Data4Help
Software Engineering II

Requirements Analysis and Specification Document

Eugenio Cortesi, Chiara Criscuolo

Document version: 1.1
November 11, 2018

Contents

1. Introduction

1.1 Purpose

Software Requirements Specification Document (RASD) is unambiguous and complete specification document, that has the purpose to describe the systems Data4Help and AutomatedSOS by TrackMe company.
Through the list of goals, requirements and assumptions, this document is necessary to clients to better understand the software and to developers, managers and testers to describe better what it is. This paper also shows the constrains and the limit of the software and simulates the typical use cases that will occur after the development. RASD is intended to all developers and programmers who have to implement the requirements, to system analyst who wants to integrate other systems with this one and could be used as a contractual base between the user and developer.
Data4Help is a software-based service developed by TrackMe company that sends data from subscribed users to third parties that requested it.
This is done by requesting the registration to third parties and individuals and consequently by saving this big data in a database that can manages this kind of queries.
It also allows users to access to their personal data and health track.
AutomatedSOS is an addictive service based on Data4Help that monitors the vital parameters of subscribed elderly people in order to start an emergency procedure in case of health problem. The software analyses the data of the patient, checks if they are under a certain threshold and in case of anomalies it calls an ambulance.
In the worst case the purpose is to guarantee that the reaction time is less then 5 seconds.

1.2 Scope

1.2.1 Description

In a world where people want to know precisely their health data, Data4Help provides a history of personal data for each subscribed user.
As well, third companies need to do mining and analysis of big data and the system Data4Help divides the data with geographical and age criteria to speed up any possible research.
Pharmaceutical companies and all others commercial business are interested in collect correct and complete data about a single person or a larger number of people and for this reason a group holding decided to finance the design and implementation of this system.
In order to do so, in a first release of the system, the software will be free for all the users.
Clients join Data4Help system inserting personal data and accepting the privacy policy of the TrackMe company.
It is important to guarantee the privacy for each user. When a request on data arrives to Data4Help database different approaches are possible.
To guarantee the personal privacy and avoid a misuse of data if the query interests a single individual, the user himself has the possibility to accept or deny the request.
If a third party asks for data related to a specific group of people, the service is able to select a group of data under certain external constrains.
For instance, if a third party asks for data about people over 60 living in a certain desolated area of Russia and the number of these people is four, then the company that will receive the data could identify their identity. Then, to avoid this risk TrackMe will not accept the request.
Therefore, in order to properly anonymize data, the system automatically checks that the group selected is composed by at least 1000 individuals, in such case the request is accepted, and the data sent to the company; otherwise the company is advised that the request is denied.
AutomatedSOS is a software (part of Data4Help) created for elderly people that needs constant observation.
Now and in the future the number of old people will increase exponentially and not all of them could be observed in a nursing home or

in a hospital. This category of people is more subjected to injuries and disease and often needs a prompt reaction.
The software continuously checks the vital parameters of user and in case they are under a threshold, starts an emergency procedure.

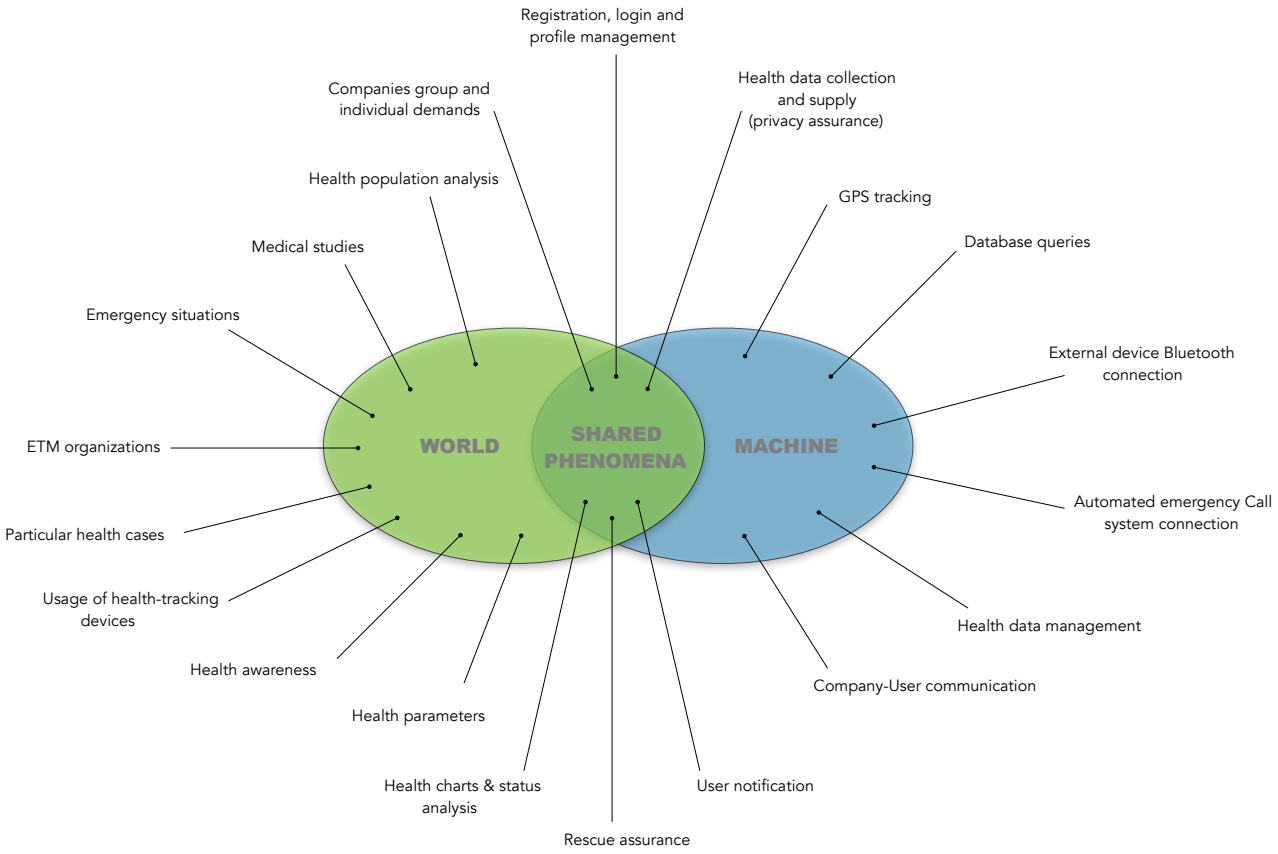

Figure 1.1: Shared phenomena between "The World" & " The Machine".

1.2.2 Goals

[G1] - Allow management and maintenance of personal and health data associated to location from a large amount of people.

[G2] - Allow a user to register, provide a unique sort of identification and login via the provided credentials.

[G3] - Allow a user to access and interpret their health track and current status associated to the location.

[G4] - Allow a user to be warned when vital parameters are not below threshold.

[G5] - Allow people to participate to studies on health, yet protecting their privacy.

[G6] -Allow third parties to take possession of data from single or a group of individuals. Also allow them to have to possibility to always receive new data when available.

[G7] - Allow an elderly person to be sure that, in case of emergency, an ambulance is called, even if he is not able to do it.

[G8] - Allow the reduction of the waiting time for an ambulance to arrive to an emergency location.

1.3 Definitions, Acronyms, Abbreviations

1.3.1 Definitions

- Database: a structured set of data held in a computer, it could be distributed if it isn't in the same physical computer.

- Emergency Procedure: is a plan of actions to be conducted in a certain order or manner, in response to an emergency event

- Query: a request (usually from a third party) in a portion or a single piece of data

- Personal data: it is any information that relates to an identified or identifiable living individual.

- Third party: an external company that is interested in the software and its data.

- User: a person who is subscribed and uses the system

- Vital signs or parameters (often shortened to just vitals) : are a group of the 4 to 6 most important signs that indicate the status of the body's vital (life-sustaining) functions

1.3.2 Acronyms

- RASD: Requirements Analysis Specifications Document

- DBMS: Database Mangement System

- GPS: Global Position System

- DB: Database

- EMT: Emergency Medical Technician

1.3.3 Abbreviations

- [Gn]: for the n-th goal.
- [Rn]: for the n-th requirement.
- [Dn]: for the n-th domain assumption.

1.4 Revision History

The document has been update: version: 1.1.

1.5 Document Structure

The document is essentially structured in four parts:

- Section 1: Introduction, it gives a general description of the document and some basic information about the software, the goals and the domain.
- Section 2: Overall Description, gives general information about the software product with more focus on constrains and assumptions.
- Section 3: Specific Requirements, this part lists requirements, typical scenarios and use cases, all of them are related to goals and constrains with the support of sequence diagrams. To give an easy way to understand all functionality of this software, this section is filled with UML diagrams and Mockups that are related to the external interface requirements and physical implementation.
- Section 4: Appendix, it contains the Alloy model and its generated world that shows the results.
- Section 5: Effort Spent, it contains the hours spent by each group member to write the document.

2. Overall Description

2.1 Product Prospective

Data4Help is an all-new application by TrackMe. It can be installed on mobile devices such as smartphones and tablets, with integrated GPS and Bluetooth. Due to their different employment, companies have access to Data4Help via the web application. In order to use the application, users will be asked to pair their health-tracking device. Data will be collected locally regardless of Internet connection, but it will be sent to TrackMe only under such condition. TrackMe stores data of all user and manage it with a DBMS, ready to provide to companies the material for their studies. Companies, registered to Data4Help, can ask through the web application for the research they are interested in.
For elderly people is also available the AutomatedSOS service. It provides them the assurance of been rescued from an ambulance in case of emergency, minimizing waiting time and excluding the risk of not be assisted if they are unable to ask for help.

