POLITECNICO DI MILANO

Department of Electronics, Information and Biomedical Engineering

MSc. Course in Computer Science and Engineering

Software Engineering 2

TrackMe

Requirements Analysis and Specification Document

Instructor: Prof. Matteo ROSSI

Authors: Buse GUNAY   (matr. 903822)

                Ezgi TASTI   (matr. 917104)

                Talip TURKMEN   (matr. 915609)

11.11.2018

Politecnico di Milano

Table of Contents

Abstract

The purpose of this document is to specify the requirements that our system needs to fulfill to adopt the ISO/IEC/IEEE 29148 dated Dec 2011 standard for RASD documentation.

The document includes the functional and non-functional requirements via UML diagrams (use case diagram, class diagram, sequence diagram, state diagram and activity diagram). Also, the document presents a high-level specification of the overall system. And for the last part of the document, an alloy model is presented for modeling specifications of system.

Chapter 1

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 detailed presentation of the proposed solution and its purpose, listing its goals, and the requirements and assumptions through which they will be achieved. The document is meant to be used by individuals such as elderly people and runners; and also by the third 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 to facilitate the monitoring of the health status of people and to transmit information to third parties. Also, it ensures that information is kept and shared both individually and anonymously.

Health status monitoring is critical for people who are particularly sensitive to health, want to share health information with their hospital and want to view this information in detail. If necessary, it is vitally important to direct a health service to an individual's position. TrackMe offers all these services together and makes it easier for individuals to track their health status for both themselves and third parties.

TrackMe also provides a platform for running. Thanks to this application, the organizers can specify the run and define the path for the run, while at the same time the spectators can easily track the runners on the map.

1.2. Scope

1.2.1. Description of the Given Problem

The TrackMe system is designed to provide individuals to track their location and health status regularly and, if necessary, to ensure that it is forwarded to third parties. TrackMe supports three different services such as Data4Help, AutomatedSOS and Track4Run.

In Data4Help, when the individual register to the system, the system takes personal information of them for instances SSN, e-mail, gender, chronic disease that have and nationality. In addition, an individual's location, pulse, blood rate and heart rate is taken with the help of sensors on the smart device. These data are stored in a database and shared with third parties if requested by them within the permission of individuals, such as a hospital. On the other hand, this system supports access to anonymized data of groups of individuals when the identity of them should not be identified by these third parties. At the same time, individual will be able to follow own data via the application and to approve or reject the third parties that want to use own data.

AutomatedSOS, the second service offered by TrackMe, is more engaging for especially elderly people. The data of individuals registered to AutomatedSOS are tracked real-time. The values, such as pulse meter, accelerometer, gyroscope and barometer, related to the health data of individuals are taken and controlled by thresholds. When the health data of these people fall below a certain threshold, the health status and location of them are transmitted to the nearest hospital. An ambulance is sent to an individual by providing the necessary service by the hospital. After the SOS notification is sent, individuals' health status, location, notification time and which the third party is shared with this information keep on TrackMe side and the individual will be able to view it in history at any time.

Another service is Track4Run supports to organizers define a path for the run. Organizers define the running date and location on TrackMe. Runners register to organization and track own path and position in real-time. The instant positions of the runners are shown to the an organizers and spectators on the map.

1.2.2. Analysis of World and Shared Phenomena

World	Shared Phenomena	Machine
- Having a smart device. - Having a unique SSN. - Making the monitoring data to receive. - Measuring of values (pulse, velocity, etc.) with sensors. - Measuring health data and location by smart device. - Dispatching ambulance. - Defining threshold values. - Organizing a run.	- Receiving individuals' profile data. - Receiving health and location data from smart device. - Transmission of the received data to the application. - Requesting personal or anonymized data. - Defining pre-confirmed list. - Notifying individuals about data sharing request. - Approve or reject data sharing request. - Notifying third party about data request rejection. - Showing requested data. - View data share history. - View own data. - Notifying third party (e.g. hospital) - Defining the path. - Enrolling to run. - Tracking runners.	- Controlling SSN. - Storing the received data in database. - Controlling anonymization. - Checking pre-confirmed list. - Preparing requested data. - Storing data request. - Comparing measured values with sensors and threshold values. - Storing SOS information.

1.2.3. Goals

There are 3 main services running under TrackMe and the possible goals for each service can be listed as;

Data4Help

For Individuals

- [G1] Individuals can collect and store their data on Data4Help. The desired data from individuals are profile data, health data, location data and chronic disease.
- [G2] Individuals can share their data with third parties which they allowed.
- [G3] Individuals can update their personal information they have used while registering and delete their own account.
- [G4] Individuals can see and select which information is shared with which third parties when information of these individuals is shared.
- [G5] Individuals can see and track their own data.

For third parties

- [G6] Third parties can access specific individuals' data.
- [G7] Third parties can access anonymized data of a group of individuals.

AutomatedSOS

- [G8] Sends a notification to the nearest hospital when health values of individual fall below a certain threshold.
- [G9] Individuals can keep their SOS history.

Track4Run

- [G10] Organizers can specify paths for running.

- [G11] Individuals can register to run as a runner with a unique code of the organization.
- [G12] Runners can track the path and their location on the map in real-time during the running.
- [G13] Organizers and spectators can see runners' instant locations on a map.

1.3. Definitions and Acronyms

1.3.1. Definitions

- TrackMe: is an application which serves to third parties to monitor the location and health status of individuals.
- Data4Help: is a service for accessing data of some specific individuals and for accessing anonymized data of groups of individuals.
- AutomatedSOS: is a service which provides monitor the health status of the subscribed individuals and controls some parameters by comparing with a certain threshold.
- Track4Run: is a service for organizers to arrange a run, for participants enrolling a run, and for spectators seeing the position of all runners during the run.
- Individual: the person who wants to benefit from the service by sharing their profile, health, location and chronic disease data.
    - Profile data: profile data is information that identifies the person. It contains name, surname, SSN, e-mail, phone number, address, gender, birth date, nationality, weight and height.
    - Health data: data measured by sensors such as pulse, the pressure in the environment and acceleration is in this group of data.
    - Location data: location data is received through the GPS sensor.
    - Chronic disease: is a disease that persists over a long period of time.
- Organizer: is a third party who arrange runs and define paths

- Smart device: the device that should be present on the individual for the necessary measurements. This device must be connected to the internet and it must be connected with smartphone via Bluetooth.

- Smartphone: the electronic device that contains our application that must be kept in connection with the smart device.

- Bluetooth (Wi-Fi): The person's smart device is connected to a smartphone via Bluetooth.

- Runner: An individual user who registered to run as an athlete.

- Ambulance: The vehicle to be sent by the hospital to the person's location when the measured values fall below a certain threshold.

- Hospital: An institution or a third party which is obliged to send an ambulance to the position of individuals when necessary.

- Active: The status of hospitals when they serve.

- Deactive: The status of hospitals when they do not serve.

- Spectator: Individuals coming to watch the run, audience.

- QR Code: Before the start of the run, code that is read into the system by spectators at the entrance, to be able to follow the runners on the map. Also, this code is used by runners in order to participate to run.

- Sensors:
    1. Pulse meter: A sensor that takes the pulse accurately.
    2. Accelerometer: A sensor in mobile phones are used to detect the orientation of the phone.
    3. Gyroscope: A sensor that adds an additional dimension to the information supplied by the accelerometer by tracking rotation or twist.
    4. Barometer: A sensor is used in meteorology to measure atmospheric pressure and forecast short-term changes in the weather.
    5. GPS: A sensor that ensures that the person's position is correctly received.

- Pre-confirmed List: A list that contains the name of the third parties that an individual allows to share his/her data without giving additional approval. The list is used for speeding up the process of data sharing approval between TrackMe and individuals. Because, every time an individual data is requested by a third party, TrackMe should ask the permission of individual for data transmission. With this list, TrackMe first checks the list of the individual and if it finds the name of the third party, then system shares individuals data directly, however, if the third party is not on the list, then TrackMe asks to individual whether he/she wants to share his/her data with this third party. Each individual has their own unique pre-confirmed lists and they are able to manage their lists to remove and add new third parties.

- Indirect Approval Request: When a third party makes an individual data request, before asking directly to the individual for his/her approval, the system first looks at the pre-confirmed list of the individual to see whether the individual has already white-listed that third party. This process is called as "indirect approval request". The reason under this process is to speed up the process of the data request and not to annoy individuals constantly each time his/her data is requested by a third party.

- Direct Approval Request: If the system makes an indirect approval request and cannot find the name of the third party under the confirmed list of individual, then it directly asks the individual whether he/she wants to share his/her data which is currently being requested by the third party. And this procedure is called "direct approval request".

- Anonymity Constraint: This is the constraint which must be satisfied when an anonymous data is requested. This constraint provides an additional layer for the proof of the anonymity of data. More specifically, it defines a threshold for the minimum number of records that should be included in the data in order to make it anonymous and available for sharing. For example, if the number of records of

the result of an anonymous data request is less than 1000, then the resultant data will not be shared with the third party.

1.3.2. Abbreviations

RASD: Requirement Analysis and Specification Document

API: Application Programming Interface

RestAPI: Representational State Transfer API

GPS: Global Positioning System

1.4. Revision History

This is the version 1.0. of RASD. The document might change during the working on design document.

1.5. Reference Document

- Specification Document: Mandatory Project Assignment AY 2018-2019.pdf

- 29148-2011 - ISO/IEC/IEEE International Standard - Systems and software engineering -- Life cycle processes --Requirements engineering

- Software Engineering, 10th edition, Ian Sommerville

- Lecture slides of Software Engineering 2

- Alloy Documentation:
    http://alloy.lcs.mit.edu/alloy/documentation.html
    http://www.diit.unict.it/~acalva/SE/slides/se1/alloy.pdf

- Example Documents:
    RASD to be analyzed.pdf

1.6. Document Structure

Chapter 1 presents a brief introduction to the project. Under this chapter, the purpose and scope of the application are discussed. Also, the definitions and acronyms that were used in the document are described.

Chapter 2 gives an overall description of the project. This chapter includes the class diagram, state charts, overall requirements, actors of the application and domain assumptions of the system.

Chapter 3 describes the specific requirements of the project. This chapter includes user interfaces, hardware interfaces, software interfaces and communication interfaces. Also, this chapter talks about possible scenarios. More to that functional requirements and performance requirements are listed in this section. Then, the design constraints are explained and the chapter is concluded with the description of software system attributes.

Chapter 4 contains the Alloy model and explanation about this.

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

Chapter 6 lists the references that were used in the project.

Chapter 2

2. Overall Description

2.1. Product Perspective

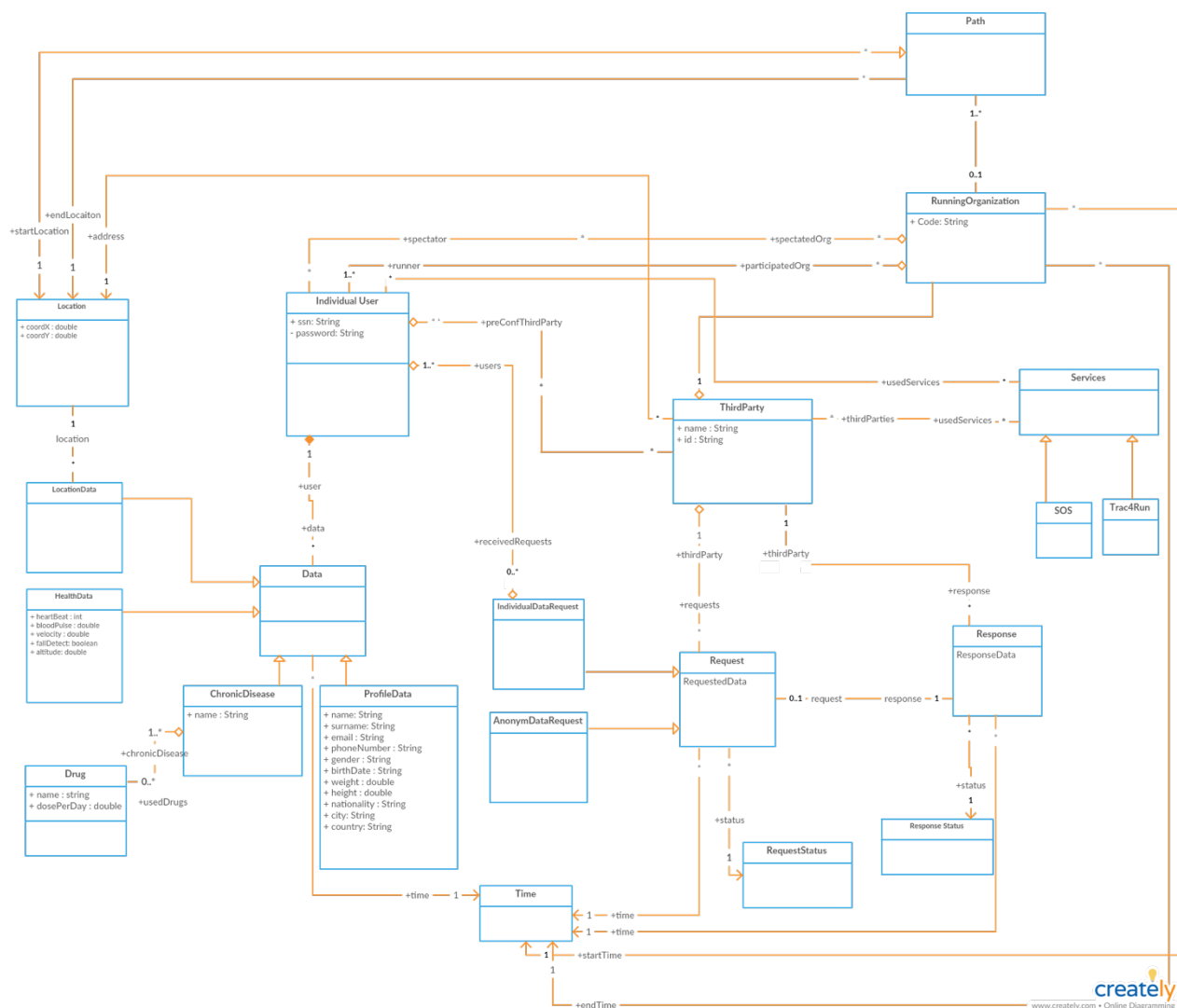#### 2.1.1. Class Diagrams

Figure 1 - Class diagram of the system

As it is seen on the Class Diagram, system is built upon individual users' data and third parties' requests. System keeps the record of every request and every rejected/approved request has a response. In order to ease the usage system

