POLITECNICO DI MILANO



SOFTWARE ENGINEERING 2 COURSE

**The TrackMe project**

**Requirements Analysis and Specification Document**

Kevin Mato, Antonio Mazzeo

Github: https://github.com/antoniomazzeo/MatoMazzeo

DATA

**Summary**

[**1** **Introduction** 2](#_Toc529211192)

[**1.1** **Purpose** 3](#_Toc529211193)

[**1.2** **Scope** 3](#_Toc529211194)

[**1.3** **Definitions, Acronyms, Abbreviations** 4](#_Toc529211195)

[**1.4** **Revision History** 5](#_Toc529211196)

[**1.5** **Reference Documents** 5](#_Toc529211197)

[**1.6** **Document Structure** 5](#_Toc529211198)

[**2** **Overall Description** 6](#_Toc529211199)

[**2.1** **Product Perspective** 6](#_Toc529211200)

[**2.2** **Product Function** 7](#_Toc529211201)

[**2.3** **User Characteristics** 7](#_Toc529211202)

[**2.4** **Domain assumptions, dependencies and constraints** 7](#_Toc529211203)

[**3** **Specific Requirements** 10](#_Toc529211204)

[**3.1** **External Interfaces Requirements** 10](#_Toc529211205)

[**3.2** **Functional Requirements** 20](#_Toc529211206)

[**3.2.1 Scenarios** 20](#_Toc529211207)

[**3.2.2** **Use Case** 21](#_Toc529211208)

# **Introduction**

This section will explain which the main scopes of the TrackMe system is and will provide a general overview of all the features involved.

## **Purpose**

This document is the RASD (Requirement Analysis and Specification Document) and its main purpose is to describe the TrackMe system, its components, constraints, functional and non-functional requirements and relationships with third party software.

This document will also provide different scenarios along with typical user use cases, a structure of the system using UML diagrams and a formal description of application features through alloy language.

This document is aimed at giving a detailed system specification to developers, testers and quality assurance.

## **Scope**

The scope of the entire project is to develop a new mobile service, TrackMe, composed of a core service invisible to the end users, Data4Help, compatible with the most popular smartwatches and bands available in the market which stores and shares health and location data with third party services.

This document will particularly focus on the interaction with AutomatedSOS, a software that will be built on top of Data4Help, aimed at helping people by contacting an ambulance in case of emergency (i.e. heart beat under or above the safe parameters).

Data4Help must share and store user data only by explicit consent respecting the General Data Protection Regulation (GDPR).

AutomatedSOS must constantly monitor user’s health, if the values go outside the safe intervals given by the WHO (World Health Organization), it must contact an ambulance.

This RASD (Requirement and Analysis Document) aims at describing the domain in which our system is going to work and determine the main use cases for this system.

This document will also define the primary users of the system, why the system is developed, the condition in which this is going to operate. Moreover, we will illustrate the functional and non-functional requirements for a better understanding of the system.

The **goals** achieved by this solution are:

[G1] Third parties can analyze users individually or in groups.

[G2] Third parties are constantly up-to-date on new data about the users.

[G3] Users can decide whether to share or not their location with the system.

[G4] Users can decide whether to share or not their health parameters to the system.

[G5] Users must receive automatically help if there is an anomaly in their health parameters.

[G6] The user can be recognized by providing a form of identification.

## **Definitions, Acronyms, Abbreviations**

**Definitions**

* Platform: system/application as a whole.
* User: An end user who is currently registered to the Data4Help application and has credentials to access.
* Guest: Person not yet registered and with limited access to features.
* Framework: Reusable set of libraries or classes for a software system.
* Cross-Platform: software able to run on different platforms.
* Partner: A third-party system using Data4Help data.
* Health data: User’s heart beat data.