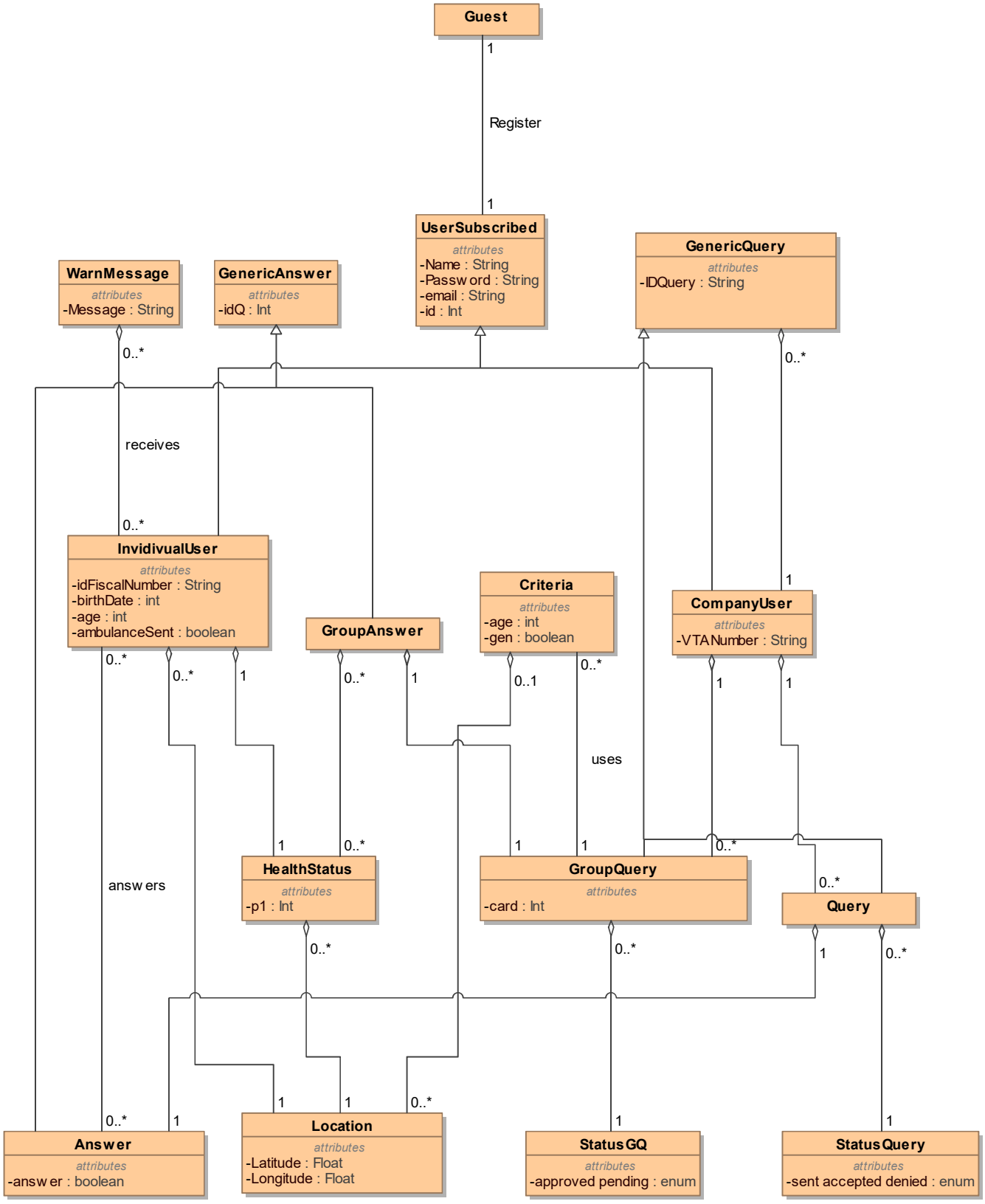

Figure 2.1: Domain Model Class Diagram.

2.2 Product Functions

2.2.1 Collecting Data

The first reason why TrackMe decided to develop Data4Help was to allow third parties to monitor the health status of specific groups of individuals for study proposes.
The base function of the program is continuously collecting heath parameters and location from as much people as possible. By storing and managing all the data received from the application of the registered users, TrackMe can create data bank that can meet the demands of the companies.

2.2.2 Forwarding Data to Companies

Data4Help provides registered companies of a web application thought which the company can insert the selection it is interested in.
As the query is sent, Data4Help starts looking in his data to see if the request can be already be answered, otherwise it waits for the data to be collected.
For privacy reasons, Data4Help is designed to release data-packages of more than a thousand individuals. All fields of the selection are maintained in the released information, only identities are classified.
The queried data is acquirable by the company directly through the web application and can be locally saved.
The function has the optional feature to release to the company new data as soon as it's available. This feature can be activated only after the company has already received a package of more than a thousand individual for that specific research.
Moreover in the case that the company is interested in the health status of a specific individual, indicated him by his social security number or fiscal code, Data4Help will send an alert to the individual with the request of communicating to the company his health status and location.

2.2.3 Health Status Awareness

The system is designed to associate each user with a virtual account from the moment of the subscription. Name, password, age, city of residence and social security number (of fiscal code) are mandatory and required during the subscription process. They can also be manages through the app in the future.
In the app the user has a personal area. The interface is designed to arrange every vital parameter that the device is able to measure. For every type of parameter, a monthly and daily charts is available. Each measurement is correlated with date and location. The user has also the possibility to save a specific chart in the "Favorites" section and share it by e-mail in form of PDF document.
Since the device for tracking parameters is required to do it continuously, as soon as a parameter is below the threshold the app will send a warn to the user and the phenomena will be highlighted in the charts.

2.2.4 AutomatedSOS

Only for 60+ users the app offers an additional service called AutomatedSOS. When a parameter is below the threshold and it has specific medical characteristic, supposing that the user is unable to do it, in 5 seconds an ambulance is called.
A service that communicates the location and data of the user to the ETM's dispatch is automatically used. The service is perfectly integrated in AutomatedSOS and it's implemented to be reliable and automated to fulfill the critical with no prior operations of the user.

2.3 User Characteristics

The possible users of the application are:

- General User: common person that decide to download Data4Help. He will have the possibility of joining the health data collection and consult its health charts.

- Elder User: person that is 60+ years old. He will as well have the possibility to join the health data collection, consult its health charts and also benefit of the AutomatedSOS service.

- Employee: person that works for TrackMe company. He can be in the team that maintains, updates and tests the system. He does not need to subscribe, because TrackMe teams have special access.

- Third Party: companies or state institutions that need health data for studies on the population or for new drugs.

2.4 Assumption, Dependencies and Constrains

[D1] - The user has installed the application on a mobile device with integrated GPS and Bluetooth.

[D2] - The user is in possession of his social security number or fiscal code; the company of the VTA registration number.

[D3] - The external emergency calls system used by TrackMe is already present on the market. It is completely developed and friendly with the ETM's dispatch.

[D4] - To send data to TrackMe and to communicate with the emergency calls service, the device is connected to the internet.

[D5] - The user application is paired with a health-tracking device, like smart-watches, fit-devices or similar.

[D6] - The health-tracking device is accurate and reliable.

[D7] - TrackMe uses a new storage system and a Database Management System.

[D8] - Only elders can benefit from the AutomatedSOS service.

[D9] - The external system doesn't fail and all emergency calls are taken in account from the EMT's dispatch.

3. Specific Requirements

3.1 External Interface Requirements

3.1.1 User Interfaces

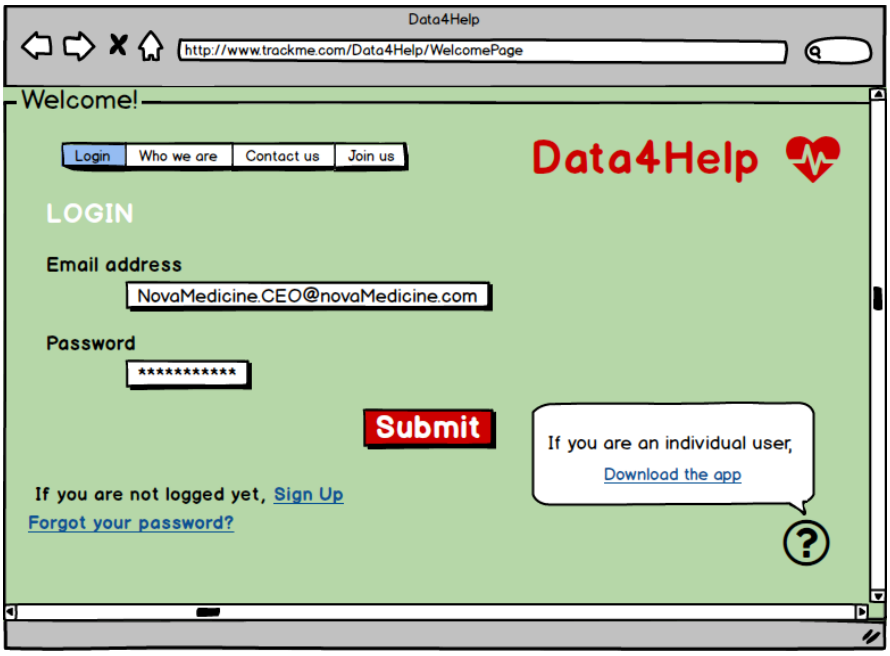

Figure 3.1: Mockups company side.

Figure 3.2: Application Mockups: Start Page and Home Page.

Figure 3.3: Application Mockups: warnings.

3.1.2 Hardware Interfaces

The app needs an external wearable device (for example a smart watch or an activity trackers). This device must be able to continuously detect vital parameters. Without it the application will enter an inactive state.

3.1.3 Software Interfaces

Data4Help doesn't need the user to operate on any external services, therefore external software interfaces are not involved in the functioning of the product.
The external emergency call service is non-intrusive, so it does not need to interface with the user.

3.1.4 Communication Interfaces

The clients communicate with the server via HTTPS protocol by using TCP and default port 443. Moreover a safe and stable connection is also established between the server and the EmergencyCallSystem. Also the communication with the storage system is connected using HTTPS protocol and DBMS protocols to ensure persistency and encryption.
In order to function the application must be paired via Bluetooth to un external device able to continuously detect vital parameters. Without it the application will enter an inactive state.

3.2 Functional Requirements

The functional requirements are grouped by the user class in which they must be satisfied. In each user class, some scenarios provide examples of usage and explain why these requirements are necessary for the functioning of the system. Furthermore use cases show step-by-step the interaction between actors and system.

3.2.1 User Class: Third Party

Scenario 1

The pharmaceutical company NovaMedicine wants to improve the characteristics of its pressure medicine BioPressure, so it contacts TrackMe to ask for people health data to analyze it.
Their studies require a large amount of people data, in the order of 10000 individuals.
The company is already subscripted to Data4Help. A NovaMedicine employee access the system thought the web application, providing the login information. He fills the selection fields and sends the demand.
Data4Help gets the queried data, checks if it's bigger than 1000 individuals (for privacy) and bigger than 10000 (for the company constraints) and eventually sends it. NovaMedicine receives the data on the web application and downloads it. Now the company employees can start analyzing it.

Scenario 2

During its studies NovaMedicine finds that there is an individual that has particular anomalies, so the company decides to request his specific parameters, indicating him by the fiscal code.
Data4Help receives the request and passes it to the specific individual, who accepts the request.
Then system automatically sends the person's data to NovaMedicine.

Scenario 3

The ASL, in Italian state institution, is facing a problem regarding a health disease spreading in the south of the country. For its studies it needs a very large amount of data, almost on all the population of the area. They study is expected to last long so the ASL activates the option to be always provided with new data, as soon as it's available.

Functional Requirements for Third Party

[R1] - *Data Selection requirement.*
In order to make possible to a company to create a personalized query, the system must provide a query form, composed by age, gender, location and cardinality. Age and gender can be not filled, instead location and cardinality are mandatory. The query is correctly filled if the cardinality is bigger than a thousand, otherwise an error message is shown and the query not sent. As soon as the query is sent, Data4Help must forward it directly to the TrackMe DBMS, that is in charge of extracting the data demanded.

[G6] [D7]

[R2] - *Company interface requirement.*
Data4Help must provide a specific web application for companies. The interface must show all the demands that the specific company made to TrackMe, and a button for the creation of a new query. The data queried must be organized by geography region and show age and gender information. The system must provide a download button for each query, so that it can be saved locally.
The interface must have a search field for a person social security number or fiscal code, of which the health status is case of interest for the company. For each query, the system must provide an option with which the company choose to be always provided with new data. This option must be available only after the company has already received a package of more than a thousand individual for that specific research.

[G6]

[R3] - *Data supply requirement.*
After the DBMS completed the selection, Data4Help must check if the cardinality available is bigger than 1000. If not so, or if the information in less than the number required, the system must put the request in a pending state and wait for other data to be collected.