offers users to define their pre-confirmed third parties. When a request is sent by a pre-confirmed third-party, request is approved directly, and data is provided to owner of request. In order to extend functionalities of third parties system allows third parties to use both AutomatedSOS and Track4Run services. Data logic of Track4Run is mainly managed by "RunningOrganization" entity. Third parties can be organizer of the organization and individual users can enroll the organization as spectator or participants.
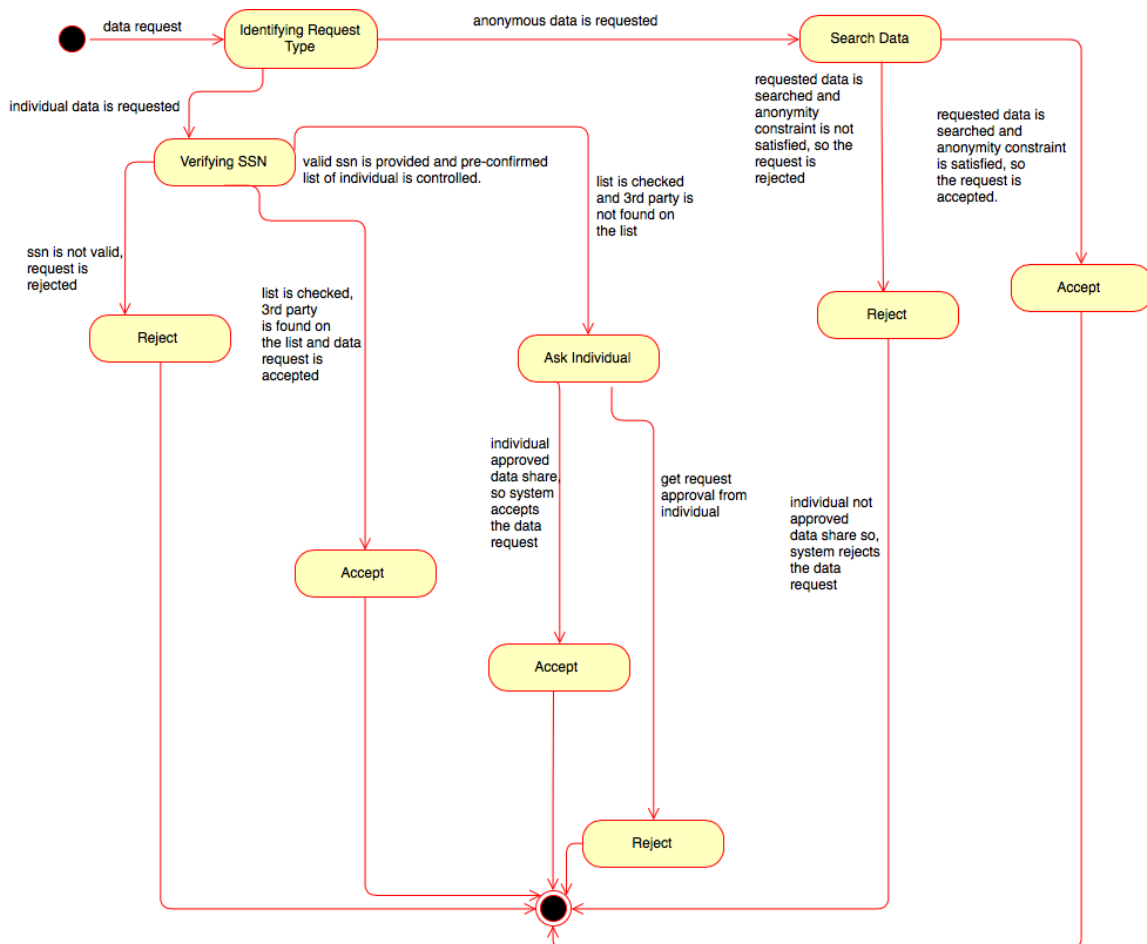
2.1.2. State Charts

Figure 2 - State diagram of "Data Request"

The above diagram is used to describe the behavior of the system when data is requested by a third party. Very briefly, there are two types of data request that

can be made by third parties; which can be named as *"Individual Data Request"* and *"Anonymous Data Request"*. According to the type of request, the system undergoes different states.
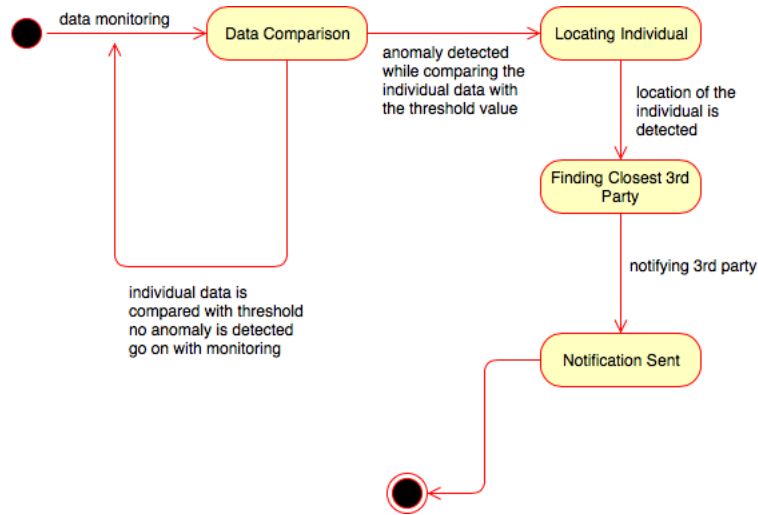
Figure 3 - State diagram of "Data Monitor"

The above diagram is used to describe the behavior of the system for the service; *AutomatedSOS.*

In this service, the system constantly monitors individual data and enters the state of "Data Comparison". On this state, it compares the health records of the individual with a predefined threshold.

If it detects an anomaly between the threshold value and individual data, it first enters a new state to locate the individual, then it enters to another consecutive state to find the third party that is closest to individual and then another consecutive state to notify this third party for the emergency situation of the individual.

If no anomaly is detected, then simply system will continue with "Data Monitoring" state and continue to monitor the data of an individual.

Figure 4 - State diagram for "Track4Run-Organizer"

The above diagram is used to describe the behavior of the system when *Data4Run* is used by organizers.

In this service, firstly the usage purpose of organizing is identified and the system decides its following state according to this.

If the organizer wants to define a path for a run, then the system will enter into an "Organizing Run" state and will request path and time of the run. Then, it will validate and save the path. After saving the path, the system will generate a unique code for the run and will pass to the "Confirm" state to finalize the process of run organization.

On the other hand, if the organizer wants to track the athletes, the system will move to "Tracking Athlete" phase and ask for the QR code. Then, it will go

on with the "verification" state and verify the given code. According to the result of verification, it will either enter to "Accept" or "Reject" state.
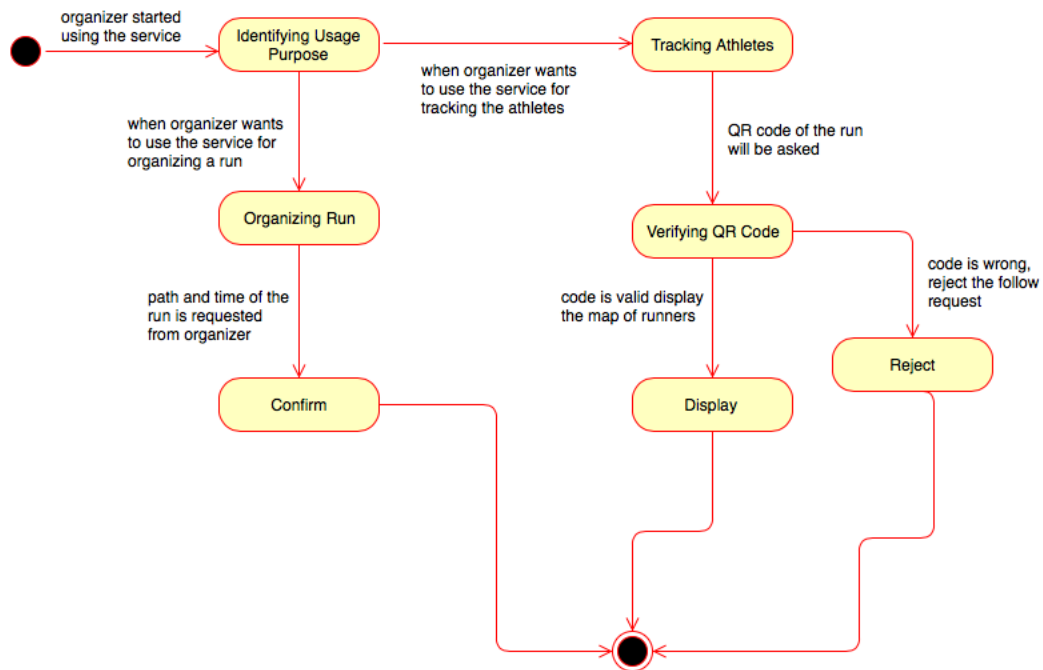

Figure 5 - State diagram for "Track4Run-Runner"

The above diagram is used to describe the behavior of the system when *Data4Run* is used by runners. Very briefly, runners will use this system to enroll for the runs.

In this service a runner is able to perform two different operations; either participate to a run or display the map of the run that he/she is being currently enrolled. And according to the choice of the runner, the system will enter two different states.
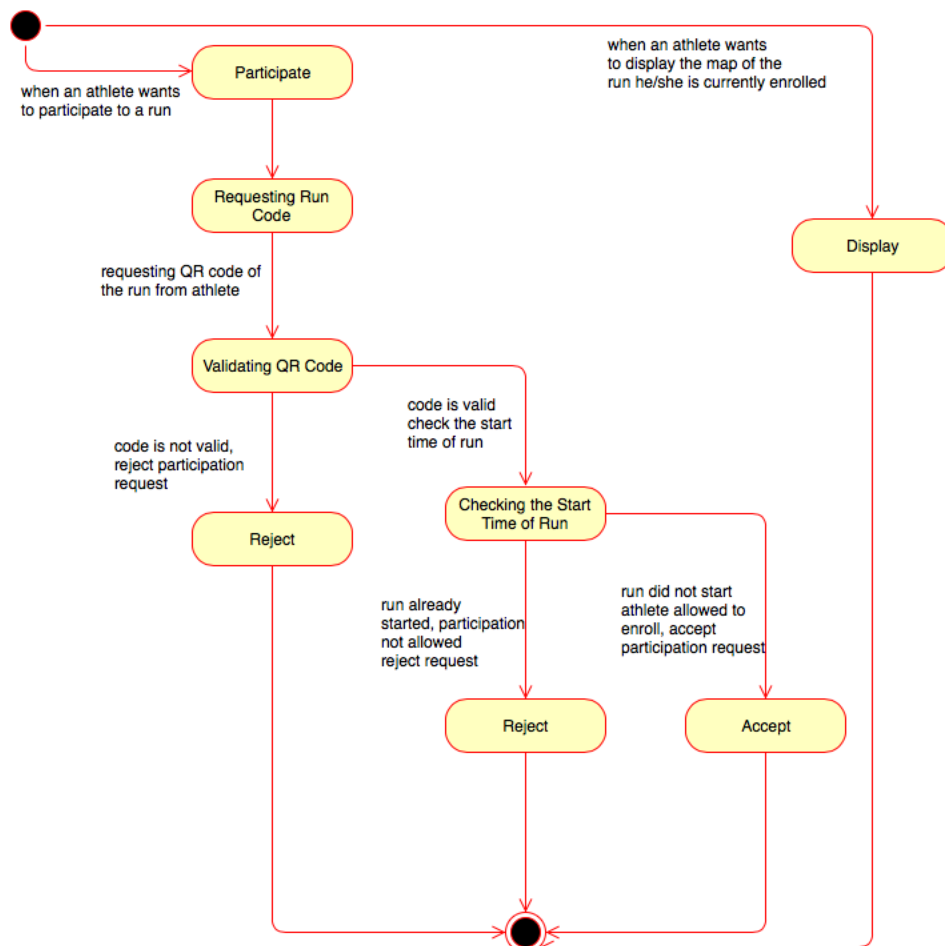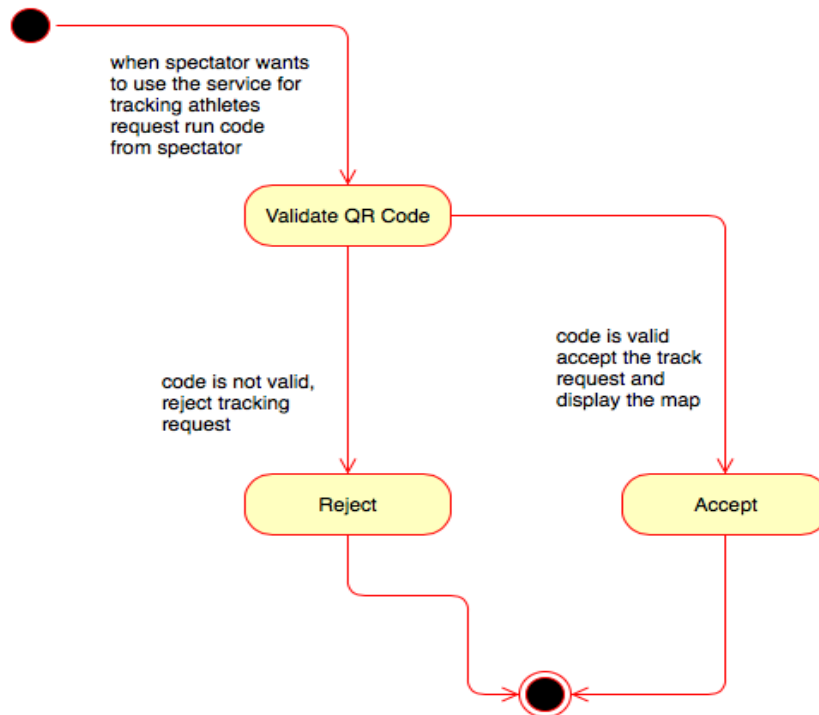
Figure 6 - State diagram for "Track4Run-Spectator"

The above diagram is used to describe the behavior of the system when *Data4Run* is used by spectators. Very briefly, spectators will use this system to track the runners on a map.

