lucia, has decided to register Chiara to the service *AutomatedSOS.* With a wearable (that she connected to the app), the cardiac rhythm is sent in real-time to *AutomatedSOS*. Yesterday, Chiara was victim of a heart attack and the service detected the attack, a few seconds after its beginning and sent an ambulance to the location of Chiara's home few minutes after. Thanks to *AutomatedSOS* and its fast response time Chiara is still alive.

3.2.4 Scenario 4

Stefania takes part in Polimirun that is a run organized by Politecnico di Milano. This year, the organizers have decided to use the service *Track4Run* to define the path of the run and invite the different participants. The organizers asked all participant to measure their cardiac rhythm with a wearable. Like Stefania, each runner subscribing to Track4Run, must be equipped with a wearable measuring the cardiac rhythm and must allow the access to these data by the organizers to participate in Polimirun. For the run with this service, Stefania's family can access in real time, her location and follow her at some key points to support her. At the middle of the run, Stefania has a loss of consciousness. Her cardiac rhythm is irregular and abnormally low, the problem is detected by the service and the organizers. A medical unit is immediately sent to her location.

3.3 Functional requirements

[G1] - Collect data (location, health status: cardiac rhythm, steps) from all registered users

R1: System should allow registration of users with codice fiscale and personal data (age, gender, blood type)

R2: System should collect and store date of the registered users

D1: Users have valid codice fiscale

D2: Location of users is obtained from GPS of the user's device

D3: Users are always connected to internet

[G2] Registered third parties can access data from individual users with permission

R3: System should allow registration of third party with valid ID

R4: System should pass requests from third party to individual based on codice fiscale

R5: System should make the requested data available to the third party only if the individual approves the request

D4: Third party knows the codice fiscale of the user

[G3] Registered third parties can request for anonymized data of groups of individuals

R6: System should be able to retrieve data based on category

R7: System should be able to accept or refuse a request based on the size of data

[G4] Registered first parties can subscribe and receive data

R8: System should allow subscription for data requests from third party

R9: System should send requested data as soon as they are produced

[G5] registered elderly users can subscribe for a personalized and non-intrusive SOS service

R10: System should allow subscription for elderly people by entering personal preferences (thresholds for health parameters)