**Acronyms**

* RASD: Requirements Analysis and Specification Document
* DB: Database
* DBMS: Database Management System
* OS: Operating System
* HTML: Hypertext Markup Language
* CSS: Cascading Style Sheets
* JS: JavaScript
* JSON: JavaScript Object Notation
* API: Application Programming Environment
* HTTP: Hypertext Transfer Protocol
* HTTPS: Hypertext Transfer Protocol Secure
* TCP: Transmission Control Protocol

**Abbreviations**

* G[n]: n-th goal
* R[n]: n-th functional requirement
* D[n]: n-th domain assumption
* UC[n]: n-th use case

(Eliminare definizioni, acronimi e abbreviazioni non utilizzati prima della consegna)

## **Revision History**

Version, date and summary

|  |  |  |
| --- | --- | --- |
| **Version** | **Date** | **Summary** |
| 1.0.0 | Data prima consegna | First Release |

## **Reference Documents**

The following documents were used:

1. Assignment document: <https://beep.metid.polimi.it/documents/121843524/3f744351-7378-4162-86b0-45eddaf10713>
2. IEEE Std 1016tm-2009 Standard for Information Technology-System Design-Software Design Descriptions.
3. IEEE Std 830-1998 IEEE Recommended Practice for Software Requirements Specifications.

## **Document Structure**

The document is divided in the following sections:

* **Section 1:** General description, overview and goals of the TrackMe applications (Data4Help and AutomatedSOS).
* **Section 2:** Overall description of the software features and implemented functions. Description of world and shared phenomena, constraints, domain assumptions and dependencies.
* **Section 3:** Characterization of functional and not functional requirements.
* **Section 4:** Formal model of the main aspects of the application using Alloy modelling language.
* **Section 5:** Effort spent for writing the project.
* **Section 6:** References and software used for the creation of the system.

# **Overall Description**

## **Product Perspective**

Both Data4Help and AutomatedSOS will be cross-platform web application which will be able to run on every mobile device.

AutomatedSOS will gather data from Data4Help through custom API.