In this service, the system will first request the code of the run from the spectator, after requesting it, the system will pass to the "Validation" state in order to validate the correctness of the code. If the given code is correct, then the system will continue with the "Accept" state and display the map to the spectator. However, if the code is not correct, then it will go on with the "Reject" state and terminate the process.

2.2. Product Functions

Considering the previously mentioned goals, the overall requirements of TrackMe can be divided into several subcategories. These subcategories can be listed as;

- Data Collection

- Data Request

- Monitor Health Status

- Track Runners

Collecting Data of Individuals

Everything that TrackMe relies on is data and the flow data between third parties and individuals. So, one of the main requirement of TrackMe is to collect data; and by "data" it means collecting the health records and location of individuals.

To make the data collection possible, there are two important key points. The first one is, an individual must be registered for TrackMe and accepted a legalized statement for the collection of his/her data. During the registration phase, it is required from the individual to provide some personal information. Like the SSN/fiscal code, nationality, birth date, gender, chronicle disease etc. Then along with the health records and location, this will be used for data request and data monitoring.

The second key point of data collection is, each registered individual must hold a smartwatch or similar device; because, the process of data collection will be performed via smartwatches. Also, it is crucial to have all the smart devices connected and up to date because it is likely that there will be a continuous flow of data. For "Data4Help", a quick loss of connection might be tolerable; but, especially for the "AutomatedSOS", a successful connection is highly important.

The minimum time for collecting each data type might differ from one another. For example; to get the pulse of an individual at least "one" minute should pass, on the other hand, it is possible to get the number of steps in each second.

According to the service that is offered, the collected data can be used for multiple purposes. For example, for "Data4Help", this data will be used whenever third parties request data. For "AutomatedSOS", the collected data will be used to monitor the health status of elder individuals and to help them urgently in an emergency case. And finally, for the "Track4Run", data of the individuals (the location of the athletes) will be used to provide a mechanism for spectators to track the location of the athletes on a map.

Data Request Of Third Parties

The next important functionality of TrackMe is data request. By using TrackMe, third parties, who have been registered to TrackMe, can request data from the system.

During the data request, third parties can state whether they want to have the data of an individual or a group of people (anonymous data).

Whenever, a third party request an individual data, it will be asked to provide the SSN of an individual, then this SSN will be controlled and if it is verified, the application will turn back to individuals for the approval of the request. The approval requests will be performed in two steps each can be called as *"indirect approval request"* and *"direct approval request"*. During the *"indirect approval request"*, the application looks at the *"Pre-Confirmed List"* of the user and if it encounters with the name of the third party, then it will automatically provide the requested data to the third party without waiting confirmation from the individual. Then, the application will inform the individual about the usage of his/her data. On the other hand, if the name of the third party is not encountered, then TrackMe will make a *"direct*

approval request" to the individual, if the individual accepts the request, then TrackMe shares data with the third party, if the individual refuses the request, then the third party will be informed about the unavailability of the requested data.

For the next case, whenever a third party request an anonymous data, TrackMe, searches the requested data and determines the number of individuals whose data has been used for obtaining the data that has been requested by the third party. If the number of individuals whose data has been used is higher than 1000, then the anonymity constraint of the request will be established and the anonymous data is shared with the third party. On the other hand, if this number is less than 1000 since it will likely be possible to predict the individuals that contribute to the anonymous data request, the request is rejected.

Monitoring the Health Status of Individuals

Another important functionality of TrackMe is monitoring the health status of elder people. This functionality is implemented under the AutomatedSOS service of TrackMe and in order to benefit from this service individuals must be first registered to TrackMe and then they need to activate the AutomatedSOS service.

During the monitoring phase, as soon as data came from an individual, it is compared with a predefined threshold value. If this value is lower than the threshold, then the application will continue to collect data and monitor the individual. However, if data is higher than the threshold, then it is likely that something bad about to happen. So within seconds, the application finds the location of the individual via the GPS in his/her smart device. Then, locates the closest third party and notify it to provide an urgent help to the individual. After notifying the third party, TrackMe will save the current *"SOS firing"* data of the individual for possible future usages.

Tracking Runners

The last important point about TrackMe is its third service called as Track4Run. Different from the previously mentioned two services; Data4Help and AutomatedSOS, this service includes the *"Spectator"* s. To use this service firstly all of the actors must register to TrackMe.

A third party can use Track4Run for organizing a run. To make this, the third party must select the time and location of the run. Then, the system will save the given information and generate a unique for the run to be shared with the runners and spectators.

Runners (individuals) can enroll to the runs that are organized by third parties. Inside the Track4Run service, athletes can use the QR code of the run for enrolling to it. During the run, TrackMe will collect the health records and location of the runner via their smart device. The process of data collection in here is the same as the data collection process which was mentioned above. During the run, the real-time location of the athletes will be simultaneously displayed on a map.

And finally, spectators can register for the runs by using the provided QR code. After successfully registering the run, they can simultaneously track the real-time location of runners on a map.

2.3. User Characteristics

The following actors are users of this application:

- Individuals

A person who is registered to TrackMe and can use TrackMe's offered individual services such as data sharing. There are four services that are offered by TrackMe and can be used by individual users. These are:

- Data4Help Service: is the main service of TrackMe and every individual user can exploit. There is no extra operation to use Data4Help service. The user signed up TrackMe can use Data4Help service directly.

- AutomatedSOS Service: is the health check service which is provided only for the users who enabled it. After registering to TrackMe user must take special action, enabling, to use this service.

- Track4Run Runner Service: can be used by TrackMe users who registered a running organization as a runner. TrackMe user can register organization by entering Organization Code. User shares him/her location to third parties or spectators who registered to service.

- Track4Run Spectator Service: can be used by TrackMe users who registered a running organization as a spectator. TrackMe user can register organization by entering Organization Code and follow runners' location.

- Third parties

A party that is registered to TrackMe as a third party and can request and retrieve data of individual users' (specific users' or anonymous) data. Third parties can use three services of TrackMe. These are:

- Data4Help Service: can be used by third parties to request anonymous or specific data of TrackMe users. After enrolling to TrackMe third party can use this service directly.

- AutomatedSOS Service: is used to track individual user's health status and dispatch SOS service to user's location in case of emergencies. Third parties must register this service to use it.

- Track4Run Service: Third parties who enrolled this service can define a path of running event and follow the location of runners in the organization.

2.4. Assumptions, Dependencies and Constraints

2.4.1. Domain Assumptions

Data4Help

[D1] – Individual users have an active smart device.

[D2] – Smart devices used by individual users can monitor following features; location, heart rate, EKG, blood pressure, fall detect.

[D3] – Smart devices used by individual users provides an interface to retrieve health and location data.

[D4] – Each individual user has a valid SSN or Fiscal Code.

[D5] – Each individual user enters valid and correct information while registering.

[D6] – The health and location data provided by smart devices is accurate and real-time.

AutomatedSOS

[D7] – Third parties have a system which receives notification from AutomatedSOS and can dispatch the ambulance to requested location.

[D8] – Latency of sending SOS notification to the third party is less than 3 seconds.

[D9] – Network reliable and available.

[D10] – There are threshold values which are defined by a health organization.

[D11] – Individual users never turn their data acquisition devices off.

Track4Run
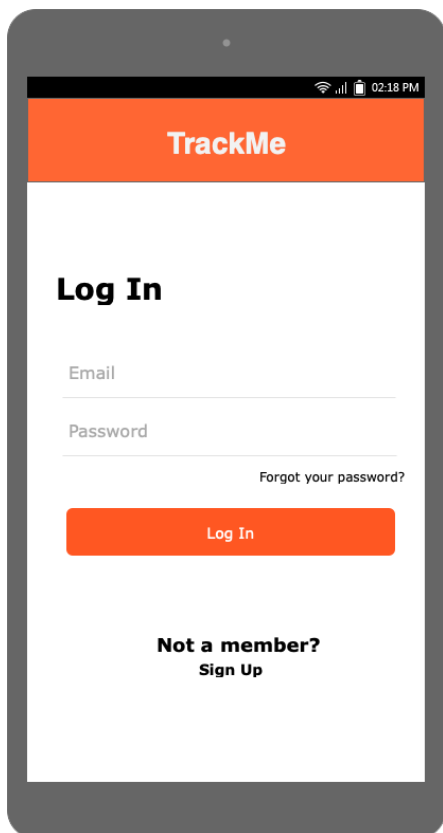
[D12] – Path defined by organizer is valid and accurate.

[D13] – Location services of runners are available and accurate during the run.

Chapter 3

3. Specific Requirements
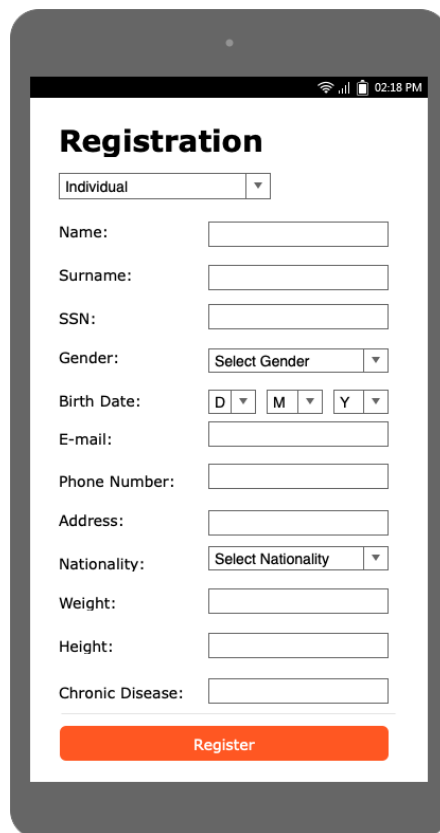
3.1. External Interface Requirements

3.1.1. User Interfaces

Figure 7 - "Login" screen

Figure 8 - "Register" screen for individual
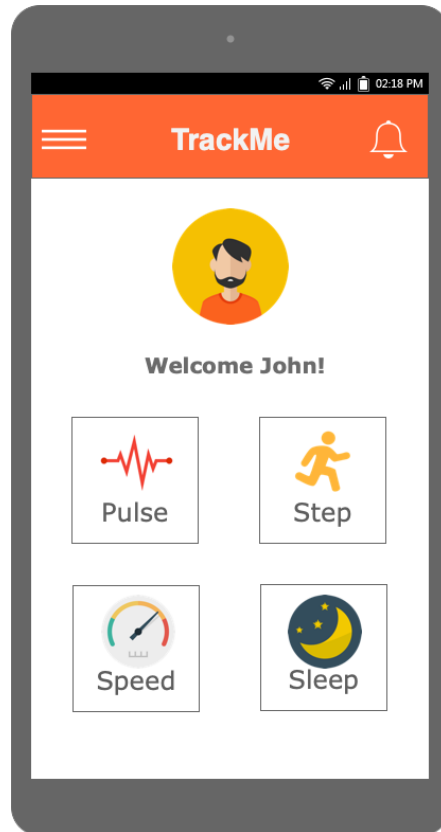
Figure 9 - "Register" screen for third party
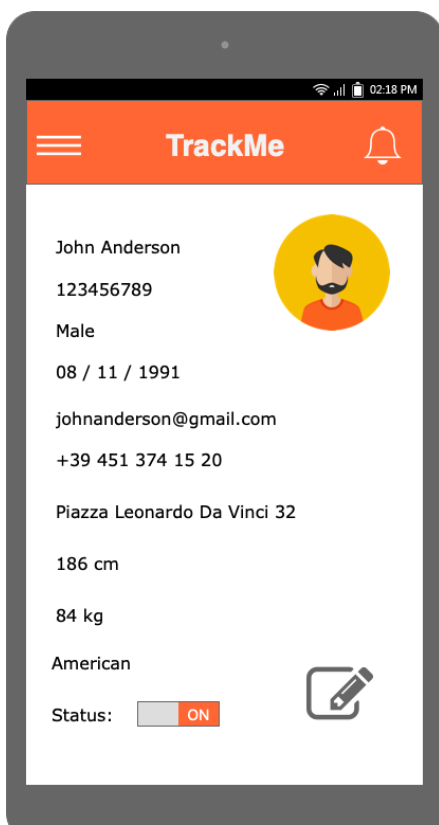
Figure 10 - "Dashboard" screen for individual
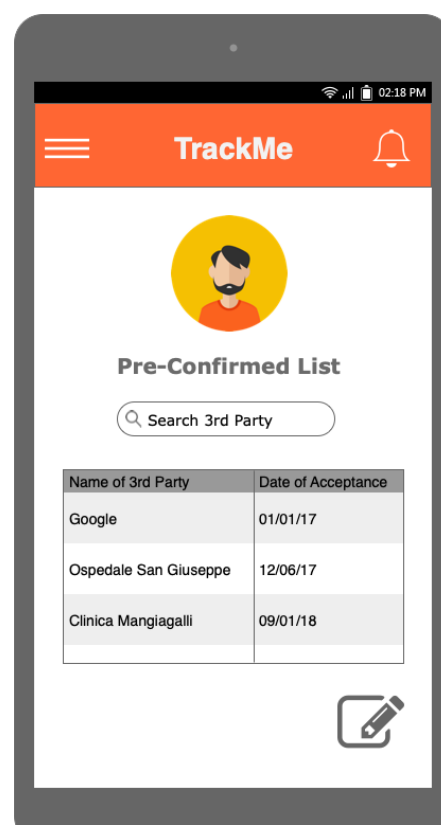
Figure 12 - "Personal Page" of individual