[G5] [G6] [D7]

[R4] - *Data anonymization requirement.*
Before delivering to companies, Data4Help must properly anonymize data by classifying every personal information.

[G5] [D7]

[R5] - *Individual request requirement.*
When a company requests data of a specific user, Data4Help must send him the request. When he will respond and in case he accepts, his data, associated to his personal information, will be sent to the company interested.

[G5] [G6]

[R6] - *Continuous data delivery requirement.*
The system must dispose of an always open communication channel, through which all new data available of a specific query is automatically sent to the specific company.

[G6]

3.2.2 User Class: General User

Scenario 1

TrackMe released Data4Help a few months ago and, as planned, it has already a large number of subscribers. They accepted to share their health and location data with the company, so since that moment, TrackMe started to store the information of each one of them. By now the

company database is rich of data and every day it continues receiving all user parameters as soon they are collected.

Scenario 2

Amelia Fiore decides to join the Data4Help group. She downloads the application and starts to fulfill the subscription form. The registration ends with the acceptance of the Data4Help privacy rules.

Scenario 3

Cecilia Magri does frequently walks during the morning and she wants to know her cardiac rate statistics during the last month. She opens the section “Cardiac Rate Track” on Data4Help and realizes that the day before she has done the same path with a lower cardiac rate and so she understands that she can improve her training.

Scenario 4

Matteo Bedussi wants to participate to the next New York marathon, so he decides to start training. He doesn't do much activity, so during the first running session he gets from Data4Help a warning that his cardiac rate is much faster than the threshold. Thanks to the warning Matteo understands that he is doing an anaerobic activity and that, for the first training, he would better stop.

Scenario 5

Carlo Campioni, Data4Help user, this morning received a warn notification from the application, saying that NovaMedicine is interested specifically in his health status. Carlo joined the service precisely because he wanted to be part of the progress contributing with the collection of his health data, therefore he accepted the privacy request, allowing NovaMedicine to receive his data. However he is internet in knowing for what studies his data has been requested, so he is going to contact NovaMedicine with the company information provided by the app.

Functional Requirements for General User

[R7] - *Data collection requirement.*
Data4Help must always be ready to collect data. Every time new user's parameters are available the system must immediately send the information to the TrackMe storage system.

[G1] [G5] [D4] [D7]

[R8] - *Health tracking device connection requirement.*
The system must require as mandatory the paring with the health-tracking device right after the registration. The application must always be connected with it and must be able to handle the reception of new data with a high frequency. Each parameter received must be immediately stored inside the app.

[G1] [G3] [G4] [D5] [D6] [D7]

[R9] - *Identification requirement.*
The system is required to identify uniquely the users with the social security number or fiscal code.
Third parties companies will be identified with the VTA registration number.
Data4Help have to request it with a mandatory field in the subscription form.

[G2] [D2]

[R]10 *Authentication requirement.*
In order to authenticate a user/company, Data4Help must prove an authentication form at the start of the application/web page in which e-mail and password are required. Authentication must be mandatory to access into the system.

[G2] [D1]

[R11] - *User interface requirement.*
The Data4Help application must provide a specific interface for users, in which a user can access:
- his profile page;
- different sections organized by parameter type;
- in each section a chart of last period parameters, with the possibility of seeing a specific day track;
- for each chart "Share" and "Mark as Favorite" options.

The application interface must also provide a screen that opens when the user is asked to share his data with a company. The screen must contain the company name and information and the buttons to accept or refuse the request.

[G3] [D1]

[R12] - *Warns requirement.*
As soon as the heath tracking device detects a parameter above the threshold, the application must send a warn to the user.
When a Data4help user is been asked to share his location and health data with a company, the application must send a warn to the user, who is required to respond to the request in the application.

[G4] [G5] [D1]

3.2.3 User Class: Elderly User

Scenario 1

Francesco, grandfather of three, has AutomatedSOS service in Data4Help that monitors his health status. He is allergic to peanuts and accidentally has eaten them during dinner. As the anaphylactic shock occurred, AutomatedSOS got his cardiac anomaly. Since he has passed out and he is home alone no one could call the ambulance for him. AutomatedSOS checks that the cardiac rate is not below the threshold, so it starts the emergency procedure. The service calls the ambulance and sends his parameters and location. An ambulance arrives at Francesco's house and EMTs can rescue him.

Functional Requirements for Elderly User

[R13] - *Different emergency cases analysis and emergency code generation requirement.*
When an anomaly is detected AutomatedSOS starts the emergency procedure.
In order to prevent useless calls, the system is required to understand whether to wait for analyzing also the upcoming parameters or immediately assign the emergency code. It will be used to communicate the severity of the condition to the EMT's dispatch.

[G7] [G8] [D1][D5] [D6] [D8]

[R14] - *Automated and within 5 seconds management of the external system requirement.*
AutomatedSOS must always be connected with an external emergency calls service. When the emergency code is assigned, AutomatedSOS must provide to the external system all user's personal data, current location and the emergency identification code.
This must be done without requesting any interaction from the user, and within 5 second from the emergency code assignation. A message with the information about the arriving of the ambulance is shown.

[G7] [G8] [D1] [D3] [D4] [D5] [D9]

3.3 Use Cases & Sequence Diagrams

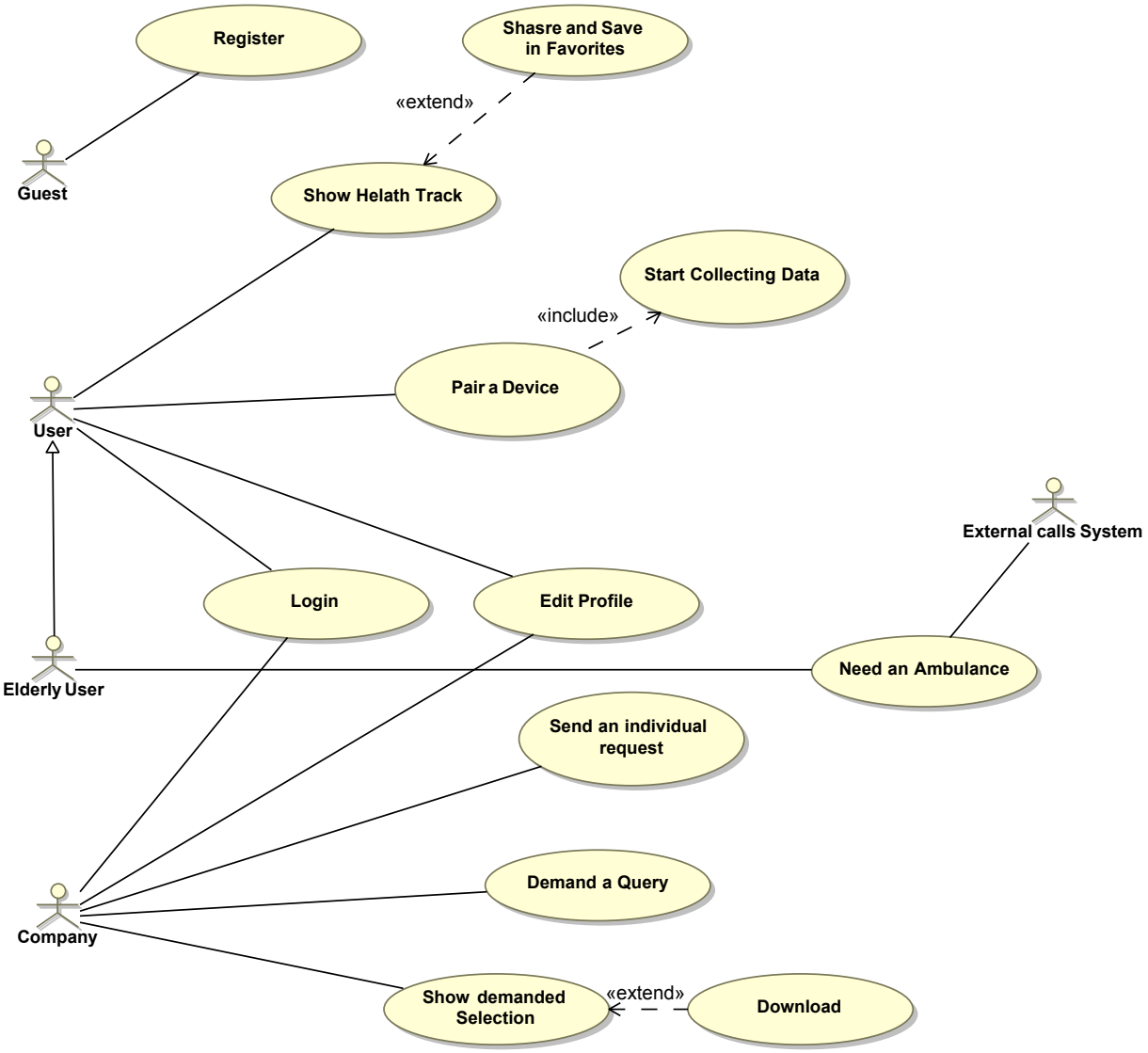

Figure 3.4: Use Case Diagram.

Name	Registration
Actors	Guest
Goals	G2, G3, G5
Input Condition	A person is on the app start page.
Event Flow	1. The guest opens the Data4Help app and clicks on the "Sing Up" button. 2. The registration form is showed and the guest inset his personal information, comprehensive of social security number/ fiscal code. 3. The guest agrees to the privacy condition, then clicks the "Confirm" button. 4. The new user's data is saved in TrackMe database.
Output Conditions	The new user is logged and lead inside the app, where he is asked to pair a health-tracking device.
Exceptions	If the guest inserts an already existing e-mail or social security number/ fiscal code, an error is shown and the system waits for another attempt.

Table 3.1 : User Registration Use Case.

Name	Registration
Actors	Guest
Goals	G2, G3, G5
Input Condition	A company employee is on the home page of the system.
Event Flow	1. The guest opens the Data4Help web page and clicks on the "Business Registration" button. 2. The registration form is showed and the guest inset the company information comprehensive of VTA number. 3. The guest agrees to the privacy condition, then clicks the "Confirm" button. 4. The new company's data is saved in TrackMe database.
Output Conditions	The company is logged to the system and the employee redirected to the home page.
Exceptions	If the guest inserts an already existing e-mail or VTA number, an error is shown and the system waits for another attempt.

Table 3.2 : Company Registration Use Case.