[G6] Call an ambulance for subscribed elderly people if needed

R11: Monitor health parameters of subscribed elderly people continuously

R12: Call an ambulance only when health parameters go below threshold

D5: Facility to call an ambulance is available

[G7] Run organizers can set up a path for the run

R13: System should allow registration of run organizers

D6: System uses existing map application

[G8] run organizers can set up enrollment process

R14: System should allow creating an enrollment process

[G9] run participants can enroll for a run

R15: System should allow run participants to enroll for a run

D7: Run participants should be registered users of the Data4Help service

[G10] showing the situation of the run to spectators

R16: system should create a view of the map with all run participants positions

R17: system should allow spectators to have access to the view

D8: spectators have internet access

3.3.1 Use case diagram

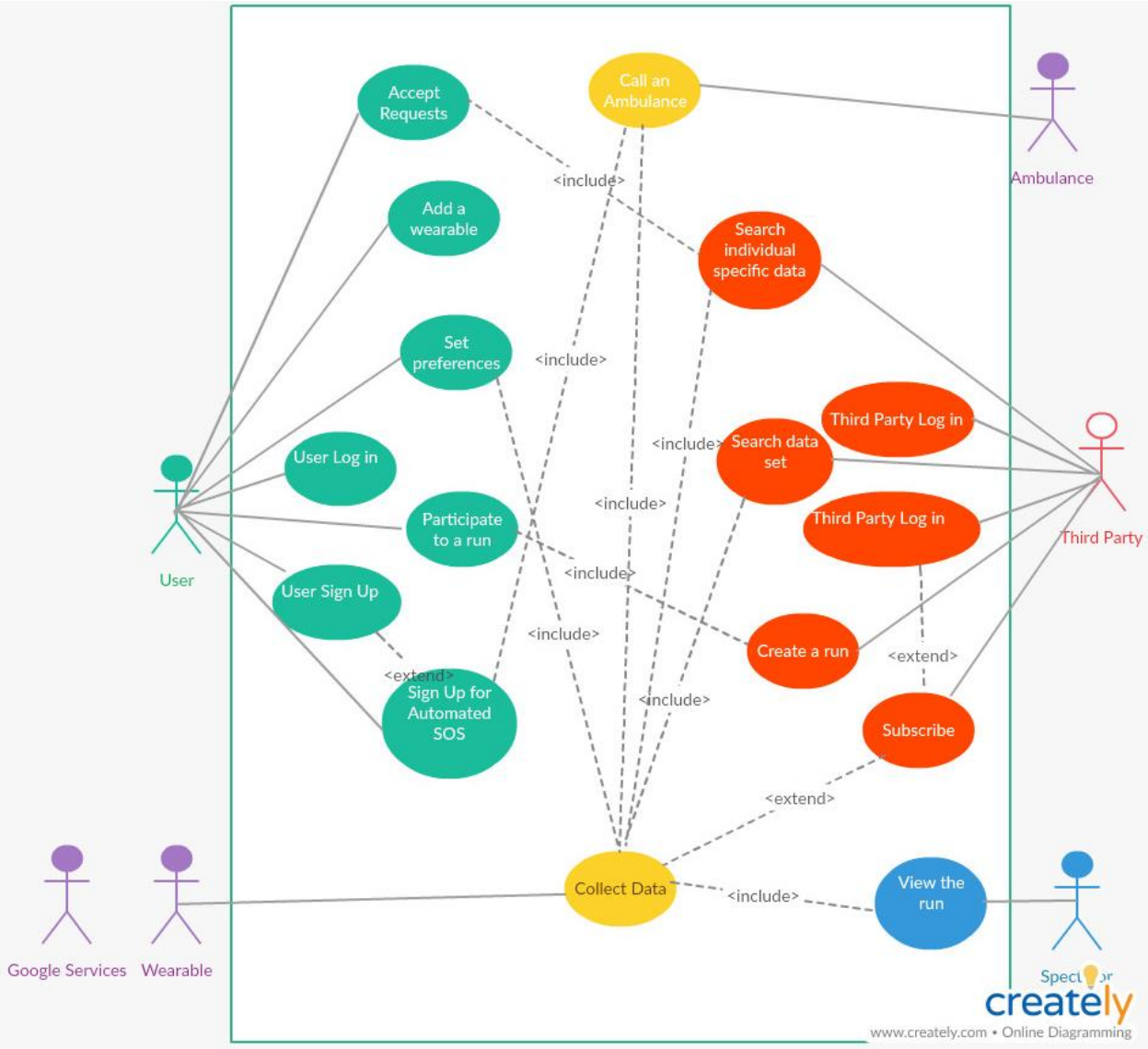


Figure 14: Use Case Diagram

3.3.2 Use cases

Name	**User Sign Up**
Actor	User
Entry conditions	The user has installed the application on her/his device.
Events flow	1. The user clicks on the « Sign Up » button. 2. The user gives her/his codice fiscale. 3. The user fills all the mandatory fields and provides the necessary information. 4. The user clicks on « Confirm button ». 5. *Data4Help* saves the data.
Exit conditions	The user has successfully registered and now she/he is able to use the application.
Exceptions	1. The user is already signed up. 2. The user did not fill all of the mandatory fields with valid data. 3. The codice fiscale is already taken. 4. The e-mail is already registered. 5. All the exceptions are handled by notifying the user and taking him back to the sign-up activity.

Name	**User Log in**
Actor	User
Entry conditions	The user has successfully signed up and has the application installed on her/his device.
Events flow	1. The user opens the application on her/his device. 2. The user enters her/his codice fiscale in the “Codice fiscale” and “Password” fields of the log in page of *Data4Help.*

	3. The user clicks on the “Log in” button. 4. The user is successfully logged in her/his “Data4Help” and the system automatically redirects her/him to the home page.
Exit conditions	The user is successfully redirected to the home page.
Exceptions	1. The user enters an invalid codice fiscale. 2. The user enters an invalid Password. 3. All the exceptions are handled by notifying the user and taking her/him back to the login activity.

Name	**Accept requests**
Actor	User
Entry conditions	The user has already logged in.
Events flow	1. The user clicks on the button “Requests”. 2. The user clicks on the button “Approve” to accept each request sent. Otherwise, he clicks on the Button “Disapprove”. 3. The user clicks on the “AUTHORIZE” button to confirm her/his choice. 4. The system saves the preferences.
Exit conditions	User preferences are saved to her/his profile and are considered.
Exceptions	There are no requests and the user is notified with an empty list of requests.

Name	**Add a wearable**
Actor	User
Entry conditions	The user has already logged in
Events flow	1. The user clicks on the button “Wearables”. 2. The user clicks the option “Add a wearable”.

	3. The user chooses the wearable in the list proposed by *Data4Help*. 4. The user confirms her/his choice on clicking one the button "Associate". 5. The system saves the new wearable.
Exit conditions	The new wearable is saved to her/his profile and is considered.
Exceptions	1. The system finds no wearables and notifies the user with an empty list of wearables. 2. The system does not succeed to associate the wearable and notifies the user of its failure.

Name	**Set preferences**
Actor	User
Entry conditions	The user has already logged in.
Events flow	1. The user clicks on the button "Preferences". 2. The user clicks on a button representing a wearable connected with the system. 3. The user chooses the set of health parameters measured by the wearable. 4. The system saves the preferences.
Exit conditions	The preferences are saved to her/his profile and are considered.
Exceptions	1. The system finds no wearable connected to the service and notifies the user. 2. The system does not find any health parameters that can be measured by the wearable and notifies the user.

Name	**Collect data**
Actor	*Data4Help*

Entry conditions	1. One or more wearables are associated with the system. 2. One or more wearables can collect some health data.
Events flow	1. *Data4Help* gathers all data provided by external services and wearables. 2. *Data4Help* saves data on the servers.
Exit conditions	The health data are saved on the servers.
Exceptions	1. There are problems in the collecting of health data and *Data4Help* notifies the user.