Figure 11 - "Pre-Confirmed List" of individual
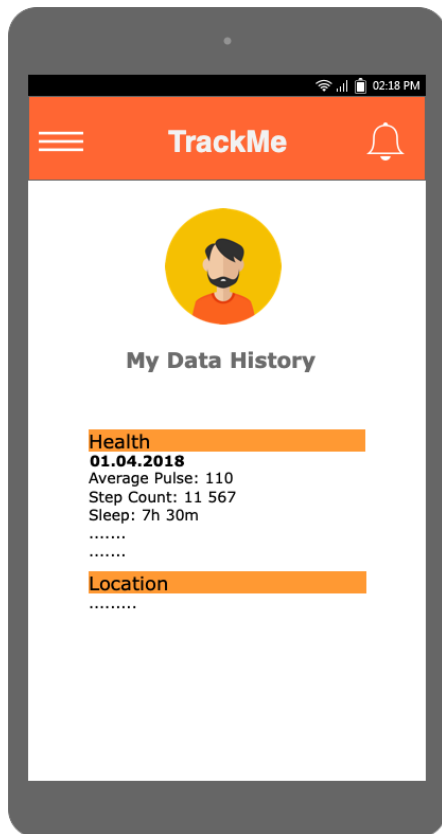
Figure 13 - "Collected Data History" of individual
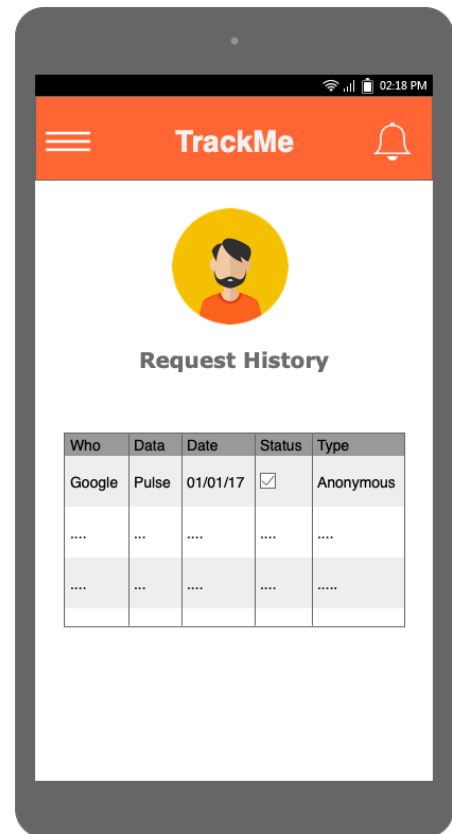
Figure 14 - "Request History" of individual
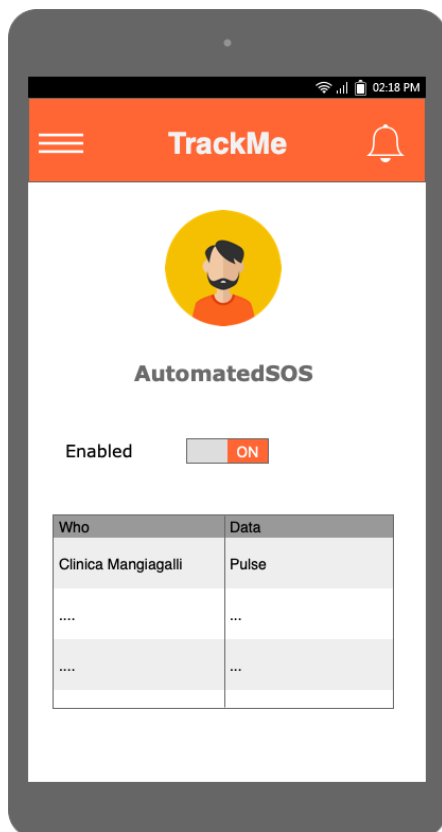
Figure 16 - "AutomatedSOS" screen
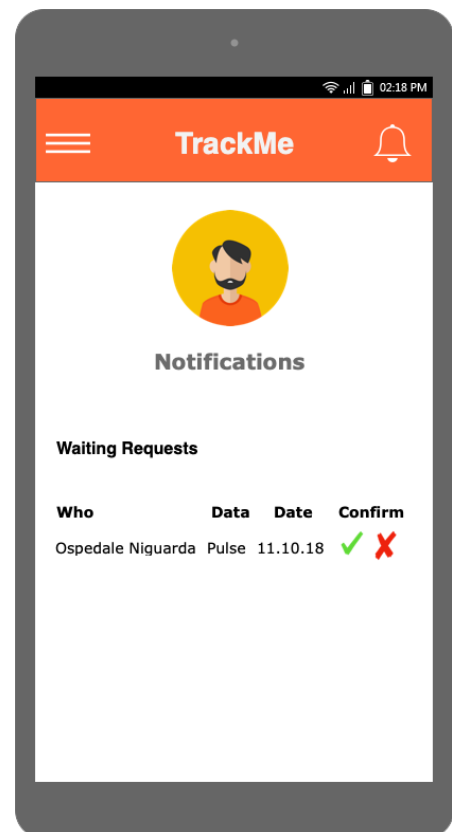
Figure 15 - "Notifications" of individual
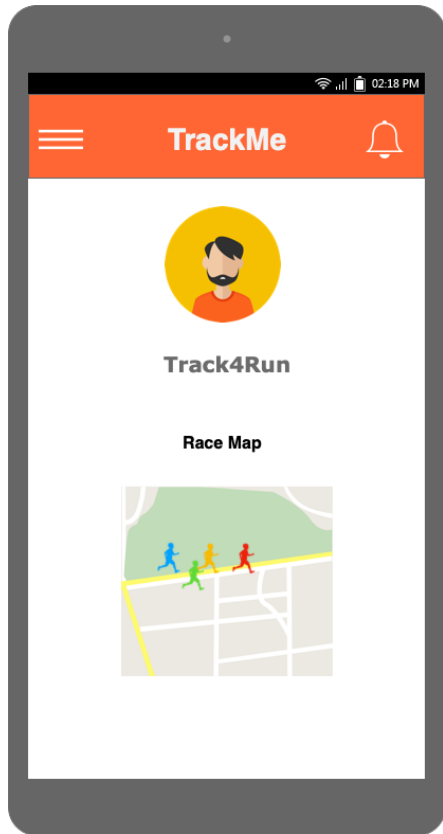
Figure 18 - "Spectate" screen of individual
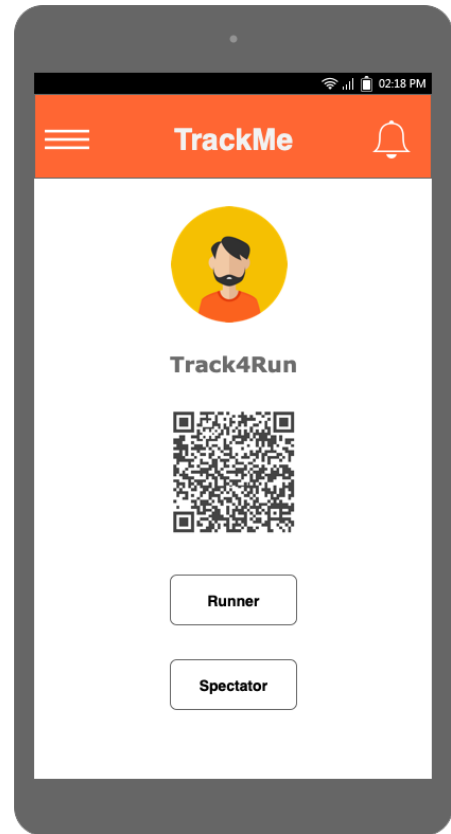
Figure 17 - "Enroll Run" screen for individual
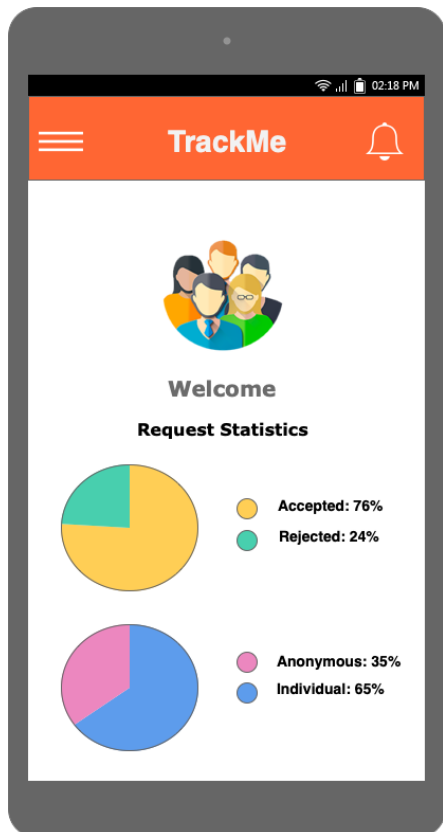
Figure 20 - "Dashboard" of third party
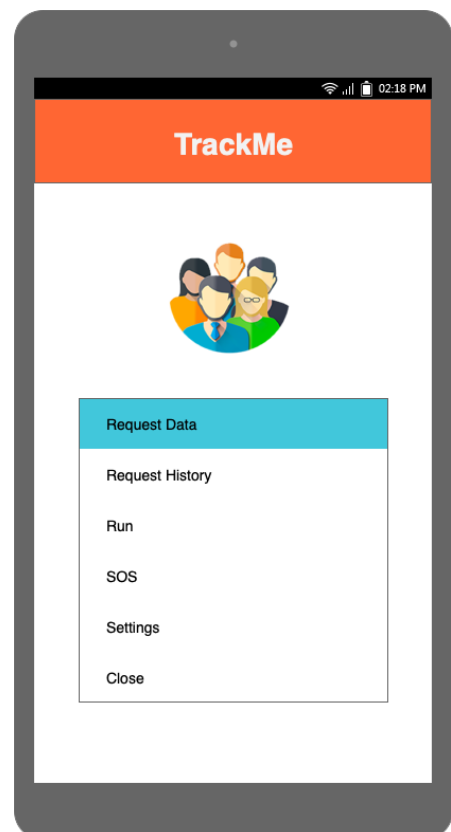
Figure 19 - "Menu Bar" of third party
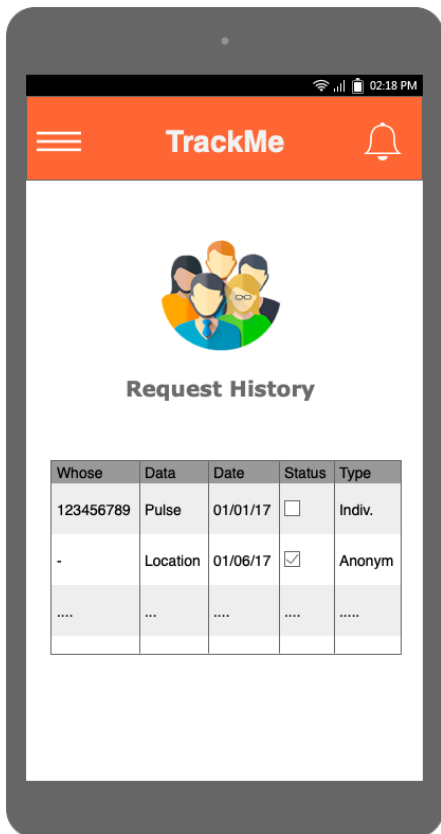
Figure 22 - "Request History" of third party
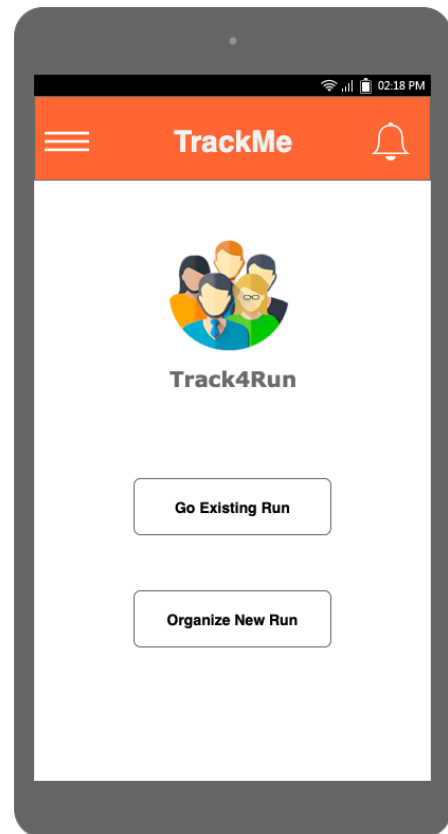
Figure 21 - "Track4Run" screen for 3ʳᵈ party
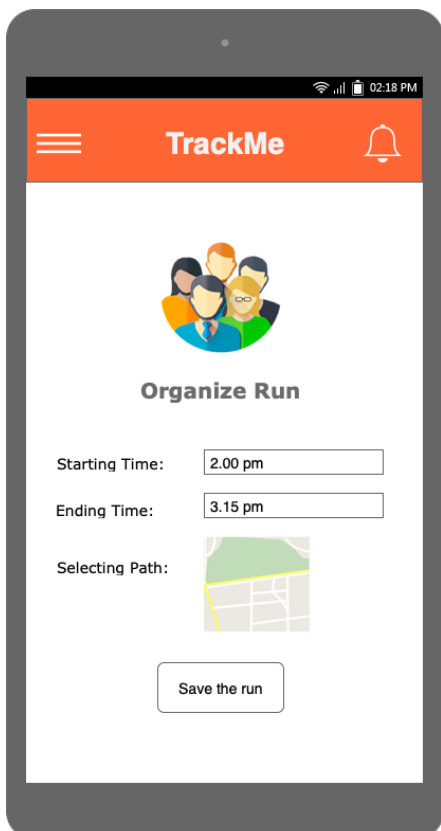
Figure 24 - 3rd party "Organizing a Run"
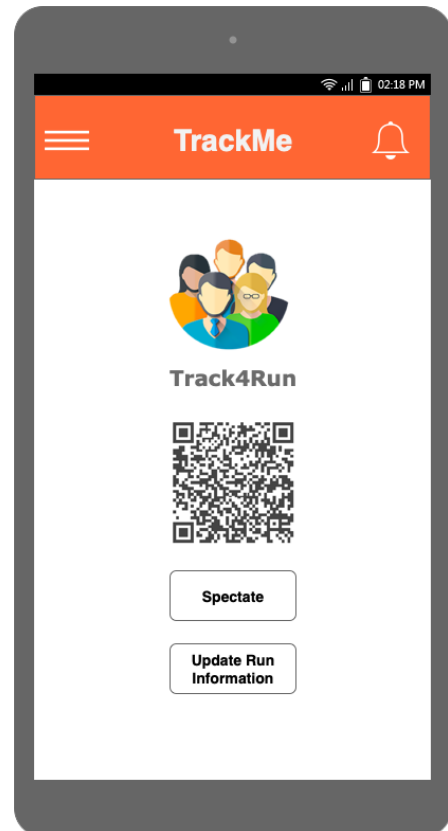
Figure 23 - 3rd party providing QR code
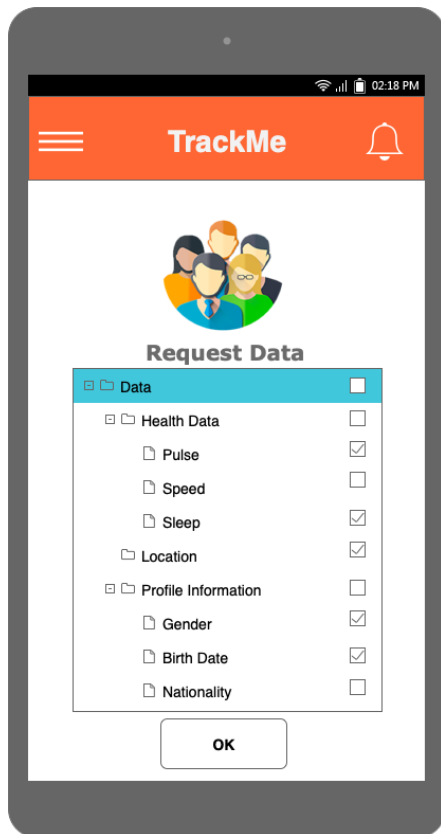
Figure 26 - "Data Request" screen of 3rd party – Part1
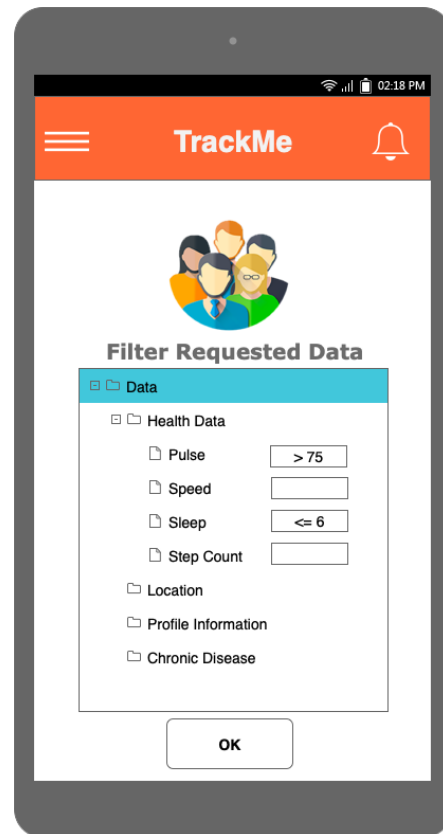
Figure 25 - "Data Request" screen of 3rd party – Part2