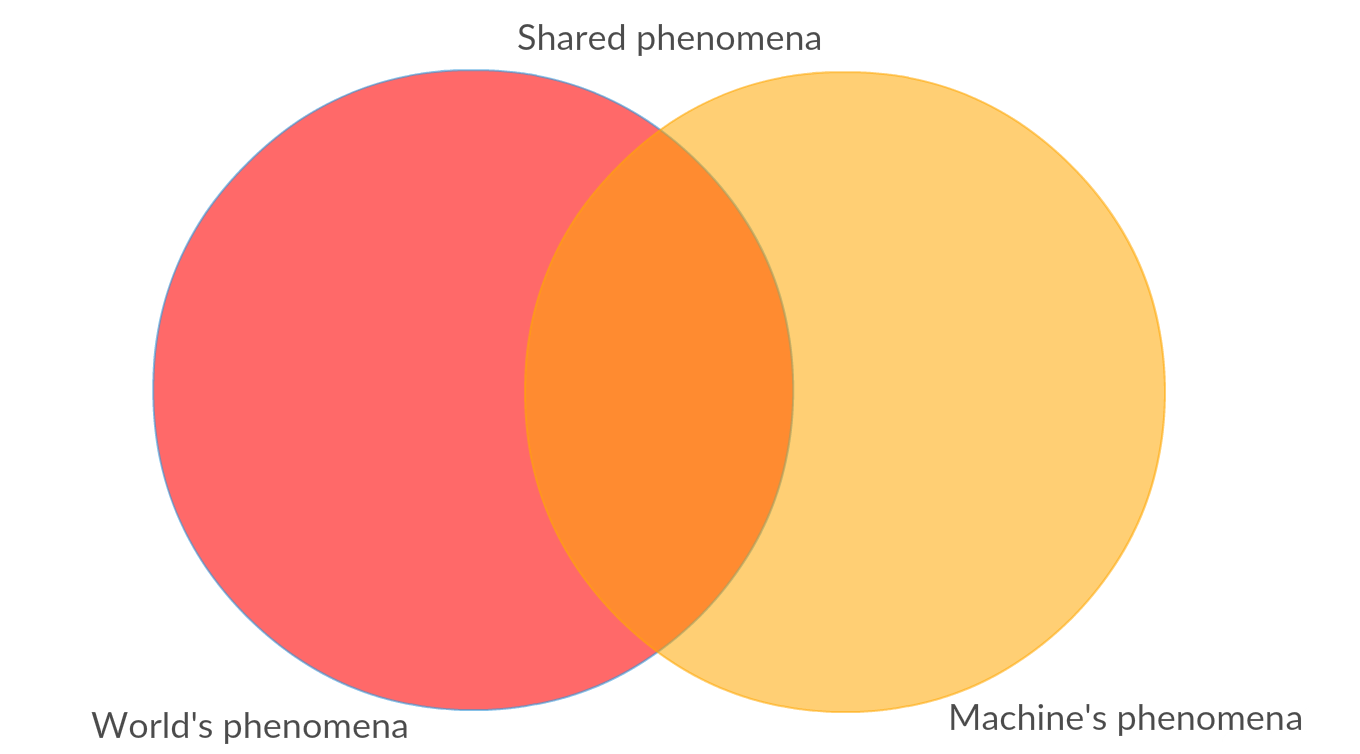


Figure 2.1.1 Relation between world and machine phenomena

**World** is everything that is outside the system scope and **machine** isthe opposite. Therefore, **world’s phenomena** are all the external events happening in the outside world and **machine’s phenomena** are events related to the system. **Shared phenomena** are the intersection of the two i.e. events observable by both parts.

What can be a **world’s phenomenon**?

For instance:

1. The weather for a certain day is extremely hot, accompanied by a high humidity in the air.
2. The diet followed by the user is rich in unsaturated fats.
3. Emergency office getting help calls.

What can be a **machine’s phenomenon**?

1. The registration of a new user in the database.
2. A thread spins up to save an event, after the computation of an anomaly.
3. Database queries.

What can be a **shared phenomenon**?

1. Locating a user.
2. Notifying an ambulance with the address of a user.
3. User’s health data.

## **Product Function**

Our system should be very simple to use and its strong points should be the ease with which the users could understand the statistics concerned about their health. The third parties should also have data about single and group of users respecting the constraints (see section [2.4](#_Domain_assumptions,_dependencies) ).

The system should also be able to:

1. Let the user register and log in.
2. Ask users for their data on behalf of third parties.
3. Let a user choose and edit his privacy preferences.
4. Make itself easy-to-use to everyone.
5. Compute aggregations of users according third parties’ requests.
6. Give a real time evaluation of the data.
7. Notify a third party if certain data are not available and why.
8. Interact with sharing services.
9. Send emergency help to users in danger.
10. Avoid any privacy misuse of data.
11. Understand and store location and health status of a user.
12. Can identify people as unique entities thanks to the information they provided.

## **User Characteristics**

To use the application, no special skills are required. To understand the data and the information computed by the system a user needs a certain level of knowledge, which differs from what usage he wants to make.

The actors in the system are:

* **Guests**: People not registered yet. Guests cannot access any application functionality apart sign up.
* **Users**: People currently registered to the system, they can log in and access full service with no restriction.

The **users**, in the perspective of Data4Help,can be of two types:

* **Third parties:** They have higher privileges in order to perform queries and access data through custom APIs.

Third parties can request an account contacting directly Data4Help providing a legit VAT identification number.

* **The normal user**: Final user that can perform normal tasks (see section [3](#_Scenarios,_use_case)).

From AutomatedSOS perspective we have just **normal users**.

## **Domain assumptions, dependencies and constraints**

**Constraints**

1. **Hardware limitations:** The client application under study must run on every mobile device as smartphones and tablets, regardless of the OS (i.e. Android, iOS, …).

For AutomatedSOS the user must have a smartwatch or smartband with heart beat monitor capabilities. Unfortunately, there’s no standard API to gather information from such devices, so only the most famous companies’ devices will be supported (Fitbit, Samsung, Garmin, Xiaomi, Sony).