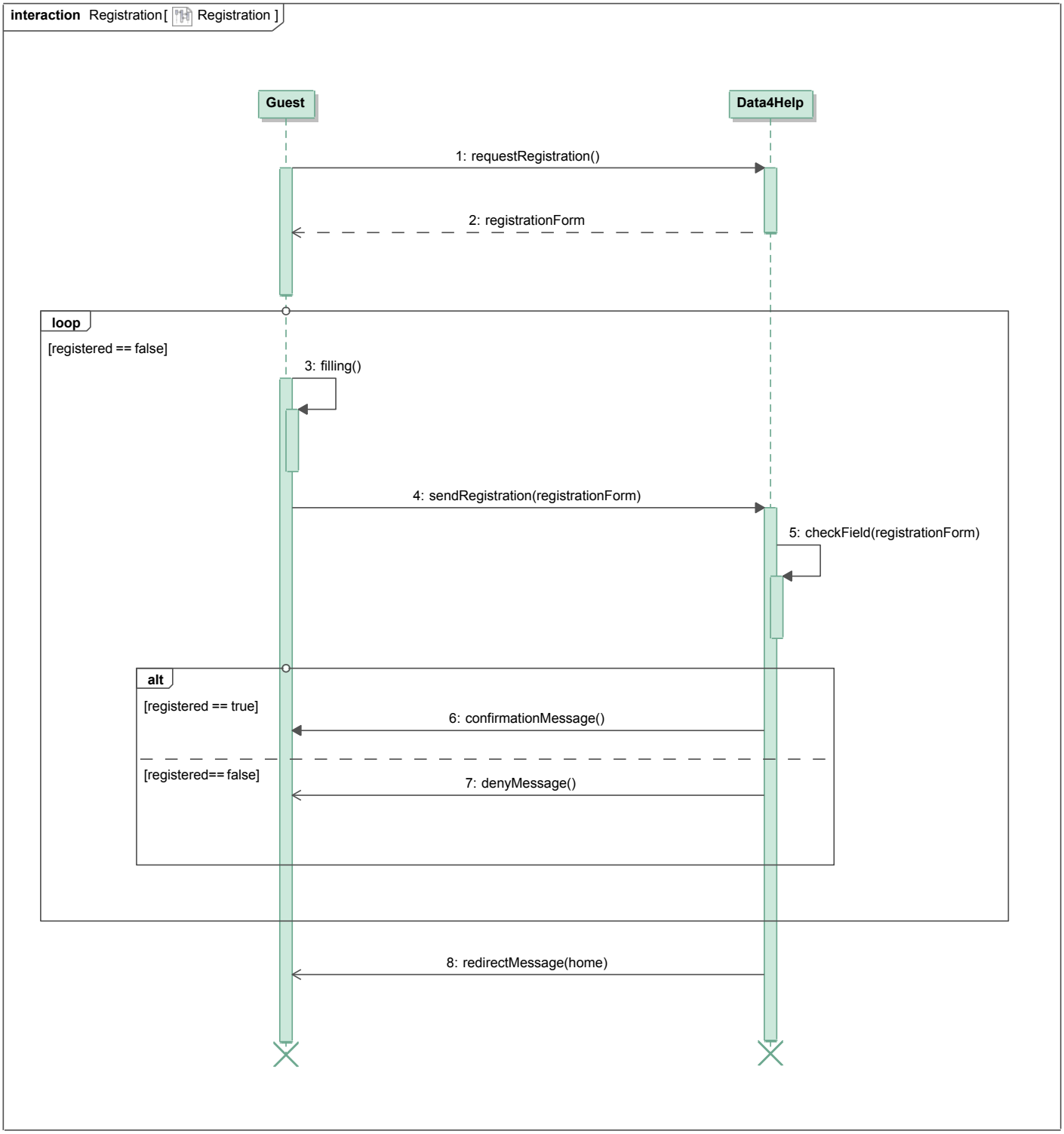

Figure 3.5: Registration Sequence Diagram.

Name	Login
Actors	Company Employee
Goals	G2
Input Condition	Employee opens the web page, his company is already been registered, but it's not authenticated.
Event Flow	1. He inserts the company e-mail and password 2. He clicks on the "Enter with your Company" button.
Output Conditions	The company is logged in the system and the employee is directed to the home page.
Exceptions	If the employee inserts an wrong e-mail or password, an error is shown and the system waits for another attempt.

Table 3.3 : User Login Use Case.

Name	Login
Actors	User
Goals	G2
Input Condition	User opens the app, he is already registered, but he is not authenticated.
Event Flow	1. The inserts his e-mail and password 2. He clicks on the "Sign In" button.
Output Conditions	The user is logged and lead inside the app.
Exceptions	If the user inserts an wrong e-mail or password, an error is shown and the system waits for another attempt.

Table 3.4 : Company Login Use Case.

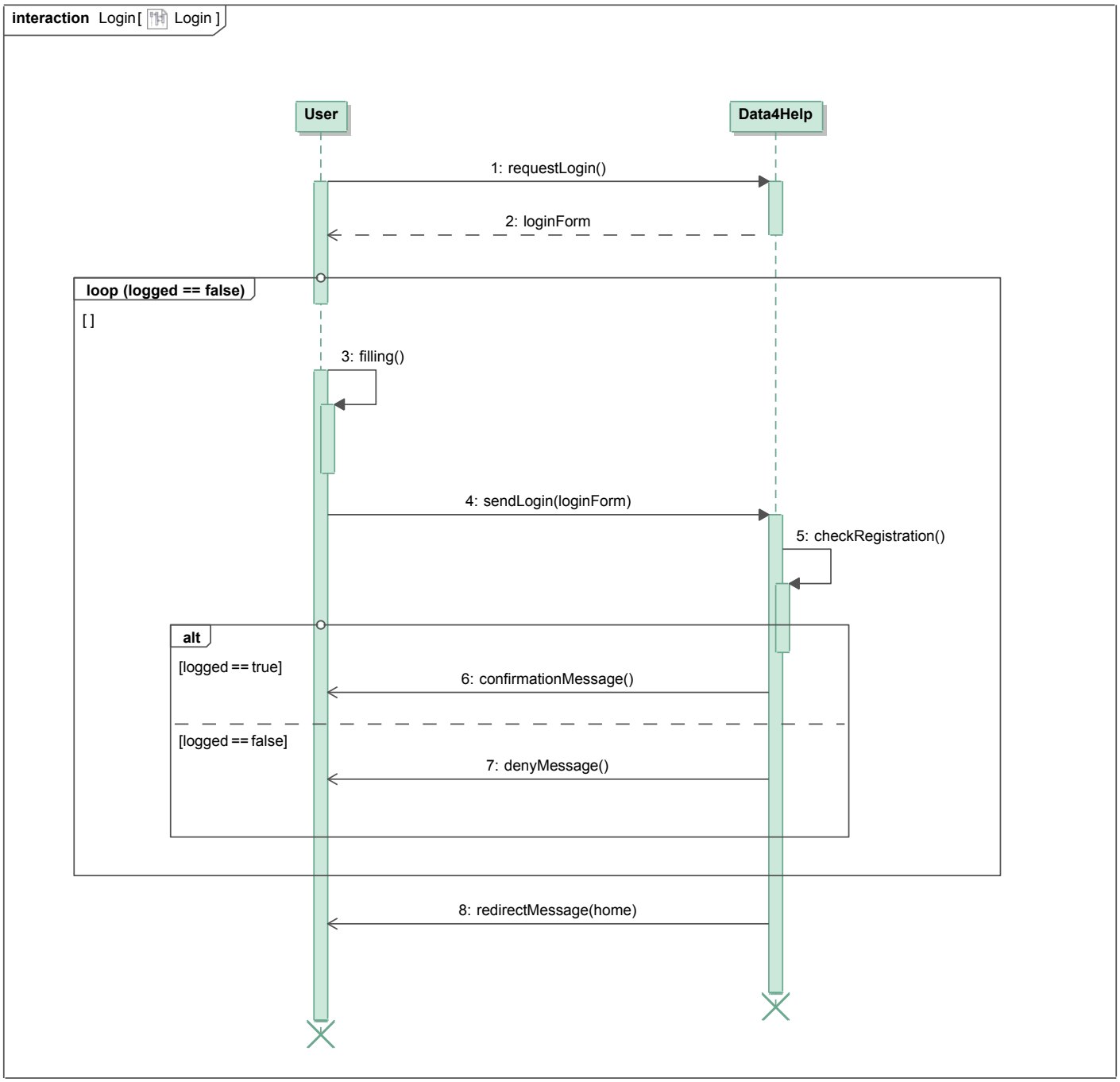

Figure 3.7: Registration and Login Activity Diagram.

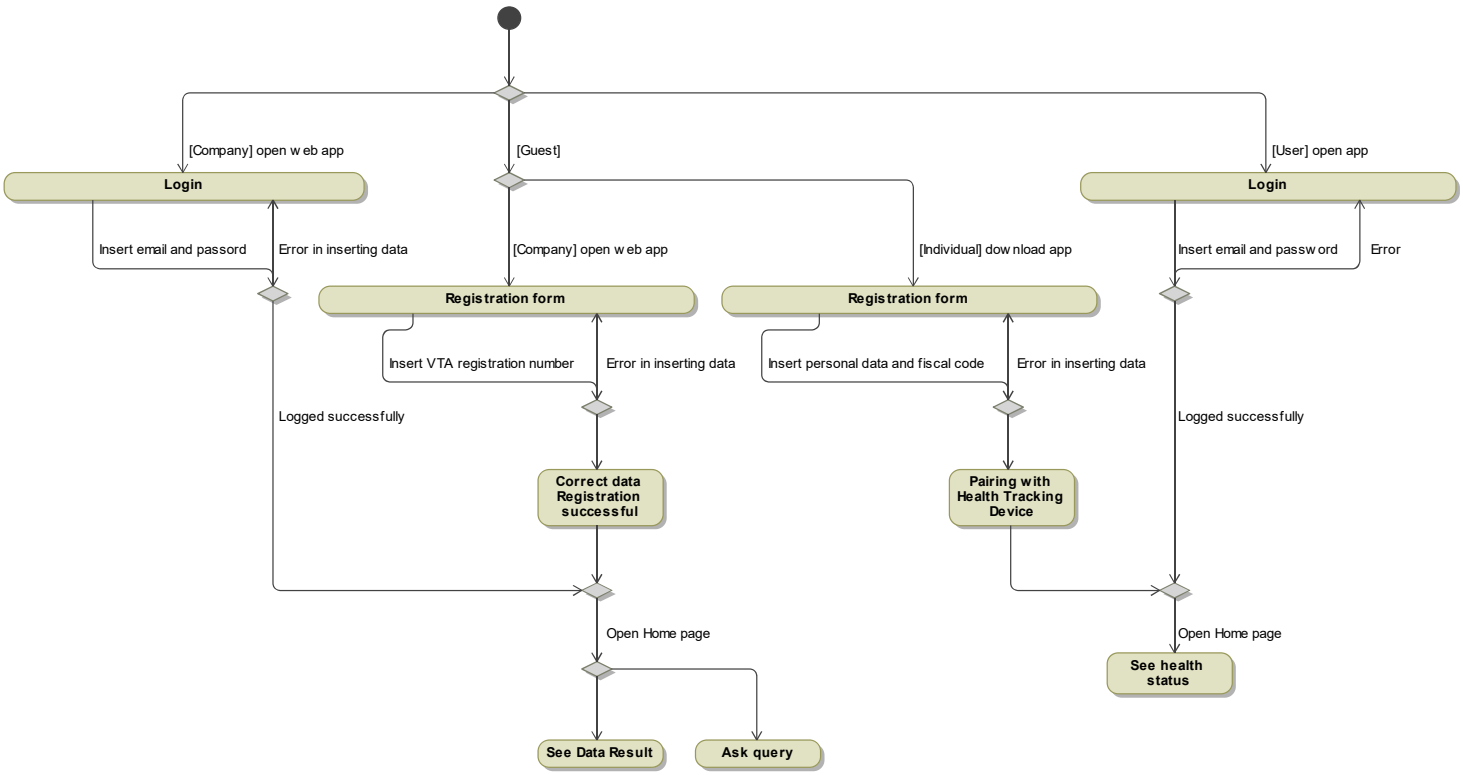

Figure 3.6: Login Sequence Diagram.