3.1.2. Hardware Interfaces

TrackMe is developed to collect individual user's health and location data. It is connected to smart devices by exploiting APIs provided by smart devices. It requires a smartphone which is connected to smart devices. Also, in order to connect smart devices, users should turn on their Bluetooth.

3.1.3. Software Interfaces

TrackMe uses external software interfaces to retrieve location data from users and to show this location on the map.

- **Location and Map Services**

The application needs to access to the individual location and show the location on the map. In order to access users' location, GPS services are going to be used. User's device's GPS services should be turned. In order to show location on map, there are various options such as Google Maps, Microsoft Bing Maps, Yandex Maps etc. the system is going to use one of these services. In the design process, it's going to be decided which of them is more appropriate to use.

3.1.4. Communication Interfaces

There are mainly two external communication interfaces that the system uses. One of them is exploited to retrieve data from user's smart devices and the second one is used to sending SOS notification to the third parties who are AutomatedSOS users.

To retrieve data from smart devices TrackMe uses APIs which are provided by smart devices. These APIs may be published as a web service or RestAPI. Our system should exploit interfaces which are provided in both cases.

To send SOS notification to AutomatedSOS third parties, third parties should provide us with an API to send health and location data of individual when health values exceed threshold values.

3.2. Scenarios

3.2.1. Scenario 1

"DataPower" is a new start-up which is formed by a group of Politecnico di Milano master students who aim to create robots that will be used to increase the health quality of different groups of people that are living in different regions

of Milan. In order to develop this intelligent system, first, the people of *"DataPower"* need to learn the average values of some health records in specified regions of Milan. And to get this values, they were advised to use the application; *TrackMe*. In order to get those values, people of *"DataPower"* registered "DataPower" to *TrackMe* and they performed some anonymous data requests based on their predefined regions. For each request, *TrackMe* searched its system for the requested data and checked whether the requested data satisfied the anomality constraint. Luckily, all of the requests that were made by "DataPower" satisfied the anomality constraints and *TrackMe* been able to provide all of the data that has been requested from itself.

3.2.2. Scenario 2

"D.A.T.A" requests data of an individual whose SSN is 112344583. When *TrackMe* receives the request, first it checks the given SSN and verifies it. After verifying SSN, the system looks for the name of the third party; *"D.A.T.A"* inside the pre-confirmed list of the given individual. However, it could not find the name of the third party in the list and makes a direct approval request to the individual to get his/her permission for sharing his/her data. Ms. Black who is the holder of this SSN receives the direct approval request from the system. But, she preferred not to share her data with *"D.A.T.A"* and rejected the approval request that is coming from *TrackMe.* Since individual rejected the request, TrackMe also rejects the data request of *"D.A.T.A"* and informs *"D.A.T.A"* about the unavailability of the data.

3.2.3. Scenario 3

"Us!" is an important organization in Italy. Additionally, it is one of the most reliable and oldest users of *TrackMe* and the user of "Data4Help". When the people of *"Us!"* heard about "AutomatedSOS" service, they wanted to join it. Since they are already registered to *TrackMe*, in order to use the "AutomatedSOS", they have just opened their personal page activated the

service; "AutomatedSOS". After a couple of time, the system detects an anomaly in the heart rate of Mr. Adams and to provide him an immediate help, it immediately detects the location of Mr. Adams and finds the third party that is closest to him. Then, it turns out that *"Us!"* is the closest third party to Mr. Adams, so TrackMe notifies *"Us!"* about the case and then it stores the record of this case in its own database for possible future usages.

3.2.4. Scenario 4

"Milan Health Institution" wants to organize a run for raising public awareness on the importance of doing sports. *"Milan Health Institution"* is already an active user of *TrackMe* and he prefers to use the "Track4Run" service of *TrackMe* to organize this run. For this purpose, he opens the application, follows the buttons and selects the option for organizing a new run. Then, from the already opened "form-like page", he defines the starting-ending time and selects a path for the run. After getting all that information, the system verifies the provided data. Since the date and location that is given by *"Milan Health Institution"* is found to be true, *TrackMe* generates a unique code for the third party to be shared with the possible runners and spectators of the run.

3.2.5. Scenario 5

John Anderson hears from his friends that *"Milan Health Institution"* is organizing a run for the Sunday morning. He is good at running, so he decides to participate in the run. Then, he learns that *"Milan Health Institution"* organizes this run from a service running on a popular application called as *TrackMe* and in order to enroll to run, the first thing that he needs to do is to register for *TrackMe.* He makes a quick search about *TrackMe* and registers to the application. After completing the registration, he follows the directions inside the application and clicks to the "Track4Run" option. Then, he provides the code of the run that is given to him by the organizers of the run. After providing run code, the code is verified by the system and two options are listed for John Anderson,; whether

he wants to enroll to run as a runner or whether he wants to be a spectator just for tracking the runners. Since he wants to participate as a runner, he selects the first option. After this selection, the system checks the starting time of the run with the current time in order to be sure that the run hasn't already started. After this controlling process, the system concludes that there is no problem with the timing and it successfully registers John Anderson to the run.

3.2.6. Scenario 6

A couple of students from "Polimi" see an advertisement on street about a run that will be organized by a well-known student organization. On the advertisement, they see a note saying that the run will be available for online spectating for the *TrackMe* users and they can use the given code to track the runners during the run. These students are already users of *TrackMe,* so they open the application and write down the given code. Then, they select the required option to spectate the run. After their selection, the system checks the validity of the provided code, verifies it and then, displays a map of the run which shows the position of the runners on the map.

3.3. Functional Requirements

3.3.1. Use Case Diagram
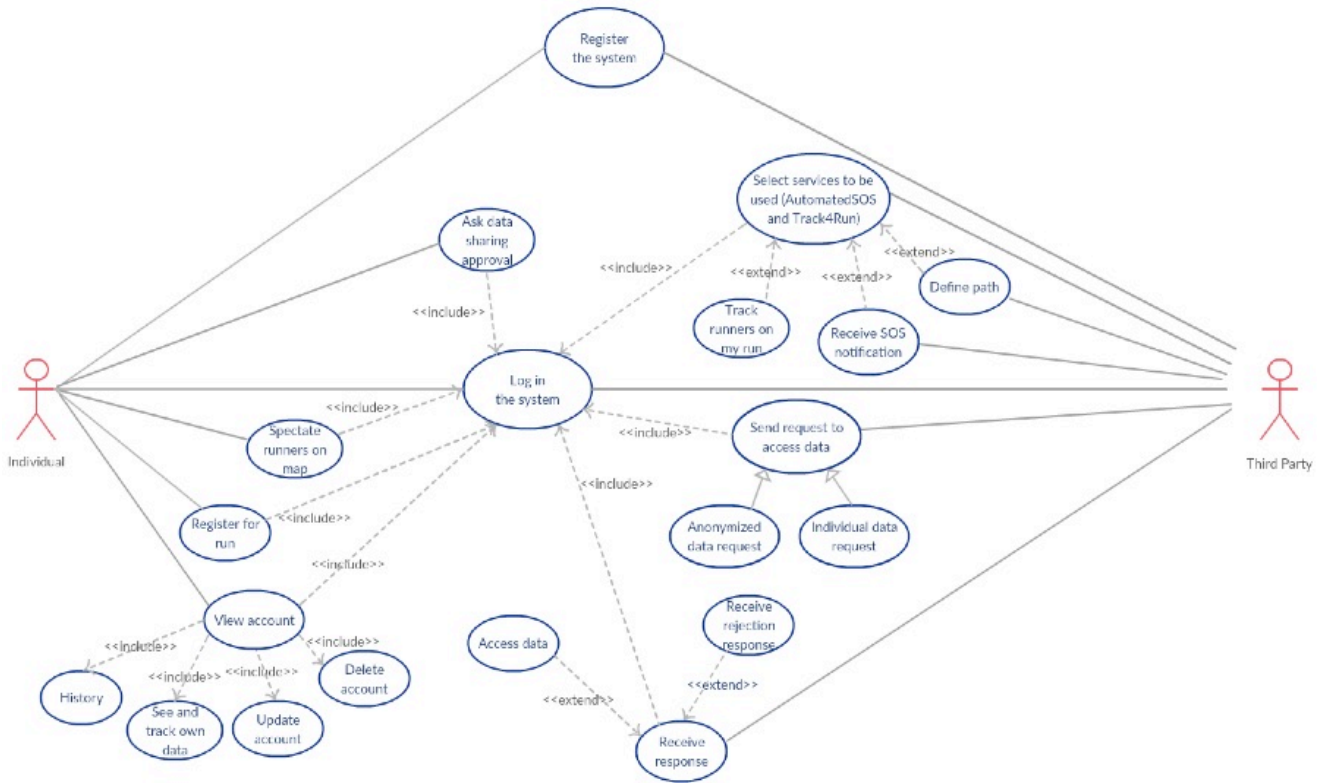
Figure 27 - Use case diagram of the system

3.3.2. Use Cases

Name	Register the system
Actor	User (Individual User, Third Party)
Entry Conditions	Individual/third party has installed the application to his/her device.
Events flow	1. Click to "Not a member? Sign up." Button. 2. Fill the mandatory fields. 3. Click to "Register" button. 4. Registration data will be saved by the system.
Exit conditions	Individual/Third party has successfully registered to the system and ready to use the application.
Exceptions	1. Individual/Third party has already registered. 2. Individual/Third party did not fill all of the mandatory fields. 3. Data in the mandatory fields are not valid.