Name	**Third party Sign Up**
Actor	Third party
Entry conditions	The third party has installed the application on its device
Events flow	1. The third party clicks on the « Sign Up » button. 2. The third party fills all the mandatory fields and provides the necessary information. 3. The third party clicks on « Confirm button ». 4. *Data4Help* saves the data.
Exit conditions	The third party has successfully registered and now it is able to use the application.
Exceptions	1. The third party has already signed up. 2. The third party did not fill all the mandatory fields with valid data. 3. The username is already taken. 4. The e-mail is already registered. 5. All the exceptions are handled by notifying the third party and taking him back to the sign-up activity.

Name	**Third party Log in**
Actor	Third party
Entry conditions	The third party has successfully signed up and has the application installed on her/his device.
Events flow	1. The third party opens the application on its device. 2. The third party enters its credentials in the “Username” and “Password” fields of the log in page of “Data4Help”. 3. The third party clicks on the “Log in” button. 4. The third party has successfully logged in its *Data4Help* and the system automatically redirects it to the home page.
Exit conditions	The third party is successfully redirected to the home page.
Exceptions	1. The third party enters an invalid Username. 2. The third party enters an invalid Password. 3. All the exceptions are handled by notifying the third party and taking its back to the login activity.

Name	**Search individual specific data**
Actor	Third party
Entry conditions	The third party has already logged in.
Events flow	1. The third party clicks on the button “Search”. 2. The third party writes the codice fiscale of the targeted user. 3. The third party chooses the health data that it wants to collect. 4. The system sends a request to the user if it is the first time that the third party wants to collect these data. 5. If the user allows the request, the third party can collect the chosen health data.
Exit conditions	The health data of a specific individual are collected.
Exceptions	1. The codice fiscale is not valid, the third party is notified that the code does not work, and the third party is taken back to the login activity.

	2. The request is rejected, the third party is notified and is taken back to the login activity. 3. The request is accepted but there are no data, the user is notified and is taken back to the login activity.

Name	**Search data set**
Actor	Third party
Entry conditions	The third party has already logged in
Events flow	1. The third party clicks on the button “Search set of data”. 2. The third party chooses the criteria that define the targeted population (such as age, sex, locations…). 3. The third party selects the health data that it wants to collect on this population. 4. If the size of the targeted population giving these data is larger than 1000, the request is accepted. 5. The third party can collect the requested health data of the targeted population.
Exit conditions	The health data of a targeted population are collected.
Exceptions	1. The size of the population is less than 1000, the third party is notified and is taken back to the login activity.

Name	**Subscribe**
Actor	Third party
Entry conditions	The third party has already logged in
Events flow	1. The third party clicks on the button “Parameters”. 2. The third party clicks on the button “Subscription for data in real-time”

	3. The third party fills all the mandatory fields and provides the necessary information. 4. The system saves the subscription.
Exit conditions	The subscription is saved.
Exceptions	1. The third party has been already subscribed, the third party is notified and taken back at the home page.

Name	**Sign up for AutomatedSOS**
Actor	User
Entry conditions	The user has already logged in.
Events flow	1. The user clicks on the «AutomatedSOS» button. 2. The user confirms his/her subscription to *AutomatedSOS.* 3. The user fills all the mandatory fields and provides the necessary information. 4. The user provides her/his location and/or the device that can locate her/him. 5. The user specifies the health data to monitor. 6. The user defines a threshold for each health data. 7. *AutomatedSOS* saves the data.
Exit conditions	The user has successfully registered to *AutomatedSOS* and the thresholds are saved.

Exceptions	1. The user is already signed up. 2. The user did not fill all the mandatory fields with valid data. 3. The user does not provide her/his location or a device that can locate her/him. 4. The user does not specify any health data. 5. The chosen health data cannot be measured by any wearables. 6. The user does not specify any thresholds. 7. The chosen thresholds are not valid. 8. All the exceptions are handled by notifying the user and taking him back to the home page

Name	**Call an ambulance**
Actor	*AutomatedSOS*
Entry conditions	1. The user is registered to *AutomatedSOS.* 2. The health data collected are below the thresholds defined.
Events flow	1. The health data collected are below the thresholds defined. 2. The system gets the user's location with the location given by the user or by a wearable connected to her/him. 3. A request with the user's location and health data is sent to a medical service. 4. The request is treated by the medical service. 5. The system notifies the user that a medical service is called. 6. A request is sent by the medical service to confirm the ambulance. 7. The system notifies the user that an ambulance is coming to her/his location.
Exit conditions	A request sent by the medical service to confirm the ambulance.

Exceptions	1. The health data are already higher than the threshold for 5 seconds, *AutomatedSOS* stops its procedure and goes back to the entry conditions after notifying the user. 2. The system does not receive any request of the medical service after 1 minute. A new request with the user's location and health data is sent to the medical service until having an answer.