Name	Edit Profile
Actors	User, Company Employee
Goals	G2
Input Condition	The logged-in user is on the home section.
Event Flow	1. He clicks on the “Personal Information” button. 2. The page with the form filled with personal information is loaded. 3. He clicks on the “Edit” button. 4. Done the modifications, he clicks on the “Done” button.
Output Conditions	The logged-in user is on the home section.
Exceptions	If the fields are not correctly filled only the “Cancel” button is available.

Table 3.5: Edit Profile Use Case.

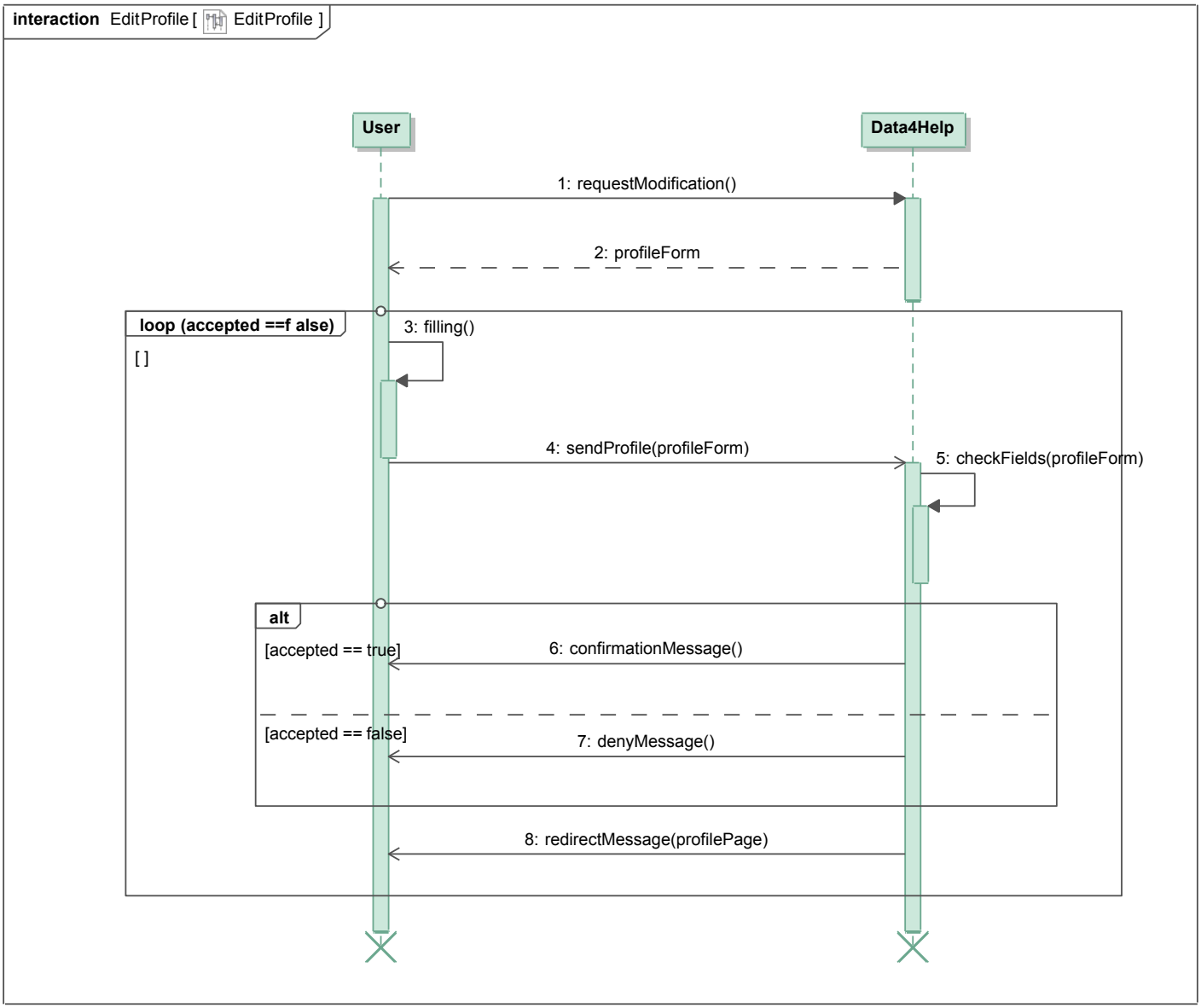

Figure 3.8: Edit Profile Sequence Diagram.

Name	Send Data Request
Actors	Company Employee
Goals	G6
Input Condition	The company is logged-in and the employee is on the home section.
Event Flow	1. The employee clicks on the "Start a new Selection" 2. He inserts the age and location ranges, sex constraint, entity of data desired 3. The system sends to track me query.
Output Conditions	The company is logged-in and the employee is on the home section.
Exception	If the quantity of data required is less than the privacy-constrain the selection can not be sent.

Table 3.6: Send Data Request Use Case.

Name	Access Response Material
Actors	Company Employee
Goals	G6
Input Condition	The company is logged-in and the employee is on the home section.
Event Flow	1. He clicks on one of the query he previously asked to the service. 2. The table with data organized according to the selection demanded is loaded.
Output Conditions	The company is still logged in.
Exceptions	There are no exceptions.

Table 3.7: Access Response Material Use Case.

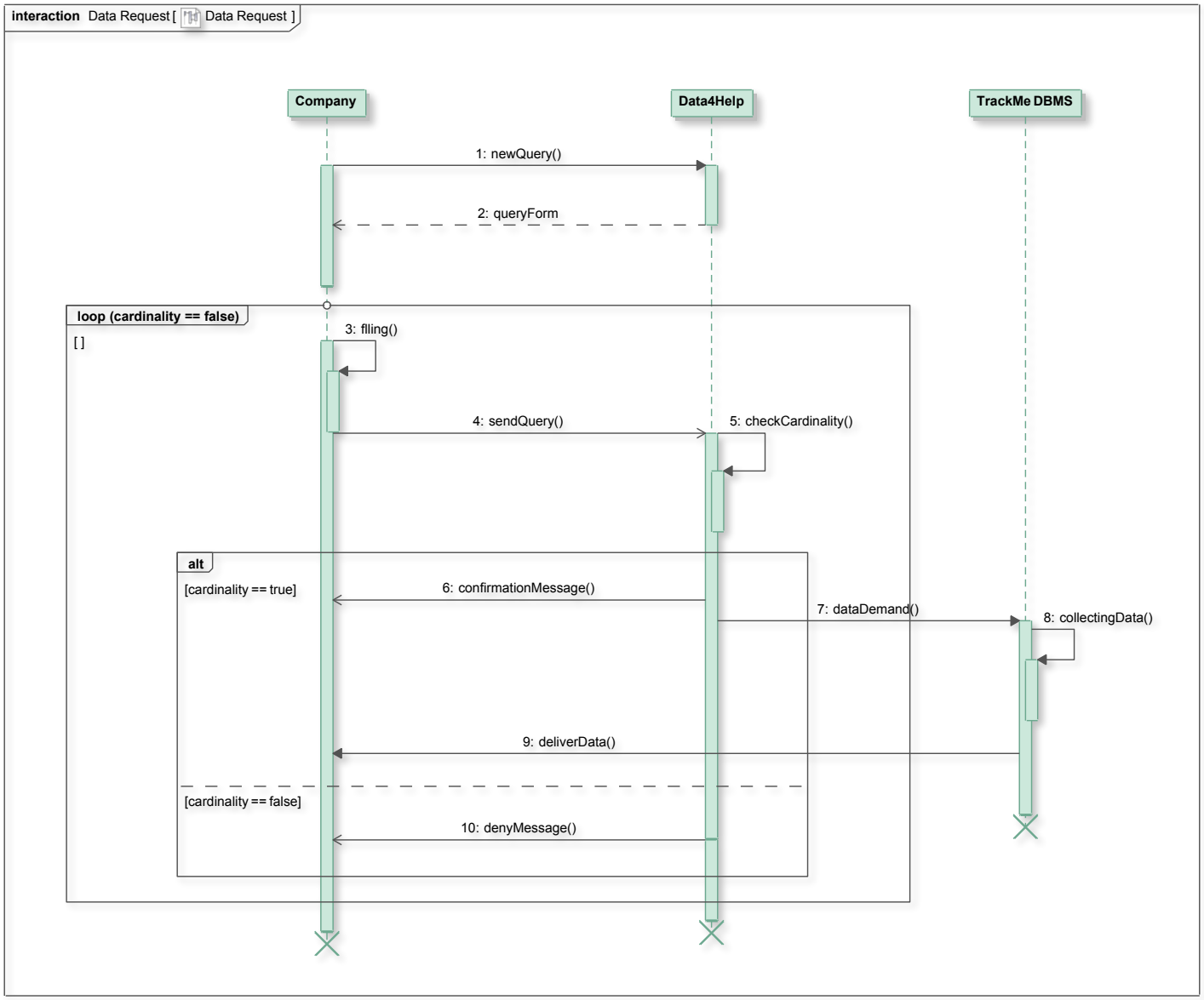

Figure 3.9: Data Request and Supply Sequence Diagram.

Name	Individual Request
Actors	Company, User
Goals	G5, G6
Input Condition	User and Company are logged in.
Event Flow	1. The company sends the individual request, specifying the person's social security number / fiscal code. 2. The user gets the warn and can accept and deny. 3. In the warn is also specified the company information, so that he can contact them.
Output Conditions	User and Company are logged in.
Exceptions	If the social security number or fiscal code is not associated to a Data4Help user, the information in communicated to the the company employee right above the individual research filed.

Table 3.8: Individual Request Use Case.