Name	Login
Actor	User (Individual User, Third Party)
Entry Conditions	The user or third party is previously successfully signed up.
Events flow	1. The user opens the application on his/device or on opens web application 2. User enters his credentials in the "Email" and "Password" fields of the login page of "TrackMe" 3. The user clicks on the "Log in" button 4. The user is successfully logged in his/her "TrackMe" and the system automatically redirects him/her to the dashboard view
Exit conditions	The user is successfully redirected to the calendar view
Exceptions	1. The user enters invalid Username 2. The user enters invalid Password 3. All the exceptions are handled by notifying the user and taking him/her back to the login activity

Name	Update Account
Actor	Individual
Entry Conditions	Individual opens his/her personal page from the application.
Events flow	1. Individual navigates through the "Personal Information" section. 2. Individual changes the information inside the fields that he/she want to change. 3. After making the changes, Individual clicks to the button to save the changes.
Exit conditions	Individual updates his/her desired data.
Exceptions	Individual might try to make an invalid update. For example, he/she might try to insert phone number into the address field.

Name	Delete Account
Actor	Individual
Entry Conditions	Individual opens his/her personal page from the application.
Events flow	1. Individual navigates to the "Settings" section. 2. Individual clicks on the button to delete his/her account.
Exit conditions	Individual account has deleted successfully.
Exceptions	If any data of individual is being used at the time of deletion request, then individual will wait until the end of the data request to delete his/her account.

Name	See and Track Own Data
Actor	Individual
Entry Conditions	Individual opens his/her personal page from the application.
Events flow	1. Individual navigates to the "My Data" section. 2. Collected data of the individual is displayed on a grid-like structure.
Exit conditions	Individual sees his/her data that has been collected from his/her smart device.
Exceptions	If an individual attempts to see his/her data history immediately after registering to the system, since system will not have any data about this individual, it will be unable to show any result.

Name	History
Actor	Individual
Entry Conditions	Individual opens his/her personal page from the application.
Events flow	1. Individual navigates to the "Request History" section. 2. All data of the individual which has been requested by third parties are displayed in a grid-like structure.
Exit conditions	Individual sees all of his/her data which has been requested.
Exceptions	-

Name	Anonymized Data Request
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in.
Events flow	1. User opens the Request menu 2. User selects Request Data 3. User selects anonym data 4. User chooses which data s/he request 5. User defines filters 6. User submits request
Exit conditions	Request is successfully sent
Exceptions	User enters invalid data for filtering, so s/he is warned about invalid data entry.

Name	Individual Data Request
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in.
Events flow	1. User opens the Request menu 2. User selects Request Data 3. User selects Individual Data Request 4. User enters SSN of individual 5. User chooses which data s/he request 6. User defines filters 7. User submits request
Exit conditions	Request is successfully sent
Exceptions	User enters invalid data for filtering, so s/he is warned about invalid data entry.

Name	Ask Data Sharing Approval
Actor	Individual
Entry Conditions	Third party is not on the pre-confirmed list of individual.
Events flow	1. An approval request for getting the permission of the individual about the request of third party is sent to individual by the system. 2. Individual respond to the approval request of the system.
Exit conditions	If individual accept the request, system will share data with the third party. If individual rejects the request, then data will not be shared with third party.
Exceptions	-

Name	Access Data
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in. 2. Third Party's request is confirmed by individual user or TrackMe
Events flow	1. User can see the approval of its requests from notification button 2. User opens the Request menu 3. User selects Request List 4. User can see status of his/her request's response as approved 5. User click Data button to see requested data
Exit conditions	User access the data successfully
Exceptions	-

Name	Receive Rejection Response
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in. 2. Third Party's request is rejected by individual user or TrackMe
Events flow	1. User can see the rejection of its requests from notification button 2. User opens the Request menu 3. User selects Request List 4. User can see status of his/her request's response as rejected
Exit conditions	User is notified about rejection successfully
Exceptions	-

Name	Select Services to be used
Actor	Third Party
Entry Conditions	Third Party user is successfully logged in.
Events flow	1. Third Party user opens the profile menu 2. Third Party user chooses the additional services to the Data4Help (AutomatedSOS or TrackMe) 3. Third Party user saves his/her profile
Exit conditions	Services which is going to be used is saved
Exceptions	-

Name	Track runners on my run
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in. 2. Third Party user is activated Track4Run service
Events flow	1. Third Party user opens the Run menu 2. Third Party user enters the code of run 3. Position of runners showed on the map.
Exit conditions	User press exit button
Exceptions	1. The user enters invalid code and user is warned about it 2. Time of the run is the past or in the future user is warned that s/he can't track the run

Name	Receive SOS Notification
Actor	Third Party
Entry Conditions	1. Third Party user is activated AutomatedSOS service
Events flow	1. When a value of individual Automated SOS user's health data system sends notification to the closest third party which uses Automated SOS
Exit conditions	Notification is received successfully
Exceptions	-

Name	Define Path
Actor	Third Party
Entry Conditions	1. Third Party user is successfully logged in. 2. Third Party user is activated Track4Run service
Events flow	1. Third Party user opens the Run menu 2. Third Party user chooses "Create Run" option 3. A Map is shown 4. Third party selects start location and end location of run 5. Third party selects time of run 6. Third party saves the run
Exit conditions	Run is saved or user cancelled operation
Exceptions	User had already entered the same run, so system warns user about the same run.

Name	Register for Run
Actor	Individual (Runner)
Entry Conditions	Runner enters the code of the run.
Events flow	1. Runner clicks "Runner" button under the Track4Run service.
Exit conditions	Runner enrolls the run successfully.
Exceptions	The code is not valid.

Name	Spectate Runners on Map
Actor	Individual (Spectator)
Entry Conditions	Spectator enters the code of the run.
Events flow	1. Spectator clicks "Spectate" button under the Track4Run service. 2. The map that shows the location of runners on the path is displayed.
Exit conditions	Spectators track the runners from map.
Exceptions	1. The code is not valid. 2. There might be a temporary problem while displaying the map.

3.3.3. Sequence Diagrams
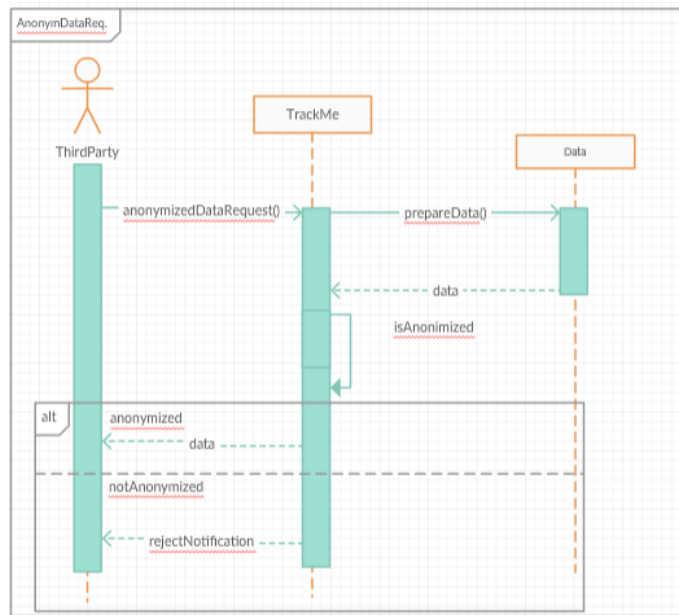
Figure 28 - Sequence diagram of "anonymous data request"
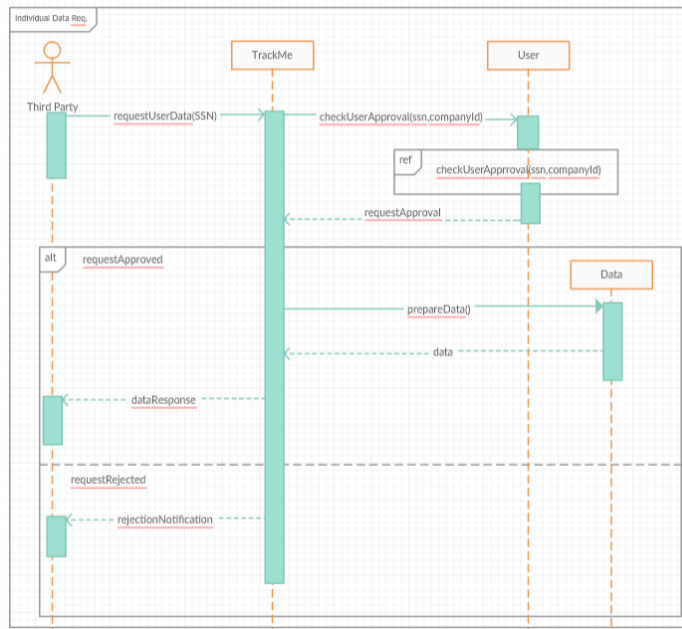
Figure 29 - Sequence diagram of "individual data request"
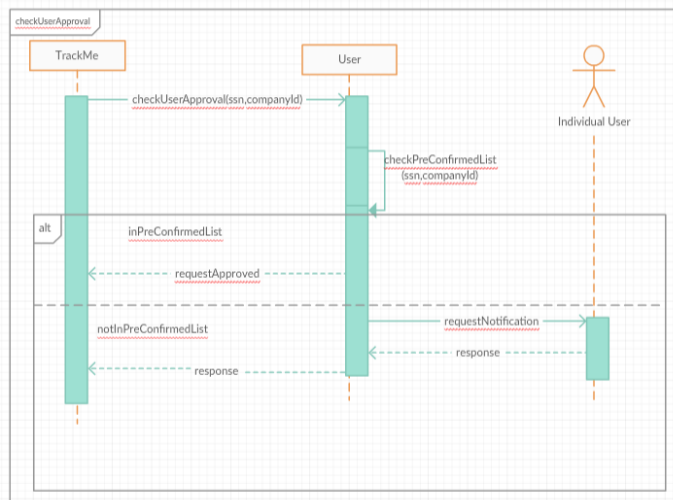
Figure 30 - Sequence diagram of "checking user approval"
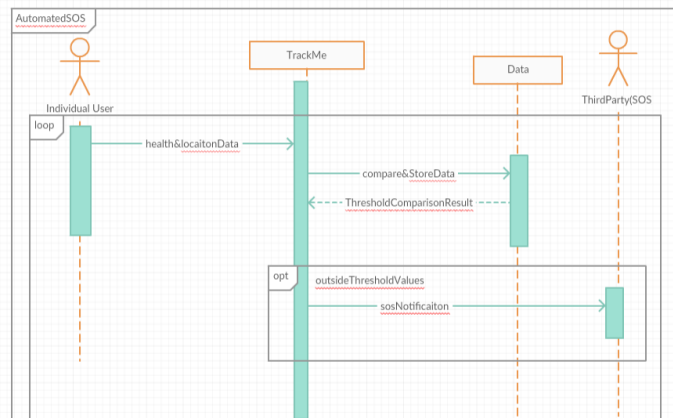
Figure 31 - Sequence diagram of "AutomatedSOS"
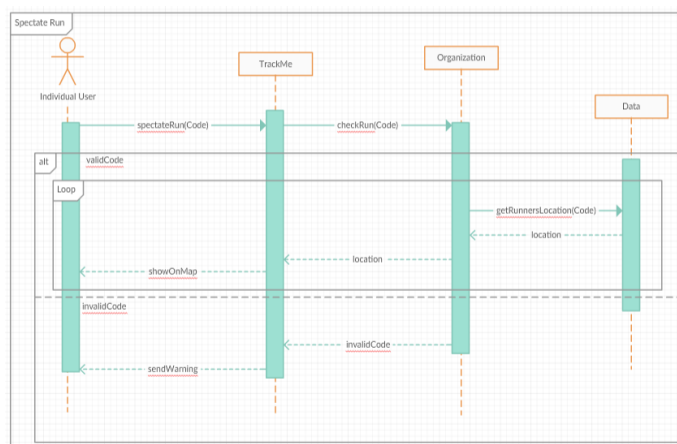
Figure 32 - Sequence diagram of "spectating run"

3.3.4. Activity Diagram
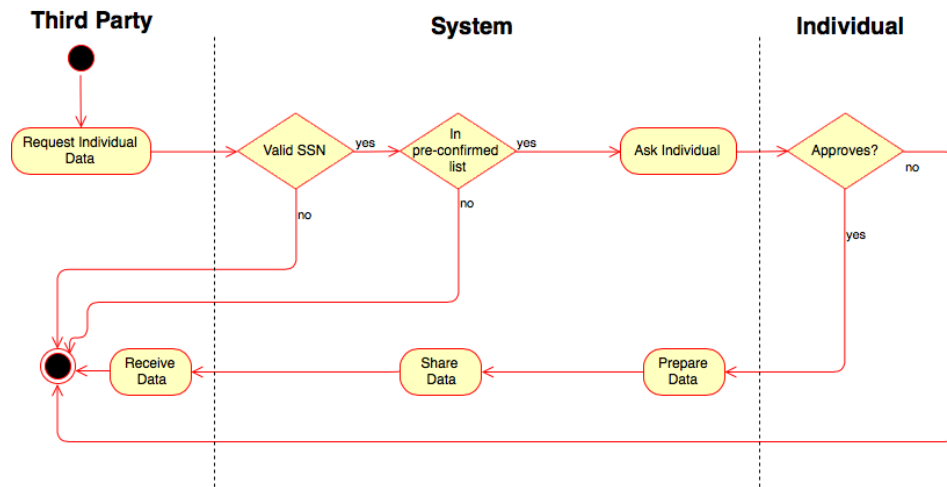
Figure 33 - Activity diagram of "individual data request"

3.3.5. Mapping on Functional Requirements

Data4Help

<u>For Individuals</u>

- [G1] Individuals can collect and store their data on Data4Help. The desired data from individuals are: profile data, health data, location data and chronic diseases.

    − The system should allow user to insert their 'chronic disease' and 'profile data'.

    − The system should ask the individual to choose which smart devices to be communicated by the application.

    − System should retrieve health and location data by exploiting the APIs that are provided by smart devices application.