Name	**Create a run**
Actor	Third party
Entry conditions	The third party has already logged in.
Events flow	1. The third party clicks on the "Track4Run" button. 2. The third party chooses the criteria of the target population participating in the run (age, location, sex, …). 3. The third party searches the location where the run will take place. 4. The third party defines the steps of the run on the map. 5. The Third party chooses health data that can be collected. 6. The third party chooses if this collecting is mandatory. 7. The system sends an invitation to all people satisfied by the criteria.
Exit conditions	The system sent the invitations.

Exceptions	1. The system does not find the location of the run. 2. The third party does not provide a correct path of the run. 3. No one was found according to the chosen criteria. 4. All the exceptions are handled by notifying the third party and taking him back to the *Track4Run* activity.

Name	**Participate to a run**
Actor	User
Entry conditions	The user has already logged in.
Events flow	1. The user clicks on the “Track4Run” button. 2. The user accepts the invitation(s). 3. The user fills all the mandatory fields and provides the necessary information. 4. The user allows the health data collected for the run.
Exit conditions	The user has successfully registered.
Exceptions	1. The user declines the invitation and the invitation is deleted and the user is taken back to the home page. 2. The user does not specify any health data. 3. The user does not allow the organizers to collect the health data, if the collecting is mandatory the user is notified, and the registration is still incomplete, so the user goes back to the step 4.

Name	**View the run**
Actor	Spectators
Entry conditions	The spectator has installed the application on her/his device.
Events flow	1. The spectator clicks on the “Track4Run” button. 2. The spectator searches a run. 3. The spectator visualizes the run with the location of all participants.
Exit conditions	The map is displayed.
Exceptions	1. The searched run does not exist, the user is notified and she/he is taken back to the Track4Run page.

3.3.3 Sequence Diagram

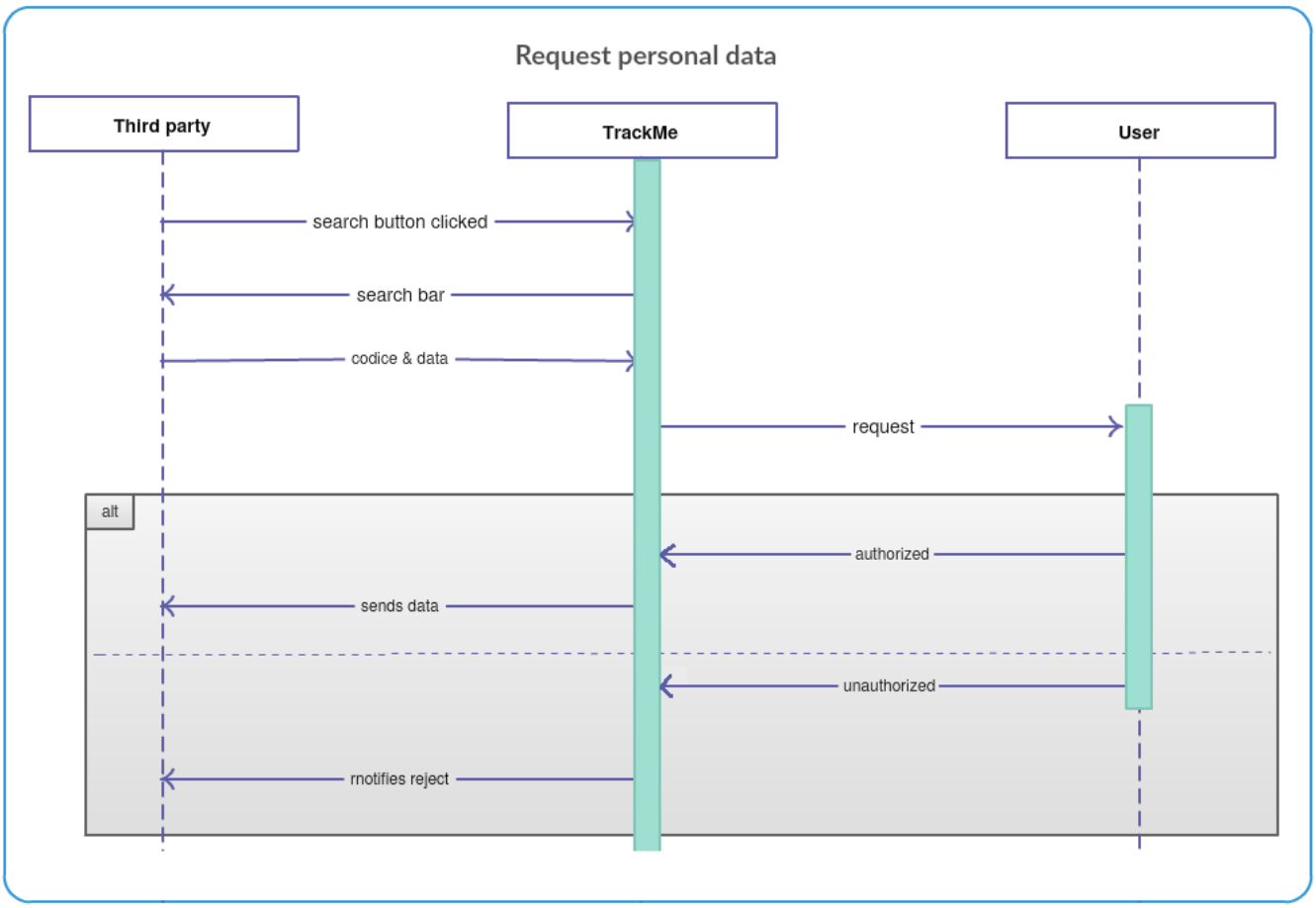


Figure 15: Sequence Diagram to show the flow for individual data request made by third party