1. **Interface to other applications:** As said in the previous point, the application needs to interface itself with other third-party applications to read health data from the user.
2. **Parallel operations:** The application must be able to handle not blocking parallel requests from several users grating a high reactivity.
3. **Availability limitations:** The system’s uptime and availability must be around 99.999%, corresponding to a maximum downtime of 5 minutes in the whole year, since the application deals with people’s lives.
4. **Privacy constraints:** The application must only handle data that the user previously accepted to share.

Moreover, a third-party can access data of single or group of users only if:

1. The single user accepts the third-party request.
2. The cardinality of the group of users fitting the given criteria is greater or equal than 1000.
3. **SOS constraint:** AutomatedSOS must contact the emergency office in less than 5 second.

**Domain assumptions and Dependencies**

[D1] **Health-smart device**.  
All the users have a device that collects health data.

[D2] **GPS devices.**All users have a device that possesses a built-in GPS functionality.

[D3] **Connectivity**.  
All the devices that take part to the system, have a stable connection.

[D4] **Veracity of the data.**  
All the information about the users are correct.

[D5] **Well-rounded algorithms.**  
The algorithms for anomaly detection are infallible.

[D6] **Help** **availability in time.**  
Ambulances can be contacted anytime.

[D7] **Help Range.**  
Ambulances are spread on the areas of the users of the system in the most optimized way.

[D8] **Data Mining possibilities.**  
Users give all the personal information needed to be clustered in groups thanks to different objective functions.

[D9] **Privacy Regulations.**  
There are strict regulations about privacy.

[D10] **Big Data oriented.**

All the information is structured.

[D11] **There are no privileged users apart third parties.**

[D12] **Every user is independent from each other.**

[D13] **Every user has several contact info.**

[D14] **Timestamps sharing.**  
Every agent in the system share the same time.

[D15] **API Availability.**  
Whatever will interact with the system has available API.

[D16] **Battery Duration.**  
Every user keeps its devices charged continuously.

[D17] **OS Permissions Granted.**   
The user will always grant to his OS’s device the permission to access to all the needed services.

[D18] **Sharing services availability.**  
The sharing services interacting with the system have a greater uptime than the system itself.

[D19] **Host Availability.**The system has a greater uptime than the single clients.

[D20] **Sim credit status.** The user has always enough credit to send at least a SMS

# **Specific Requirements**

## **External Interfaces Requirements**

**User Interface**

The user interface must be user-friendly to guarantee to the final user an easy way to interact with the application. The development of the front end is realized through HTML and CSS to provide a wide range of combability among devices.