    − These data must be stored in Data4Help database.

    [D1] − Individual users have an active smart device.

    [D2] − Smart devices used by individual users can monitor following features; location, heart rate, EKG, blood pressure, fall detect.

[D3] – Smart devices used by individual users provides an interface to retrieve health and location data.

[D6] – The health and location data provided by smart devices is accurate and real-time.

- [G2] Individuals can share their data with third parties which they allowed.

  – The system should allow individuals to define and update their pre-confirmed third parties.

  – If the third party who requested the data is not in the pre-confirmed list, then the system should send a confirmation notification to the individual.

  – If the individual confirms the request, then the system should share data with the third party.

  – If the individual rejects the request, then the system does not share data with the third party and send a notification to it about request rejection.

  – The system should send a request to the individual, third parties seeking access to data is not on the list that the individual is allowed to access.

  [D4] – Each individual user has a valid SSN or Fiscal Code.

- [G3] Individuals can update their personal information they have used while registering and delete their own account.

  – The system should allow individuals to view and update their profile information.

  – The system should allow individuals to delete their accounts. All information must be deleted when the account is deleted.

  – The system must accept individuals as 'active' by default when registering.

– The system should allow individuals to switch between active and deactive status.

[D5] – Each individual user enters valid and correct information while registering.

- [G4] Individuals can see and select which information is shared with which third parties when information of these individuals is shared.

– The system should allow individuals to add or remove third parties from the pre-confirmed third party list.

– The system should provide individuals can select the third parties they want to share their data with.

– The system must provide the system data flow to the individual without notification if the third party sending the request is on the pre-confirmed list approved by the individual.

– When the information of the individual is shared with third parties, the request information and the shared third party information must be kept in the database of the system to be viewed to the individual.

- [G5] Individuals can see and track their own data.

– The system must display the held data.

For third parties

- [G6] Third parties can access specific individuals' data.

– The system should check if there is an active individual in the system registered with the third party SSN.

– The system must reject requests made by a deactive individual or non-valid SSN.

– The system should check if there is a third party requesting the request in the individual's pre-confirmed list.

– The system should send the result of the request directly to the third parties if it is in the pre-confirmed list of the individual.

– The system must send requests to individuals from third parties that are not on the pre-confirmed list as approve notification.

– If the third party who requested the data is not in the pre-confirmed list, then the system should send a confirmation notification to the individual.

– If the individual confirms the request, then the system should share data with the third party.

[D4] – Each individual user has a valid SSN or Fiscal Code.

- [G7] Third parties can access anonymized data of a group of individuals.

  – The system must evaluate requests from third parties.

  – As a result of the evaluation, if the number of people whose data is desired is higher than 1000, the system should share the data with the third party. Otherwise, the request must be rejected.

AutomatedSOS

- [G8] Sends a notification to the nearest hospital in 5 seconds when health values of individual fall below a certain threshold.

  – The system should allow third parties, who are Data4 Help users to offer ambulance services, to choose their status in the system.

  – The individual registered in the Data4 Help must be able to activate or deactivate the SOS service.

  – The system should take health values of individuals and control them with thresholds.

– The system should send a notification to the nearest active third party(hospital) if the health values of the individual fall below a certain value.

[D7] – Third parties have a system which receives notification from AutomatedSOS and can dispatch the ambulance to requested location.

[D8] – Latency of sending SOS notification to the third party is less than 3 seconds.

[D9] – Network reliable and available.

[D10] – There are threshold values which are defined by a health organization.

[D11] – Individual users never turn their data acquisition devices off.

- [G9] Individuals can keep their SOS history.

– After sending the SOS notification, the system must keep track of information such as health values, location, notification time of the individual, and the third party that the information is shared in TrackMe's cloud.

– The system should allow individuals to follow their SOS history.

Track4Run

- [G10] Organizers can specify paths for running.

– The system should allow the organizers to define the date and place for the run.

[D12] – Path defined by organizer is valid and accurate.

- [G11] Individuals can register to run as a runner with a unique code of the organization.

- Each run must have a unique code.
- The system should allow the individuals to register with a unique code.

- [G12] Runners can track the path and their location on the map in real-time during the running.

    – The system should provide a map to runners for displaying his/her location and the path during the run.
    [D13] – Location services of runners are available and accurate during the run.

- [G13] Organizers and spectators can see runners' instant locations on a map.

    – The system should provide a map to spectators and organizers for displaying the path and the location of all runners during the run.
    [D13] – Location services of runners are available and accurate during run.

3.4. Performance Requirements

- The system will be implemented to serve a fairly great number of users simultaneously. At the first version, the system aims to respond around 100.000 users.
- The system should retrieve and store data which are provided by API every 30 seconds in local storage (text file).
- Taking health data of individuals leads at most 1 second; the duration comparing with the threshold must be at most 1 second.

3.5. Design Constraints

3.5.1. Standard Compliance

- Application requests only access to location and health data.
- Application run normally when installing on SD card.
- Application continue to run in the background to retrieve data when it is closed by the user.
- Data is sent to cloud to be used in possible future requests in every 4 minutes.

3.5.2. Hardware Limitations

For this system, hardware limitations are defined by smart devices. In order for the system to work properly, each smart devices must have;

- 3G/4G connection
- Bluetooth connection and
- GPS

Also, smart devices should include some additional components like the barometer, gyroscope, pulse meter, accelerometer and sensors to collect the specific type of data.

3.5.3. Any Other Constraints

- **Regulatory Policies**

After 25 May 2018, Global Data Protection Regulation (GDPR) is started to be implemented. GDPR is a regulation in EU law on data protection and privacy for all individuals within the European Union (EU) and the European Economic Area (EEA). The GDPR aims primarily to give control to individuals over their personal data and to simplify the regulatory environment for international business by unifying the regulation within the EU. Since our

system is built upon the idea of collecting individuals data, TrackMe should be aligned with GDPR and should notify users about their data privacy.

3.6. Software System Attributes

3.6.1. Reliability

The system needs to be reliable. However, the application uses cloud services to store data, so if there is a shortage in the connection between the application and the cloud, some of the operations that are performed by the system (data request) might be negatively affected which will end up with damaging the reliability of the system.

3.6.2. Availability

When the overall structure of the system is taken into consideration, it can be said that the most important operations that the system performs are; data request, data monitoring and athlete tracking. And among these; data monitoring has the highest priority in terms of availability since it is directly related to the AutomatedSOS. Since, the main purpose of this service is to provide immediate help in case of emergency, system intent to have the highest availability. On the other hand, for practical reasons, it is not very likely for the system to be available at 24/7. As a very simple example, smartwatches are used for collecting data from individuals and in case of an interruption in Bluetooth connection, the system will be unable to receive individual data and respond to any emergency case that might occur during this period.

3.6.3. Security

In order to store the passwords of users in an encrypted format, the system must use the most recent and trustworthy encryption algorithms. Also, since the

system works with sensitive data, it must be protected against the cyber-attacks that it might encounter.

3.6.4. Maintainability

The application must be written with a clear code so it can evolve to meet the possible future changes. Also, codes should be explained with reasonable comments and all of the changes that are made during the developing phase should be explicitly documented.

3.6.5. Portability

Since the application aims to reach a high number of users, it can be integrated with different platforms and it needs to be portable. Also, it should be available to download on different operating systems. Moreover, new smart devices can be easily integrated into the system.

Chapter 4

4. Formal Analysis Using Alloy

In this section, the Alloy model is given. By using it, some of the features of the system are specified and explained in more details, with the focus being on the constraints:

Data4Help Constraints

- If the status of the request is "Received" no response should be created.
- If the request is approved or rejected there should be one response for this request. Status of the response depends on the status of the request.
- When the request is approved, response data should be the same with the requested data.
- Every response should be specific to a request.
- There can't be more than one different user which has the same data record.
- Every request should be peculiar to one third party.
- When a request is sent by a pre-confirmed third party, it should be approved, and requested data should be sent the third party.

Track4Run Constraints

- A user can't attend an organization as both runner and spectator.
- In order to attend, the runner individual must activate Track4Run. Also, the third party must activate Track4Run service to organize a run.
- The organization must access the location data of it's all runners.

AutomatedSOS

- Each AutomatedSOS user should have health and location data.
- When the health values are out of range a notification should be sent to a third party which serves AutomatedSOS service.
- For every exceed of threshold values one notification must be sent.

It should be noted that in order to simplify modeling request time and respond time is excluded from modelling. Also, for monitoring the health data we only model heartbeat. Heartbeat is modeled between 0 and 20 beats per minutes. Normal values are assumed as between 10 and 15 beats per minutes.

In order to increase readability of the diagram we show three different model instances. First one focuses on Data4Help request response relation. The second one focuses on only Track4Run and the last one focuses on AutomatedSOS.

Data4Help

Model
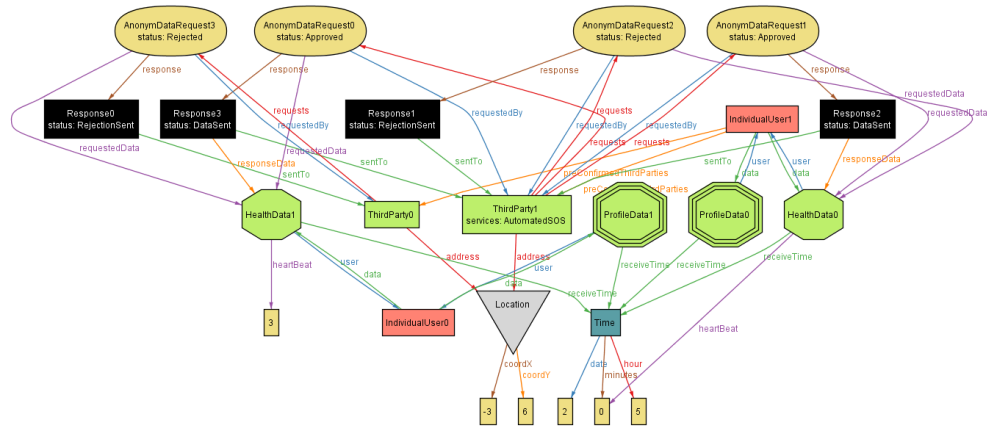
Figure 34 - Model for Data4Help

Asserts

Figure 35 - Asserts for Data4Help

Track4Run

Model
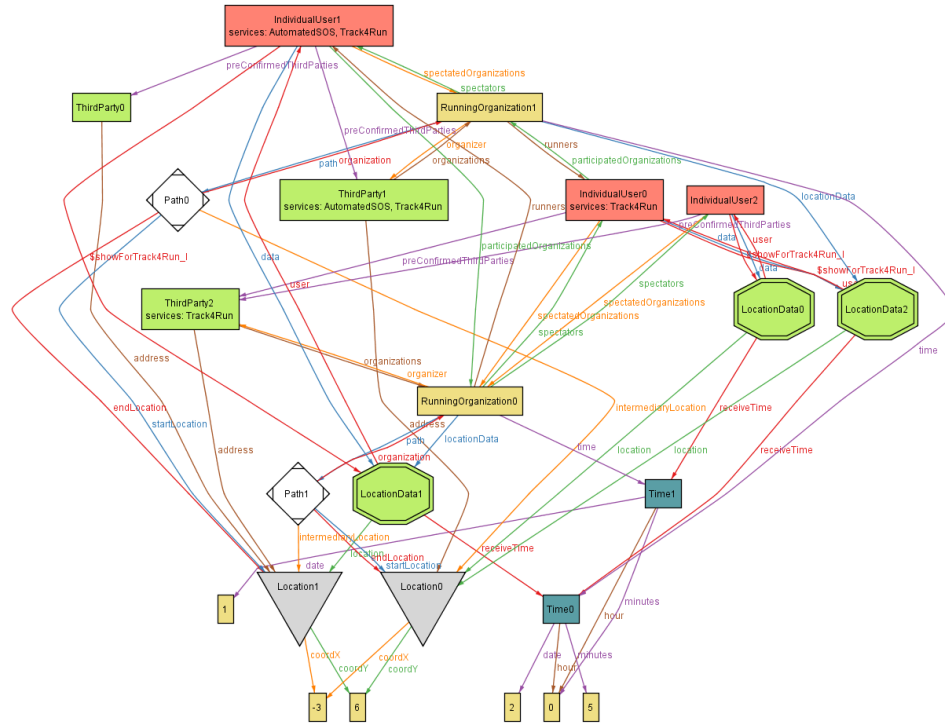
Figure 36 - Model for Track4Run

Asserts

Figure 37 - Asserts for Track4Run

AutomatedSOS

Model
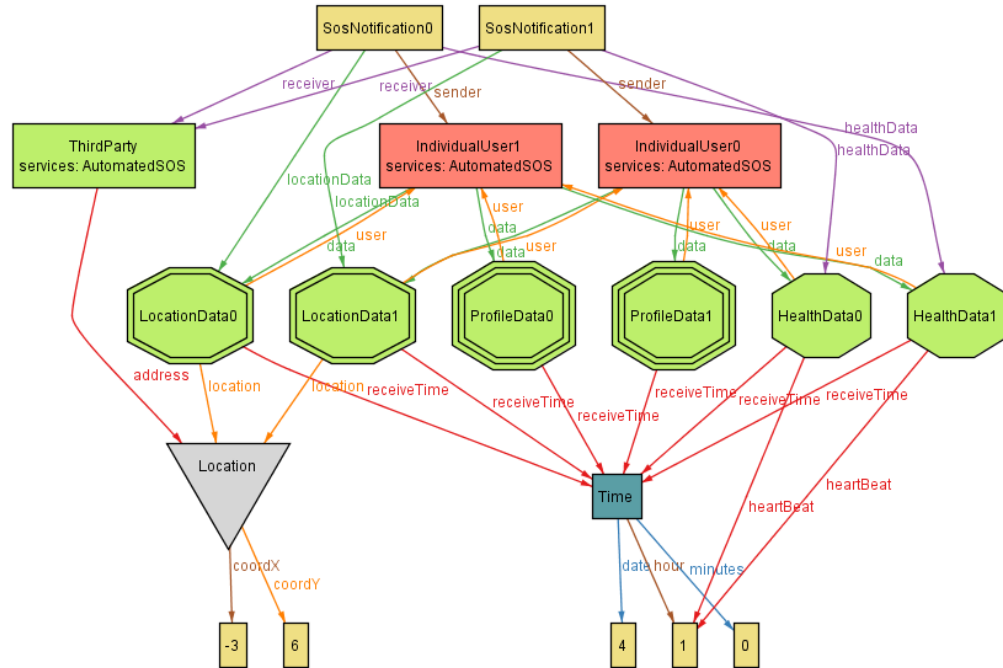
Figure 38 - Model for AutomatedSOS