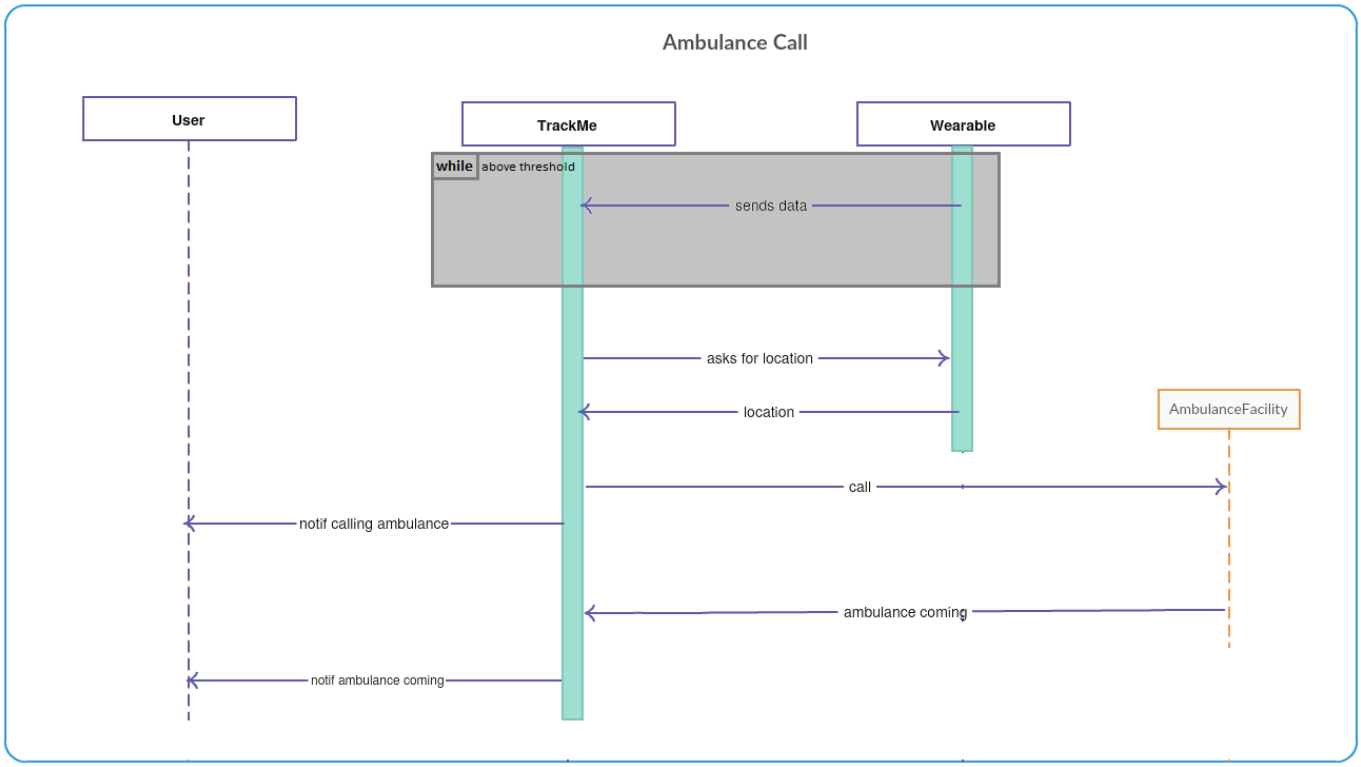


Figure 16: Sequence Diagram to show the flow for AutomatedSOS

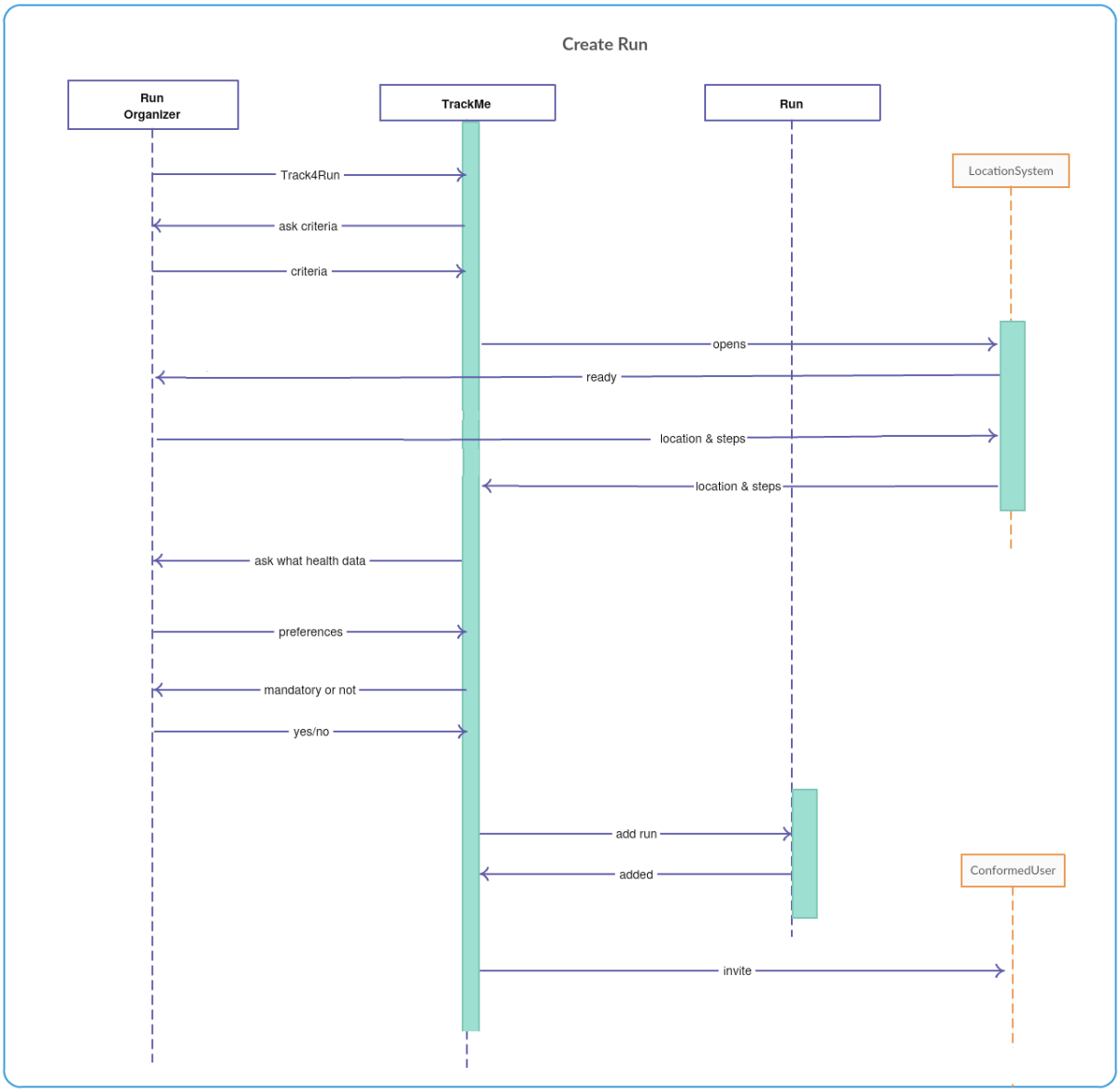


Figure 16: Sequence Diagram to show the flow for Track4Run

3.4 Performance Requirements

The system handles real time data and with the added AutomatedSOS service the reaction time expected is less than 5 seconds for the ambulance. Thus, the system is expected to be responsive and available with very quick recovery in the case of failure. Among registered users, the users registered for AutomatedSOS will get higher priority. Overall, the system is expected to support 45,000 registered users including third party registrations.

3.5 Design Constraints

3.5.1 Standards Compliance

1. The system should maintain privacy of data as mentioned. Individual data requests should be approved by user and anonymized data sets should be handled with discretion with respect to size.
2. User should be informed of the data collected and declaration should be provided while registration.
3. Health parameters related data should be handled in accordance with medical standards
4. Data between the three services should not be aggregated unless requested.

3.5.2 Hardware limitations

The main hardware limitation of the system is the requirement for a smartphone with GPS enabled and internet connectivity at all times. In case of subscription for AutomatedSOS service the device should be able to collect the required health parameters and make them available for the system to monitor the user.

3.6 Software System Attributes

3.6.1 Reliability

The application must be available 24/7. Negligibly small concessions from this requirement might be tolerated.

3.6.2 Availability

In order to guarantee high degree of availability, system of redundant servers may be considered. This way, if possibly on server fail, the other one will be ready to take over. The system is expected to be available 99.9% of the time.

3.6.3 Security

Health data is very sensitive information, such as user passwords, and it they should be confidentially stored and encrypted with high-security encryption. The health issue makes the security of the data highly important, as much as the security of the communications user-TrackMe and TrackMe-third party.

3.6.4 Maintainability

As previously mentioned, the application is going to be flexible and easy to maintain, i.e. capable to facilitate addition of new features and options. For that purpose, we will use clear code following the design patterns.

Design patterns provide a standard terminology and are specific to a scenario. For example, a singleton design pattern signifies use of single object so all developers familiar with single design pattern will make use of single object and they can tell each other that program is following a singleton pattern.

Of course, complete and detailed documentation will also be provided in order to keep the maintainability on the highest level.