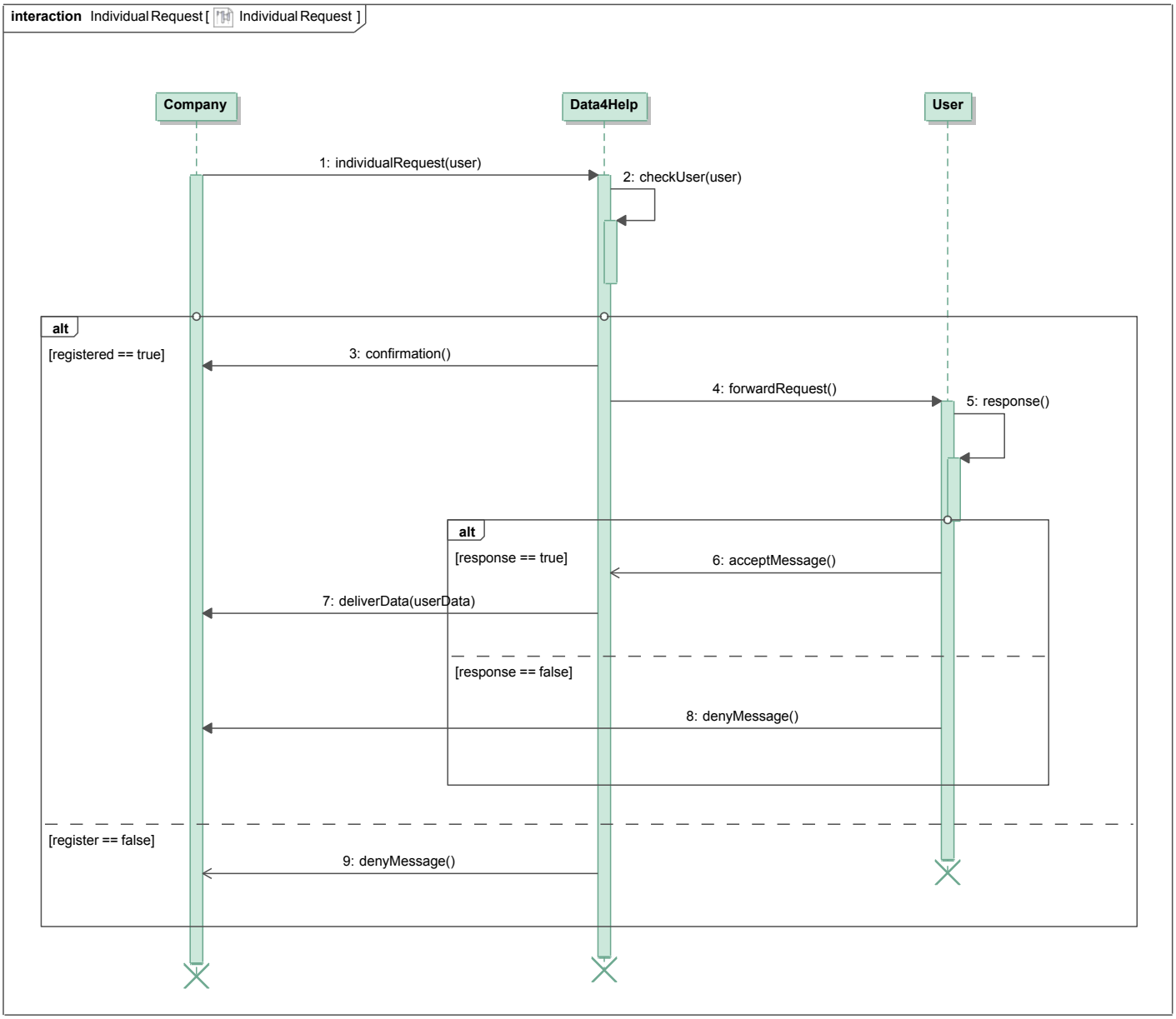

Figure 3.10: Individual Request Sequence Diagram.

Name	Pair a Device and Collect Data
Actors	User
Goals	G1, G3, G4, G7
Input Condition	The user is logged in, but he did not pair a heart tracking device yet.
Event Flow	1. After registration, he is immediately asked to pair his health-tracking device, showing the list of the bluetooth available devices. 2. Chosen the device, the user clicks on the "Pair and Start" button. 3. From now on all parameters are stored in the app and sent to trackMe.
Output Conditions	The app menu is shown, login condition not changed.
Exceptions	If the new user doesn't pair a device, every time he opens the app the pairing is asked.

Table 3.9: Device Pairing and Data Collection Use Case.

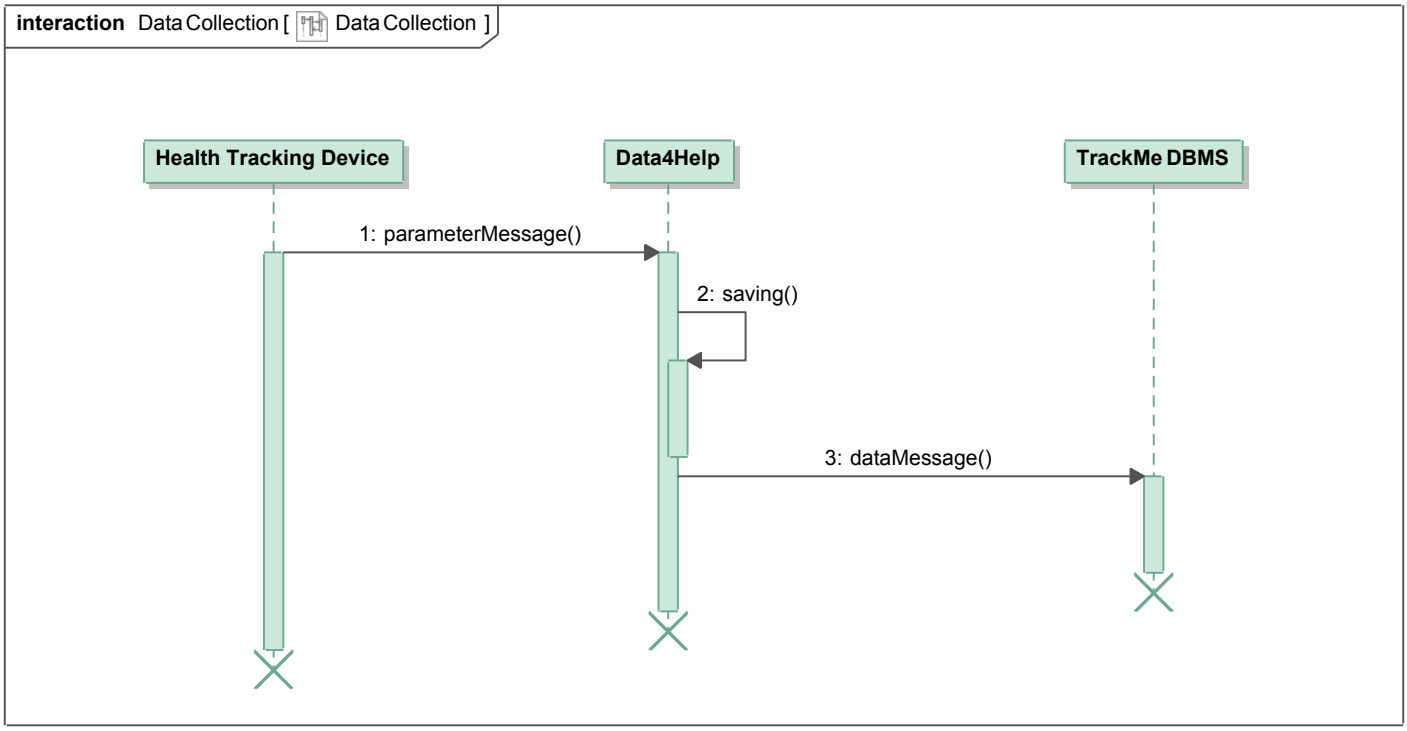

Figure 3.11: Data Collection Sequence Diagram.

Name	See Charts
Actors	User
Goals	G3
Input Condition	The logged-in user is on the home section.
Event Flow	1. He chooses the parameter-type of which he wants to see the track of. 2. The page with the monthly track is showed 3. He can select on the calendar a specific day to the his parameters in that date.
Output Conditions	The user is still logged in.
Exceptions	There are no exceptions.

Table 3.10: Charts Consultation Use Case.

Name	Emergency Call
Actors	External Emergency Calls System, Elder User
Goals	G7, G8
Input Condition	The user is logged in.
Event Flow	1. The health tracking device detects a parameters out of threshold, starts the emergency procedure 2. The external calling system is involved and the ambulance called. 3. The message is shown on the device.
Output Conditions	The user is logged in.
Exceptions	There are no exceptions.

Table 3.11: Emergency Call Use Case.

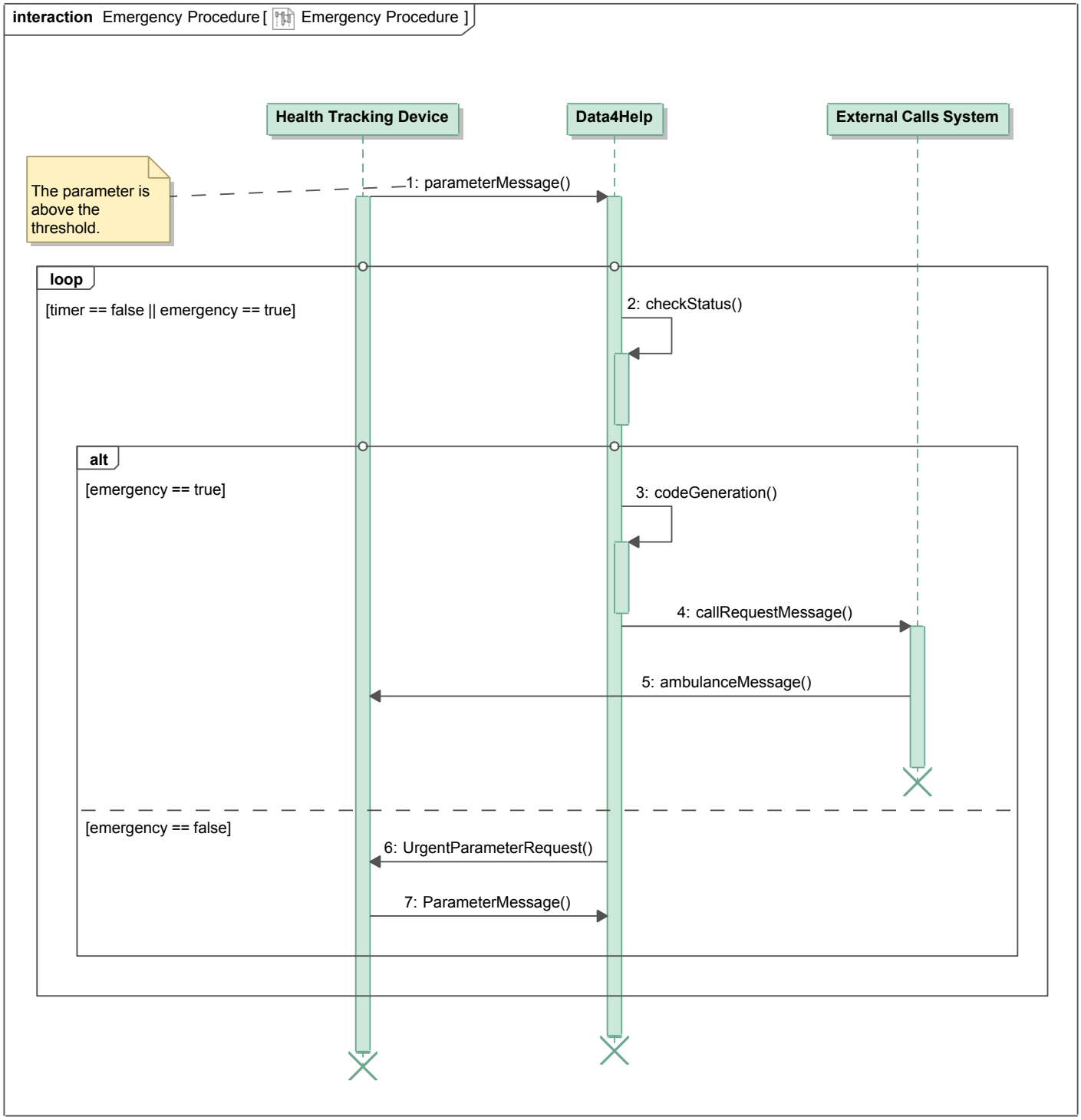

Figure 3.12: Emergency Call Sequence Diagram.