Asserts

Figure 39 - Asserts for AutomatedSOS

Alloy Model

```
open util/integer

 sig Time{
        date: one Int,
        hour: one Int,
        minutes: one Int
}
{ hour >=0 and hour<=5 and
    minutes >=0 and minutes<=5 and
    date>=1 and date <=2
 }

sig Location {
            coordX:one Int,
            coordY:one Int
                }
--{ coordX >= -90 and coordX <= 90 and coordY >= -180 and coordY <=
180 }
{coordX >=-3 and coordX<= -3 and coordY >=-6 and coordY=6 }

abstract sig Data{
            user: one IndividualUser,
            receiveTime:one Time
}
{
    this in user.data
}

sig LocationData extends Data{
        location:one Location
}
sig ProfileData extends Data {
}

sig HealthData extends Data{
        heartBeat: one Int
}
{
    heartBeat>=0 and heartBeat<=20
}

sig ChronicDisease extends Data{
        drugs:set UsedDrugs
}

sig UsedDrugs{
    dosePerDay:one Int
}
 {dosePerDay >0}

sig IndividualUser{
    data:some Data,
    services: set Service,
    preConfirmedThirdParties: set ThirdParty,
    receivedRequests: set IndividualDataRequest,
    participatedOrganizations:set RunningOrganization,
```

```alloy
        spectatedOrganizations: set RunningOrganization
}

sig ThirdParty{
    requests: set Request,
    services: set Service,
    organizations: set RunningOrganization,
    address: one Location
}

abstract sig Service{}
one sig AutomatedSOS extends Service{}
one sig Track4Run extends Service{}

one sig RunningOrganization{
    organizer: one ThirdParty,
    runners: some IndividualUser,
    spectators: set IndividualUser,
    path: one Path,
    time: one Time,
    locationData:some LocationData
}
{    this in organizer.organizations
}

sig Path{
    organization:RunningOrganization,
    startLocation: one Location,
    intermediaryLocation: one Location,
    endLocation: one Location
}
{
    no l: intermediaryLocation | startLocation=l or endLocation=l
}

abstract sig Request{
    --requestTime: one Time,
    requestedData: one Data,
    status: RequestStatus,
    response: lone Response,
    requestedBy: one ThirdParty
}
{
    this in requestedBy.requests
}


sig IndividualDataRequest extends Request{
    user: one IndividualUser
}
sig AnonymDataRequest  extends Request{
}
abstract sig RequestStatus{
}
one sig Received extends RequestStatus{}
one sig Rejected extends RequestStatus{}
one sig Approved extends RequestStatus{}

sig Response{
    responseData: lone Data,
```

```alloy
    status: one ResponseStatus,
    sentTo: one ThirdParty,
    --responseTime :one Time
}
{
    one r:Request|r.response=this
}
abstract sig ResponseStatus{
}
one sig RejectionSent extends ResponseStatus{}
one sig DataSent extends ResponseStatus{}

sig SosNotification{
    sender: one IndividualUser,
    receiver:one ThirdParty,
    healthData: one HealthData,
    locationData:one LocationData
}
{
    --Time of healthData and locationData must be the same
    healthData.receiveTime=locationData.receiveTime and
    --Notification should only be sent when healt values are out of
range
    (healthData.heartBeat<10 or healthData.heartBeat>15) and
    healthData in sender.data and
    locationData in sender.data
}

--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Data4Help Facts
start>>>>>>>>>>>>>>>>>>>>

--  If status of request is "Received" no response should be created
fact requestStatusReceivedResponseRelation{
    all r:Request| r.status= Received => #(r.response)=0
}

fact requestStatusAppRejectedResponseRelation{
    all r:Request|( r.status= Approved or r.status=Rejected) =>
r.response in Response and #(r.response)=1
}

--rejected Requests' responses should be rejection sent and
responseData should be empty
fact requestRejectedFacts{
    all req: Request,res:Response| req.response=res and
req.status=Rejected => res.status=RejectionSent and
#(res.responseData)=0 and res.sentTo=req.requestedBy

}
--approved Requests' responses should be DataSent and
responseDataShould be equal to RequestedData
fact requestApprovedFacts{
    all req: Request,res:Response| req.response=res and
req.status=Approved => res.status=DataSent and res.responseData=
req.requestedData and res.sentTo=req.requestedBy
}

--every request should have it's own response
fact differentRequestDifferentResponse{
    no disj r1,r2:Request| r1.response=r2.response
```

```
}

--Every user's data should be specific
fact differentUserDifferentData{
    no disj u1,u2:IndividualUser| u1.data=u2.data
}

--There should be bidirectional relation between IndividualDataRequest
and user
fact allIndividualRequestsAreRelatedToUser{
    all r:IndividualDataRequest, u:IndividualUser| r.user=u iff r in
u.receivedRequests
}

fact bidirectionalRelationBetweenDataAndUser{
    all d:Data,u:IndividualUser| d.user=u iff d in u.data
}

-- Requested data of IndividualDataRequest and User should be related
fact DataRelationBetweenUserAndIndReq{
        all r:IndividualDataRequest, u:IndividualUser| r.user=u iff
r.requestedData.user=u
}

--Every usedDrugs should be related with at least one disease
fact checkUsedDrugs{
    all d: UsedDrugs| some c:ChronicDisease| d in c.drugs
}

--Every request should be peculiar to it's ThirdParty
fact differentRequestDifferentThirdParty{
    no disj t1,t2: ThirdParty|  #(t1.requests & t2.requests)>0
}

--all Individual request which are requested by preconfirmedThirdparty
should be approved
fact IndReqOfPreConfirmedShouldBeApproved{
    all i:IndividualDataRequest |i.requestedBy in
i.user.preConfirmedThirdParties=> i.status=Approved
}

--Each user should has only one ProfileData
fact eachUserHasOnlyOneProfileData{
    all u:IndividualUser| #(ProfileData & u.data)=1
}

--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Data4Help Facts end>>>>>>>>>>>>>>>>>>>>
--
*********************************************************************-
-
--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Track4Run Facts
start>>>>>>>>>>>>>>>>>>>>

--User can't enroll a run as a both spectator and runner
fact userCantEnrollAsSpectatorToParticipatedRun{
    all u:IndividualUser| #(u.participatedOrganizations &
u.spectatedOrganizations)=0
}
--Spectators and runners of a run is all different
fact runnerSpectatorRelation{
```

```
    all r:RunningOrganization| #(r.spectators & r.runners)=0
}
--All runners in a run must participate it
fact runnerUserRelation{
    all r:RunningOrganization, u:IndividualUser| u in r.runners iff r
in u.participatedOrganizations
}
--All spectators in a run must spectate it
fact spectatorUserRelation{
    all r:RunningOrganization, u:IndividualUser| u in r.spectators iff
r in u.spectatedOrganizations
}
-- All organizations must be in the list of it's organizer
fact organizationOrganizerRelation{
    all  r: RunningOrganization, t:ThirdParty| r in t.organizations
iff r.organizer=t
}
--Each run has it's specific path
fact differentRunDifferentPath{
    no disj o1,o2:RunningOrganization| o1.path=o2.path
}
fact organizationPathRelation{
    all  r: RunningOrganization, p:Path| r=p.organization iff r.path=p
}
--Organizer of run should activate Track4run
fact organizerTrack4RunConstraint{
    all t:ThirdParty, tr:Track4Run| #(t.organizations)>0 iff tr in
t.services
}
--Runner should activate Track4Run
fact runnerTrack4RunConstraint{
    all t:IndividualUser | #(t.participatedOrganizations)>0 iff
Track4Run in t.services
}
--Runner should have at least 1 locationData
fact runnerLocationDataConstraint{
    all t:IndividualUser| some l:LocationData|
#(t.participatedOrganizations)>0 => t.data=l
}
--#LocationData and  #Runners should be equal in Organization
fact locationNumberAndRunnerConstraint{
    all o:RunningOrganization|#(o.runners)=#(o.locationData)
}

--Organization should access to it's runners' location. So, spectators
and organizers can access it
fact organizationShouldHaveLocationDataOfRunners{
    all r:RunningOrganization, u:IndividualUser| u in r.runners
implies one l:LocationData| l.user=u and l in r.locationData
}
fact differentOrganizationDifferentData{
    no disj r1,r2: RunningOrganization|  #(r1.locationData &
r2.locationData)>0
}
--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<Track4Run Facts end>>>>>>>>>>>>>>>>>>>

--
********************************************************************-
-
```

```
--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<AutomatedSOS Facts
start>>>>>>>>>>>>>>>>>>>

--All AutomatedSOS users should have location and health data
fact sosUserShouldHaveLocationAndHealthData{
    all u:IndividualUser, s:AutomatedSOS| #(u.services&s)=1 => some
l:LocationData, h:HealthData| l.receiveTime=h.receiveTime and l.user=u
and h.user=u
}
--sender of notification must be AutomatedSosUser
fact senderMustBeSosUser{
    all n:SosNotification, u:IndividualUser| n.sender=u=>
#(u.services&AutomatedSOS)=1
}
--receiver of notification must be AutomatedSosUser
fact receiverMustBeSosUser{
    all n:SosNotification, t:ThirdParty| n.receiver=t =>
#(t.services&AutomatedSOS)=1
}

fact forEveryExceedOfThresholdOnlyOneNotificationMustBeSent{
    no disj n1,n2: SosNotification|  #(n1.healthData &
n2.healthData)>0
}


--<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<AutomatedSOS Facts
end>>>>>>>>>>>>>>>>>>>

--Each response should be peculiar to a request
assert assert_differentRequestDifferentResponse{
    no r1,r2:Request| r1.response=r2.response
}

--
assert assert_requestRejectedFacts{
    all req: Request,res:Response| req.response=res and
req.status=Rejected => res.status=RejectionSent and
#(res.responseData)=0
}
assert assert_requestApprovedFacts{
    all req: Request,res:Response| req.response=res and
req.status=Approved => res.status=DataSent and res.responseData=
req.requestedData
}
assert assert_checkRequestAndResponsesThirdParty{
    all req:Request, res:Response| req.response=res=>
req.requestedBy=res.sentTo
}
pred
requestIndividualData[u:IndividualUser,d:Data,s:RequestStatus,r:Respon
se,rb:ThirdParty,/*t:Time,*/ i:IndividualDataRequest]{
    i.user=u
    i.requestedData=d
    i.status=s
    i.response=r
    i.requestedBy=rb
--  i.requestTime=t
}
assert checkExceedingLimits{
    no n:SosNotification| (n.healthData.heartBeat=11)
```

```
}
pred showSosUsers{
    some SosNotification
}
pred isThereAnonymDataRequest{
    some AnonymDataRequest
}
--Organization asserts
assert  isThereUserEnrolledAnOrganizationAsSpectatorAndParticipator{
    no u:IndividualUser| #(u.spectatedOrganizations &
u.participatedOrganizations) >0
}
assert someOrganizationsDoesntHaveAllRunnersData{
    all o:RunningOrganization, u:IndividualUser| u in o.runners =>
(u.data & LocationData)  in o.locationData
}

pred showForData4Help{
    (some u:IndividualUser|#(u.preConfirmedThirdParties)>0) and
    (some t:ThirdParty|#(t.requests)>0) and
    (some AnonymDataRequest) and
    (some IndividualDataRequest) and
    (no RunningOrganization) and
    (no SosNotification) and
    #(Location)=2

}
pred showForTrack4Run{ one RunningOrganization}
run showForTrack4Run

run showForData4Help for 5 but 3 IndividualUser,2 ThirdParty,1
Received, 1 AnonymDataRequest, 1 Approved, 1 Rejected

check isThereUserEnrolledAnOrganizationAsSpectatorAndParticipator
check assert_checkRequestAndResponsesThirdParty
check assert_requestRejectedFacts
check assert_differentRequestDifferentResponse
check someOrganizationsDoesntHaveAllRunnersData
run isThereAnonymDataRequest
run requestIndividualData
check checkExceedingLimits
pred show{
}
run show
```

Chapter 5

5. Effort Spent

Meeting Dates;

16/10/18: 5h -> Project overview
19/10/18: 2h -> General concepts, requirements, goals
21/10/18: 6h -> RASD to be analyzed
23/10/18: 2h30 -> Domain assumptions
26/10/18: 4h -> Requirements revisited
28/10/18: 5h -> Product perspective is discussed.
02/11/18: 4h -> Use case and state diagrams are discussed.
06/11/18: 2h -> External interface requirements are discussed.
08/11/18: 1h -> Performance requirements
10/11/18: 5h -> Design constraint and software system attributes are specified.
11/11/18: 5h -> Alloy modelling and document finalization

In addition to meeting hours, each group member worked additional following hours.

Buse GUNAY: ~38h

Ezgi TASTI: ~36h

Talip TURKMEN: ~34h

Chapter 6

6. References

Github: Used for sharing project document and version control

Alloy 4.2.: Used for Alloy modeling

Draw.io: Used for drawing activity and state diagrams

Creately: Used for drawing class and use case diagrams

Mockplus: Used for drawing user interfaces

Microsoft Word 2016: Used for creating the document