3.6.5 Compatibility

This is an Android application, so it needs to be compatible with as many devices as possible, while still implementing all the requirements defined. It should be able to run on a wide variety of devices and circumstances, especially on the wearable devices running on wear OS (in case some users want to register directly on it) because some Google services are used to collect the data.

4.FORMAL ANALYSIS USING ALLOY

Alloy model has been used to describe the system by specifically mentioning the various entities and their relationships. The alloy facts are used to describe the various constraints in the system and their dependencies with each other.

1. Every individual data request will be attached with a user with the help of the unique identifier, codice fiscale.
2. Every Data request has size constraints to ensure anonymity of the users whose data is being shared by the system
3. Every user specific data request has to be approved by the user for it to be made available to the requested third party
4. Ambulance facility is called when health parameters go beyond threshold only for the registered users who have subscribed to AutomatedSOS
5. Every registered user needs a wearable that collects data for the system to be able to collect the same
6. Availability of ambulance is checked before assigning the user's location to the ambulance

4.1 ALLOY MODEL

```
module trackme


sig CodiceFiscale {}


sig Preference {}


sig Wearable{
   CollectedHealthData: some HealthData
}


sig Ambulance{
```

```
    target: lone User,
    available: one Int
}

sig HealthDataType{}
sig HealthData{
    healthDataTypes: set HealthDataType,
    data: one Int
}

sig Threshold{
    value: one Int,
    healthDataType: one HealthDataType
}

sig AutomatedSOS{
    subscribed: one Int,
    thresholds:some Threshold
}

sig User{
    codiceFiscale: one CodiceFiscale,
    wearables: some Wearable,
    preferences: set Preference,
    warnings: set Warning,
    healthData: set HealthData,
    location: one Location,
```

```
    automatedSOS: one AutomatedSOS,
    receivedRequests: set Request
}

sig ThirdParty{
    dataSetRequests: set DataSetRequest,
    individualRequests: set IndividualRequest,
    marathons: set Marathon,
    subscription: one Bool,
    rejections: set Rejection

}

sig Date{}

sig RunPath {
    steps: some Location
}
{ #steps> 2 }

sig Location{
    coordX: one Int,
    coordY: one Int
}


sig Marathon{
```