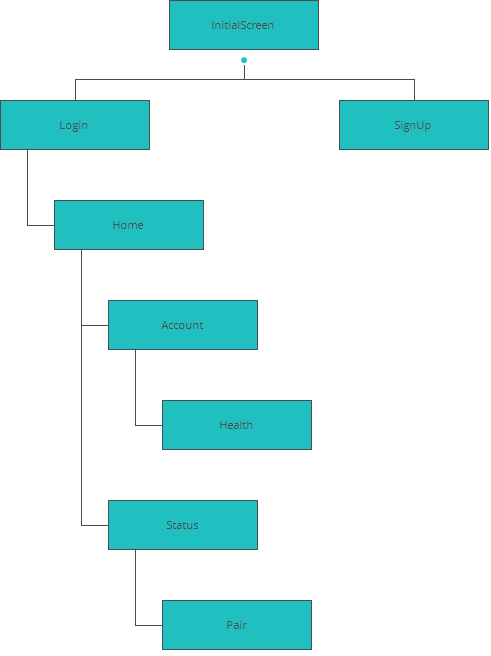


Figure 3.1.1 app map

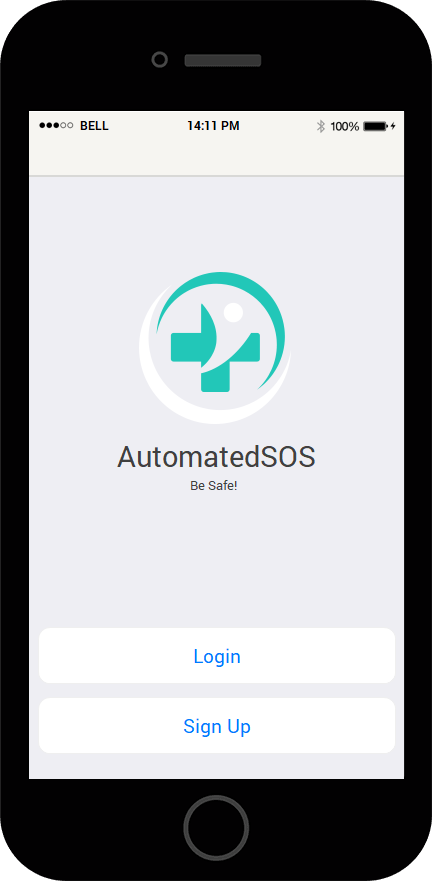


Figure 3.1.2 InitialScreen

Figure 3.1.3 SignUp

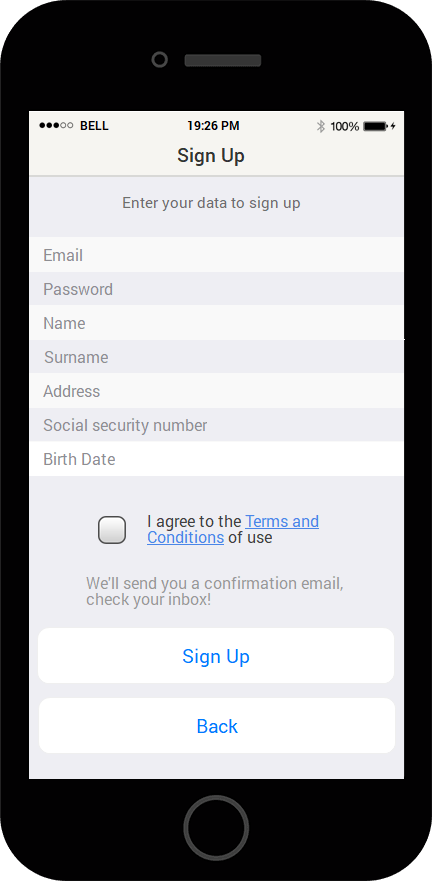


Figure 3.1.4 Login

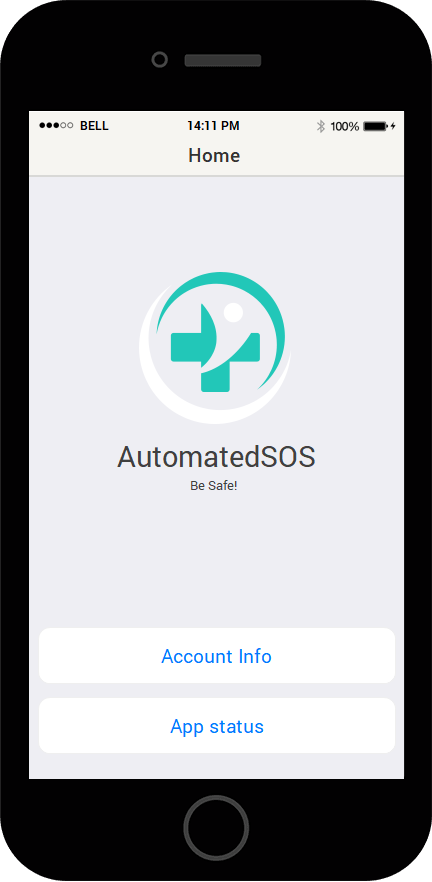
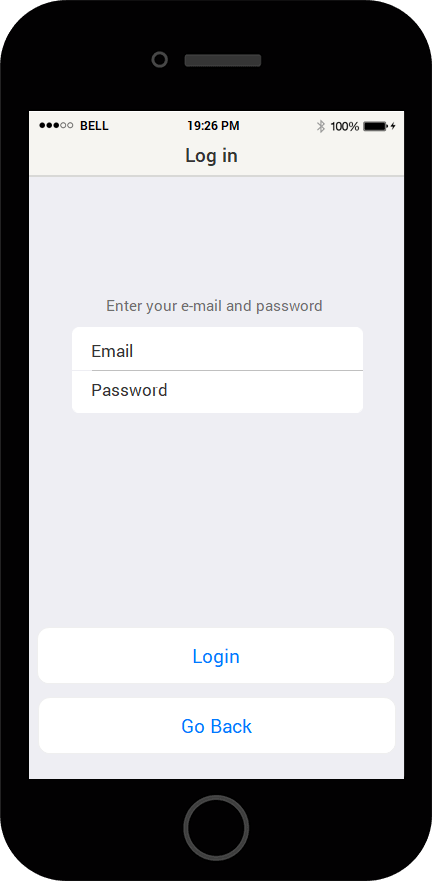