Use Case	Goals	Requirements	User Class
Registration	G2 G3 G5	R9	All
Login	G2	R10	All
Edit Profile	G2	R11	All
Send Data Request	G6	R1 R2	Third Party
Access Response Material	G6	R2 R3 R4 R6	Third Party
Individual Request	G5 G6	R2 R5 R12	Third Party to General and Elderly User
Pair a Device & Collect Data	G1, G3, G4, G7	R7 R8	General and Elderly User
See Chart	G3	R11	General and Elderly User
Emergency Call	G7 G8	R13 R14	Elderly User

Table 3.12: Traceability Matrix.

3.4 Performance Requirements

In view of a large-scale expansion, the system is provided to be connected to fairly great number of users and third parties simultaneously. The system receives queries, requests of subscriptions and has to answer to these demands and it has to manage a big amount of data continuously.
So the database will probably be distributed but very efficient and, using concurrency control, will be possible to manage parallel tasks to improve performances.
Furthermore, the system must not have an upper-bound to the number of registered users.
The performance is essentially bound by the internet connection so timing constraints related to the queries and their answers are optimal in case of a good connection.

3.4.1 Parallel Operations

The database will probably be distributed but very efficient and, using concurrency control, will be possible to manage parallel operations from different users to improve performances.

3.5 Design Constraints

The system should comply to these following minimum hardware requirements:

- Mobile application:
  3G UMTS connection at its maximum speed of 2 Mb/s,
  50 MB of available space,
  1GBofRAM,
  GPS module.

- Web application:
  Internet connection at 7 Mb/s,
  800x600 screen resolution.

The server-side of the application shall be written in Java. Therefore it can be executed on any platform that supports the Java Virtual Machine and runs JEE. The web application must support the latest version of the main browsers like IE, Google Chrome and Firefox. The mobile application must be developed for both Android and iOS architectures.

3.6 Software System Properties

3.6.1 Reliability

The company forecasts a high amount of users, with an high growth curve especially in the first months. The system should support thousands of requests simultaneously. The system should be prepared to be scalable in order to increase the capacity of users it can handle as soon as it's required.

3.6.2 Availability

The application will be accessible online 24/7. The system must offer an availability of 99%. The remaining 1% shall take into account the time spent for ordinary maintenance sessions that should be done when there are few users connected.

3.6.3 Security

The system manages users' personal information: communication between components must be secured using the TCP/IP and TLS protocols together with encrypting communications data using a sophisticated encryption algorithm with public and private key.
Passwords stored in the company database must be hashed and salted.

3.6.4 Maintainability

In order to facilitate further implementation we should adopt a MVC pattern and use best-practises.
The code of the whole software-to-be-developed must be thoroughly documented associated with unit and integration tests in order to let

future developers easily know how the system works and how it has been designed. A version control system must be used to manage and organize all the different code revisions.
Components should be implemented using a modular approach in order to facilitate maintenance and scalability.

3.6.5 Portability

The system is structured to be fully cross-platform. Its functionalities must be accessible by both web interfaces and smartphone applications.

4 Alloy

4.1 Introduction

The UML class diagram is considered a standard diagram in the first phase of modeling software system. However UML doesn't give a complete view of the system and all his relationships we're trying to model. One of the most important lack is the absence of an easy way to show constraints on semantic models. Alloy is a declarative specification language, so it's useful to model static properties of components in Software Design specifics. Alloy integrates UML model, pointing out the relationships between entities and the constraints on these relations.

The system created is simplifying all the software Data4Help and AutomatedSOS. In particular there is strict connection between the Alloy model and the UML class diagram.

Each class is represented as a signature and each constraint as a fact. Assertions check if there are constraints not valid and predicates show the worlds created by the Alloy model.

Otherwise in this Alloy Model there are simplifications about the Location class and the complexity of the Health class. The first should be modeled as a structure of two different floats (longitude and latitude) but Alloy doesn't support the comparison between two signature Float. Eventually Data4Help should support also comparison between floats because it's a possible criterion in the construction of the query.

The Health class probably would contain other parameters but the Alloy model for simplicity checks only one.

4.2 Code

```
open util/integer

--SIGNATURES--

/*What is commented is not added for simplicity
in the world representation*/

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

--Abstract structure for all users subscribed
abstract sig UserSubscribed{
--unique identifier of all users
id: one Int
/*
--name of the company or the user
name: one String,
--password of the selected by the user
password: one String,
-- email used to register the user
email: one String
*/
}
{id>0}

--The third company that asks queries to the system
sig Company extends UserSubscribed{
--all the group queries asked by the company
askedGQ: set GroupQuery,
--all the single queries asked by the company
askedQueries: set Query
/*
--id that identify in a unique way each company
vta: one Int
*/
}

--Individual that download the app and register himself
sig User extends UserSubscribed{
--age of the user
age: one Int,
--gender of the user
```

```
44 gen: one Bool,
   --set of all answer given by the user
46 answers: set Answer,
   --Health tracked of user
48 health: one Health,
   -- Boolean value is True when an ambulance is sent
50 ambulanceSent: one Bool
   /*
52 --id that identify in a unique way each user
   fiscalCode: one String*/
54 }
   {age>0}
56
   --The health status of each user
58 sig Health{
   p1: one Int
60 }

62 /*The possible criteria that a company can choose for
   selecting a precise group of people in a query*/
64 sig Criteria{
   --age criterion
66 age: lone Int,
   --gender criterion: female = 0, male = 1
68 gen: lone Bool}
   {age>0 and
70 --at least one criterion
   (age!=none or gen!=none)}
72
   --Abstract structure that define Query
74 abstract sig GenericQuery{
   --unique for each query
76 idQ: one Int,
   --company that asks this query
78 comp: one Company}
   {idQ >0}
80
   /*The structure of a query from a company that requests
82 a group of people*/
   sig GroupQuery extends GenericQuery{
84 --criteria of the query
   crit: one Criteria,
86 --cardinality of the group
```

```
   card: one Int,
88 --status of the group query
   st: one StatusGQ,
90 --eventually answer
   ga: lone GroupAnswer}
92 {card>1}

94

   --The structure of a query from a company to a single user
96 sig Query extends GenericQuery{
   --identify of the user that has to answer to this query
98 --not the id of the query
   id: one Int,
100 --answer of the query of the specific user
   ans: lone Answer,
102 --status of the query
   status: one StatusQuery}
104

   --There are all the possible status of query is
106  abstract sig StatusQuery{}
   --The query is sent but the answer does not arrived
108 one sig SENT extends StatusQuery{}
   --The query is accepted, the answer from the user is true
110 one sig ACCEPTED extends StatusQuery{}
   --The query is denied, the answer from the user is false
112 one sig DENIED extends StatusQuery{}

114 --There are all the possible status of  group query is
   abstract sig StatusGQ{}
116 --The query is approved because the search is successful
   one sig APPROVED extends StatusGQ{}
118 --The query does not already find the requested group of people
   one sig PENDING extends StatusGQ{}
120

   --Generic answer of a generic query
122 abstract sig GenericAnswer{
   --id that refers to the query
124 idQ: one Int}
   {idQ>0}
126

   --The possibile answer given by a single user
128 sig Answer extends GenericAnswer{
   --True = okay for the user to give this Health, otherwise False
```

```
130 answer: lone Bool,
    --Eventually the required Health
132 health: lone Health
    }
134
    --The possibile answer given by the system to a group query
136 sig GroupAnswer extends GenericAnswer{
    --eventually the set of all requested status
138 healthSet: set Health
    }
140
    --FACTS--
142
    -- No two distinct coinciding users
144 fact OnlyUniqueUserSubscribed{
    no disj u1, u2: UserSubscribed | u1!=u2 implies u1.id = u2.id}
146
    /*
148  For the commented parameters
    -- No two distinct coinciding users
150 fact OnlyUniqueUserSubscribed{
    no disj u1, u2: UserSubscribed |
152 u1!=u2 implies u1.email = u2.email}

154 --No two distinct users with the same fiscal code
    fact OnlyUniqueUser{
156 no disj u1, u2: User | u1!=u2 implies u1.fiscalCode = u2.fiscalCode}


158
    --No two distinct companies with the same vta
160 fact OnlyUniqueCompany{
    no disj c1, c2: Company | c1!=c2 implies c1.vta = c2.vta}
162 */

164 --No health without a user
    fact EachHealthHasUser{
166 all h: Health| h in User.health}

168 --Each health related to one user
    fact HealthHasOneUser{
170 all u1, u2 : User, h:Health | h in u1.health and
     h in u2.health implies u1=u2}
172
```

```
--No answer without a user
fact EachAnswerHasUser{
all a: Answer | a in User.answers}

--No queries without a company
fact EachQueryHasCompany{
all q: Query | q in Company.askedQueries}

--No queries without a company
fact EachGroupQueryHasCompany{
all q: GroupQuery | q in Company.askedGQ}

--No two distinct queries with the same id
fact OnlyUniqueQuery{
no disj q1,q2: GenericQuery | q1!= q2 implies q1.idQ = q2.idQ}

--No two distinct answers with the same id
fact OnlyUniqueAnswer{
no disj a1,a2 : GenericAnswer | a1!=a2 implies a1.idQ = a2.idQ}

--There are not GroupAnswer not related to Group Queries
fact EachGroupAnswerIsInsideAGroupQuery{
all a: GroupAnswer | one q: GroupQuery | q.idQ= a.idQ
<=> q.ga = a}

--There are not GroupAnswer not related to Group Queries
fact EachGroupAnswerHasAGroupQuery{
all a: GroupAnswer | one q: GroupQuery | q.idQ= a.idQ}

--There are not Answer not in a Query
fact EachAnswerHasToBeInAQuery{
all a: Answer | (one q: Query | (q.status != SENT and
((q.idQ= a.idQ) and (q.ans = a))))}

--All Queries have different Answer
fact DifferentAnswers{
no a: Answer , disj q1, q2 : Query | q1.ans = a and q2.ans = a }

--All Users have different Health
fact OnlyUniqueHealth{
no disj u1, u2 : User| u1!=u2 implies u1.health = u2.health}

--All queries have to be done by a company
```

```
fact QueriesAreDoneByCompany{
all q: Query | q.comp!= none and (one c: Company | q.comp = c
 implies (q in c.askedQueries))}

--All group queries have to be done by a company
fact GroupQueriesAreDoneByCompany{
all q: GroupQuery | q.comp!= none and (one c: Company |
q.comp = c implies (q in c.askedGQ))}

--The state of the query must be related to the answer
--Accepted , the anser must be True
fact QueryAccepted {
all q: Query | q.status = ACCEPTED <=> q.ans.answer = True}

--Denied, the answer must be False
fact QueryDenied{
all q: Query | q.status =  DENIED <=> q.ans.answer = False}

--Sent, the answer does not arrive
fact QuerySent{
all q: Query | q.status =  SENT <=> q.ans = none}

--The query has to be related to the correct user
fact QueryAcceptedHasHealth{
all q: Query | q.status = ACCEPTED <=>
( one u: User | u.id = q.id )}

--The query has to be related to the correct health status
fact AnswerHasTheRightResult{
all q: Query | q.status = ACCEPTED <=>
( one u: User | u.id = q.id implies u.health = q.health )}

--The state of the group query must be related to the answer
--Approved, the search found results
fact QueryApproved {
all q: GroupQuery | q.st = APPROVED <=> (q.ga!= none )}

--Denied, the search did not find the requested users
fact QueryDenied{
all q: GroupQuery | q.st =  PENDING <=> (q.ga= none )}

--There are not two answers related to the same Group Query
fact OnlyUniqueGroupQueryAnswer{
```

```
no disj q1,q2: GroupQuery | q1!= q2 implies q1.ga = q2.ga}
260
--The answer of the query is contained in the same query
262 fact SameIDImpliesThatAreRelated{
all q: GroupQuery | q.st = APPROVED <=> (one a: GroupAnswer |
264 a.idQ=q.idQ implies q.ga = a) }

266 --The cardinality requested is respected by the answer
fact GQHasToHaveTheRightCardinality{
268 all q: GroupQuery | q.st = APPROVED <=>
( q.card = #q.ga.healthSet) }
270
--The answer of the query must contain the right Health of the users
272 fact GQHasToHaveTheRightHealthAnswer{
all q: GroupQuery | q.st = APPROVED <=>
274  (all h: q.ga.healthSet | h!= none and
( one u: User | h = u.health))}
276
/*The Health contained in the answer must be
278 related to the criteria inserted in the query*/
fact RightHealthSetRespectTheCriteria{
280 all q: GroupQuery | q.st = APPROVED <=>
(all h : q.ga.healthSet | h!= none and (one u: User | u.health = h
282 implies ((q.crit.age!= none implies u.age = q.crit.age)
and (q.crit.gen != none implies u.crit.gen = u.gen))))}
284
--No users that satisfy the age criterion cannot be in the answer
286 fact AllUserThatSatisfyTheAgeCriteriaHasToBeInTheAnswer{
all q: GroupQuery | q.st = APPROVED and q.crit.age != none implies
288 ( all u: User | ( q.crit.age = u.age ) <=>
( one h: q.ga.healthSet | u.health = h))}
290
--No users that satisfy the gender criterion cannot be in the answer
292 fact AllUserThatSatisfyTheGenCriteriaHasToBeInTheAnswer{
all q: GroupQuery | q.st = APPROVED and q.crit.gen != none implies
294 ( all u: User | ( q.crit.gen = u.gen ) <=>
( one h: q.ga.healthSet | u.health = h))}
296
/*When the parameter p1 of the Health of an old user is
298 under the threshold the boolean value is True
For simplicity 5 is the threshold and a person is old (60+)
300 if has age >= 2*/
fact AmbulanceSent{
```

```
all u: User | (u.health.p1 <5 and u.age >= 2) <=>
 u.ambulanceSent = True}

--ASSERTIONS--

--There are not single answer related to Group Queries
assert NoAnswerRelatedToTheWrongQuery{
no a: Answer | one gq: GroupQuery | gq.idQ= a.idQ}

--There are not GroupAnswer related to single Queries
assert NoGroupAnswerRelatedToTheWrongQuery{
no a: GroupAnswer | one q: Query | q.idQ= a.idQ}

--There are not health connected to wrong queries
assert NoWrongConnectionBetweenQueriesAndHealth{
all q : Query | q.status != ACCEPTED and q.health = none}

--There are not Group Answer that are not connected to a Group Query
assert NoWrongConnectionBetweenGQAndGA{
no q: GroupQuery | q.st = APPROVED and
( one a : GroupAnswer | a.idQ = q.idQ and q.ga != a)}

--PREDICATES--

--Symple show that creates the World 1
pred sympleQueryShow{
some u: User | u.age = 3 and u.health.p1 = 4
#Health <= 2
#Company =1
#Criteria = 0
#Query = 2
#Answer = 2
}

pred sympleGQShow{
#User = 2
#Health = 2
#Company =1
#Criteria = 1
some q1: GroupQuery | q1.crit.age = 7
#GroupAnswer = 1
}
```

```
run sympleQueryShow for 6 but exactly 10 String
run sympleGQShow for 6 but exactly 10 String

check NoGroupAnswerRelatedToTheWrongQuery
check  NoAnswerRelatedToTheWrongQuery
check NoWrongConnectionBetweenQueriesAndHealth
check  NoWrongConnectionBetweenGQAndGA
```