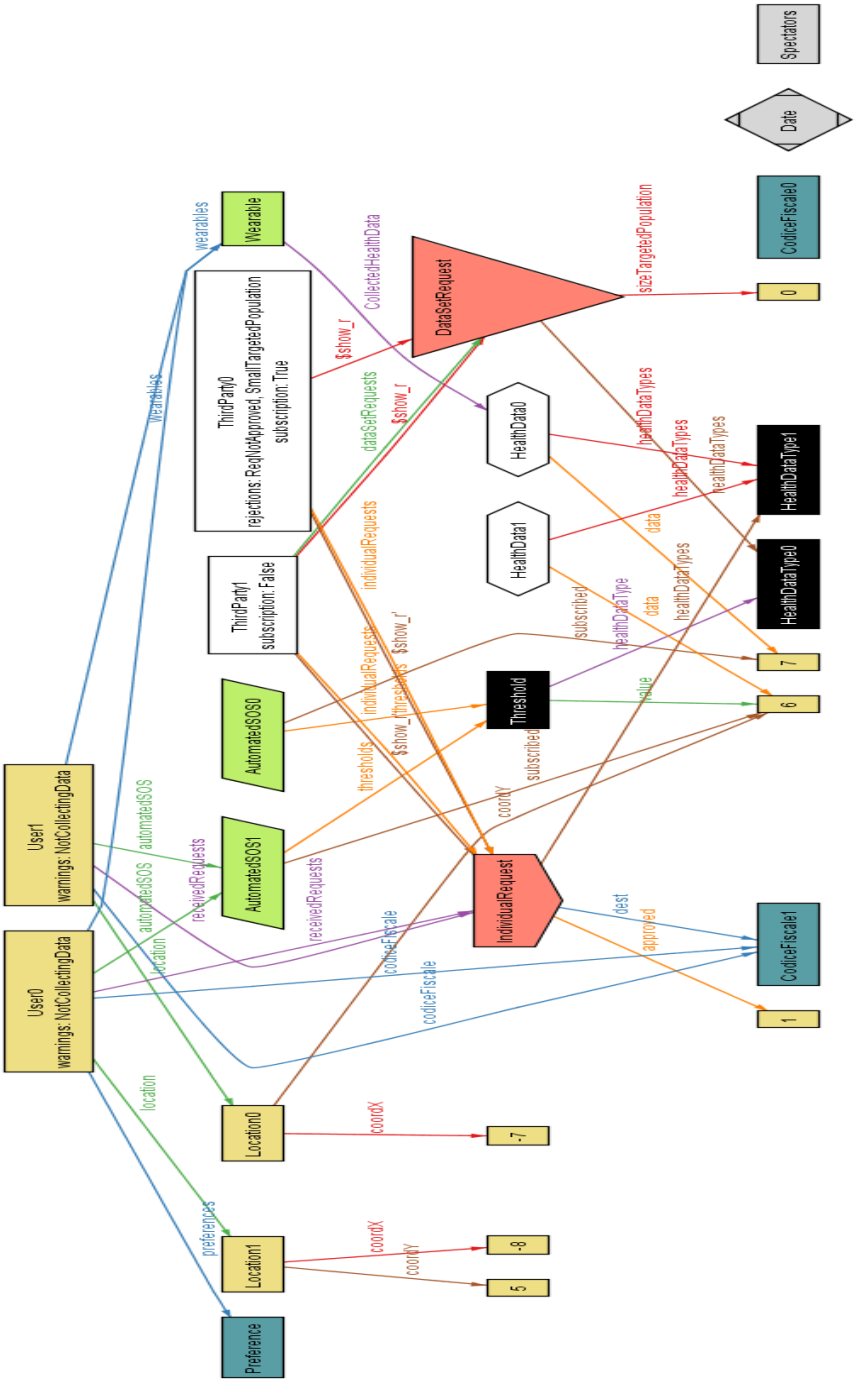
```
    date: one Date,
    runPath: one RunPath,
    participants: some User,
    organizer:one ThirdParty
}

sig Spectators{
    marathons: set Marathon
}

abstract sig Warning {}
one sig NotCollectingData extends Warning{}
one sig NoWearableAvailable extends Warning{}

abstract sig Rejection{}
one sig SmallTargetedPopulation extends Rejection{}
one sig WrongUser extends Rejection{}
one sig ReqNotApproved extends Rejection {}

abstract sig Bool{}
one sig True extends Bool{}
one sig False extends Bool{}

abstract sig Request{
    healthDataTypes: some HealthDataType
}
```

```
sig IndividualRequest extends Request{
   dest: one CodiceFiscale,
   approved: one Int
}
sig DataSetRequest extends Request{
   sizeTargetedPopulation: one Int,
   targetedUsers: set User
}


fact sentIndividualRequest{
   (all r: IndividualRequest | all u: User | r.dest=u.codiceFiscale implies u.receivedRequests=r)
}


fact smallTargetedPopulationCheck{
   all t:    ThirdParty | some r:DataSetRequest | r.sizeTargetedPopulation<1000 implies
SmallTargetedPopulation in t.rejections
}


fact approvedIndividualRequest{
   (all t: ThirdParty | some r: t.individualRequests | r.approved!=1 implies ReqNotApproved in
t.rejections)
}

fact whenCallAnAmbulance{
```

```
   (all u: User | all h:u.healthData | all t:u.automatedSOS.thresholds | one a:Ambulance | u.automatedSOS.subscribed= 1 and h.healthDataTypes=t.healthDataType and h.data<t.value and a.available=1 implies a.target=u and a.available=0)
}

fact wrongUserCheck{
   all t:ThirdParty | no u: User | t.individualRequests.dest=u.codiceFiscale implies WrongUser in t.rejections
}

fact wearableCheck{
   all u:User | ((no u.wearables) implies NoWearableAvailable in u.warnings)
}

fact dataCheck{
   all u:User | ((no u.healthData) implies NotCollectingData in u.warnings)
}

fact factAmbulance{
   (all a : Ambulance | a.available=1 and a.target=none)
}

assert ambulanceCall{
   all u: User | u.automatedSOS.subscribed!=1 implies (no a: Ambulance | a.target=u)
}

assert thirdPartygetsdata {
```

```
    all t:ThirdParty | some r: t.individualRequests | r.approved!=1 implies ReqNotApproved in
t.rejections
}


run show for 2 but 2 User, 2 ThirdParty
check thirdPartygetsdata
check ambulanceCall
pred show() {}
```

4.2 WORLD GENERATED



4.3 ALLOY RESULTS

```
Executing "Check ambulanceCall"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7434 vars. 696 primary vars. 16160 clauses. 41ms.
  No counterexample found. Assertion may be valid. 6ms.
```

```
Executing "Check thirdPartygetsdata"
  Solver=sat4j Bitwidth=4 MaxSeq=4 SkolemDepth=1 Symmetry=20
  7349 vars. 693 primary vars. 15962 clauses. 34ms.
  No counterexample found. Assertion may be valid. 3ms.
```

```
Executing "Run show for 2 but 2 User, 2 ThirdParty"
  Solver=sat4j Bitwidth=4 MaxSeq=2 SkolemDepth=1 Symmetry=20
  3809 vars. 408 primary vars. 8220 clauses. 18ms.
  Instance found. Predicate is consistent. 28ms.
```

5.EFFORT SPENT

Work Description	**Sankari Gopalakrishnan**	**Louis Lesieur**	**David Brellmann**
Purpose, scope, definitions	4	5	4
Product Perspective and Product functions	6	5	5
Domain Assumptions and constraints	5	5	5
External interface requirements	6	7	8
Functional /Non-Functional Requirements	6	5	5
Formal analysis using alloy	8	5	8

6.REFERENCES

Specification Document “Mandatory Project Assignment AY 2018-2019”

Alloy Documentation http://alloy.lcs.mit.edu/alloy/documentation.html

Slides - “Usage of Alloy in RE”