4.3 World and Conclusions

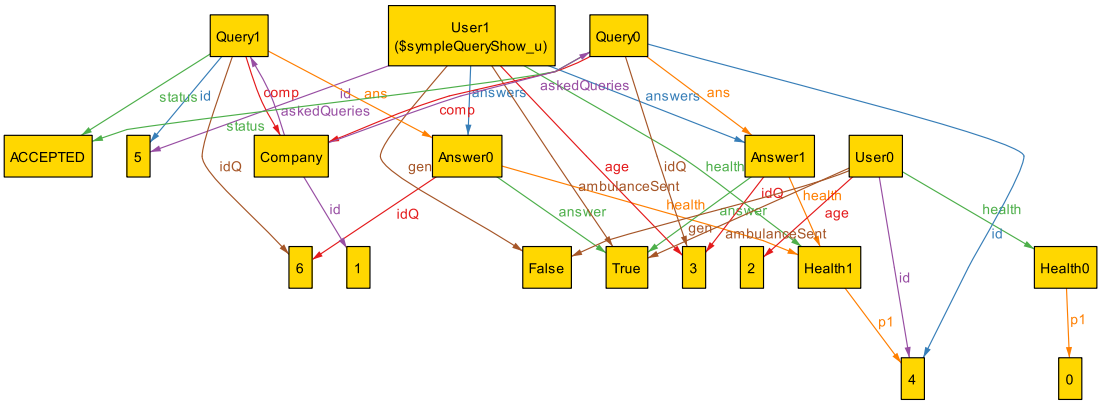

Figure 4.1: Simple Query Show.

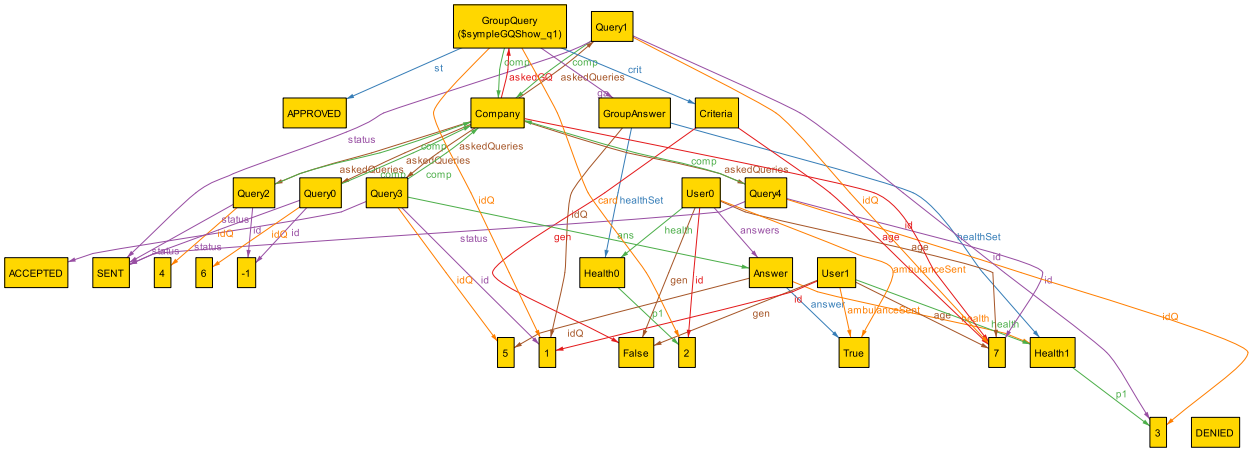

Figure 4.2: Simple Group Query Show.

The worlds represented are simplification of the possible worlds generated by this Alloy model.
Each query (single one) is related to an answer and also a group query to a group answer.
Every user has his Health and set of Answer, as the Company has his sets of queries and group queries.
In the figure 4.1 there are queries that want each one a specific Health a from specific user. Each Answer contains the right Health and the queries are accepted. Otherwise if the user is not present the query status is denied.
In this specific case the users have the health parameter under the threshold (that is simplified as 5) and they are over 60 (age >=2) so the field "ambulanceSent" is set equal to True.
In the figure 4.2 the company requires a group query under some criteria (in the example age = 7). So, the Group Answer contains all the users that satisfy this constrains and checks the cardinality.
Furthermore, there are also other 4 queries, 3 of them are only sent in fact there not Heath related to them and one is accepted and contains the corresponding Health. Also in this case the users are over 60 and their parameters are under the threshold so the field "ambulanceSent" is set equal to True.

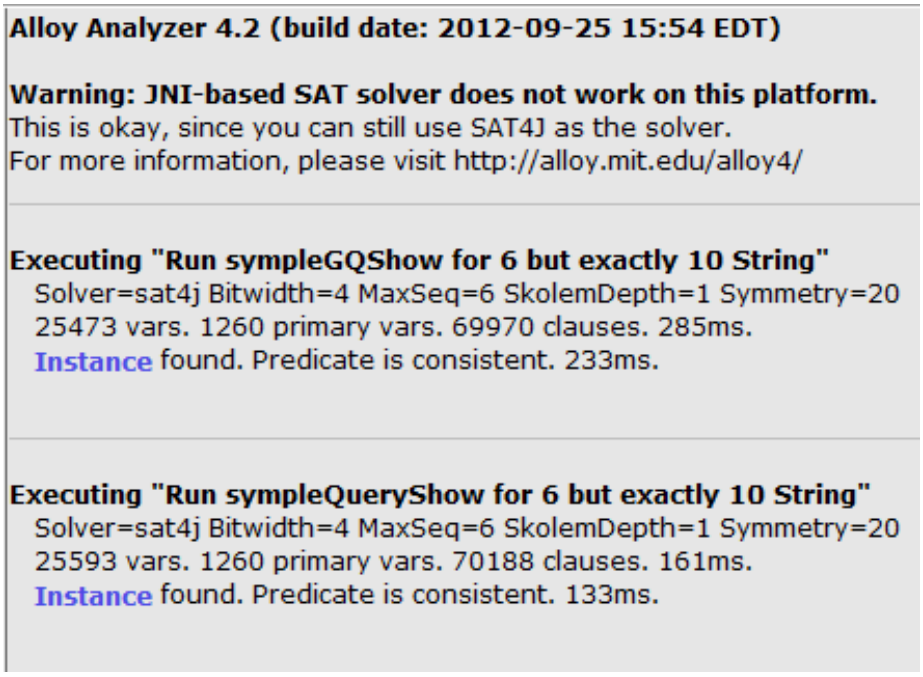

Figure 4.3: Alloy Result.

5 Effort

Date	Eugenio Cortesi	Chiara Criscuolo
27/10/18	4	3
28/10/18	4	3
29/10/18	5	5
31/10/18	2	2
1/11/18	2	3,5
2/11/18	4,5	3
7/11/18	4	5
8/11/18	6	5
9/11/18	6	6
10/11/18	2	4
11/11/18	4	4

Table 5.1: Effort Spent.

Change-log

- 1.2.1 Description: information about the profit deriving from the creation of the system was missing.

- 3.1.1 User Interfaces: which mockups are previewed have been changed.

- 3.1.4 Communication Interfaces: information on the internet protocol used was missing

- 3.12 Traceability Matrix: some boxes were wrongly associated.

- 3.4.1 Parallel Operations: information about the this type of operations management was missing.

- 4 Alloy: revision and updates. Added E*achHealthHasUser*, *HealthHasOneUser*, *EachAnswerHasUser*, *EachQueryHasCompany*, and *EachGroupQueryHasCompany* facts.