Figure 3.1.5 Home

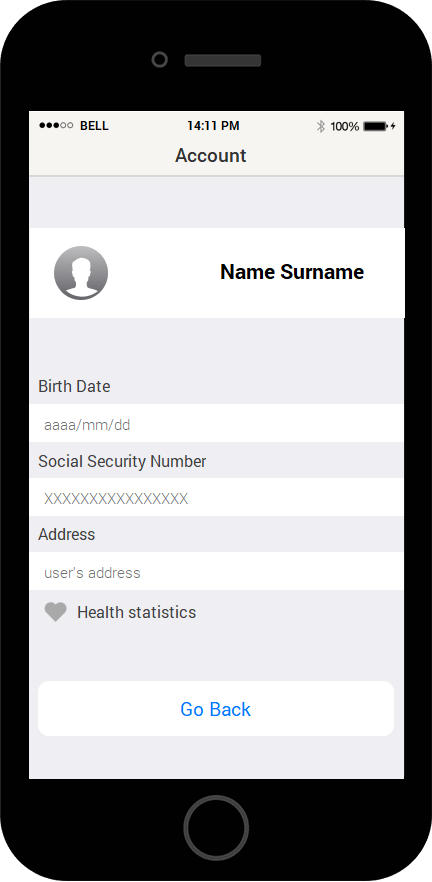


Figure 3.1.6 Account

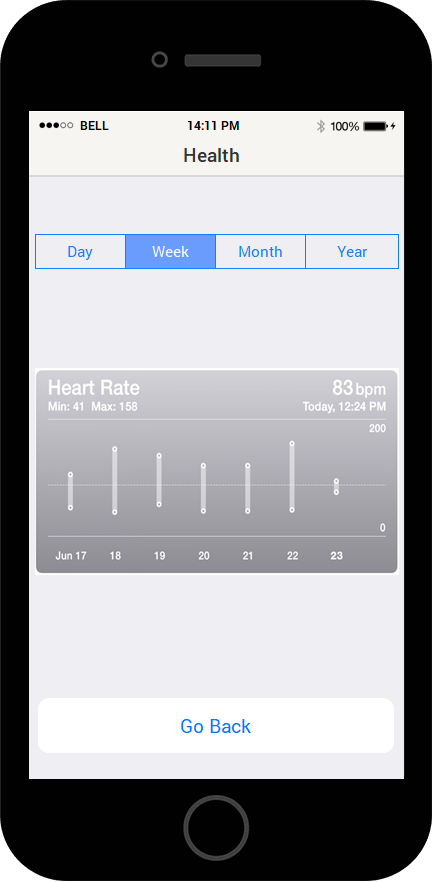


Figure 3.1.7 Health

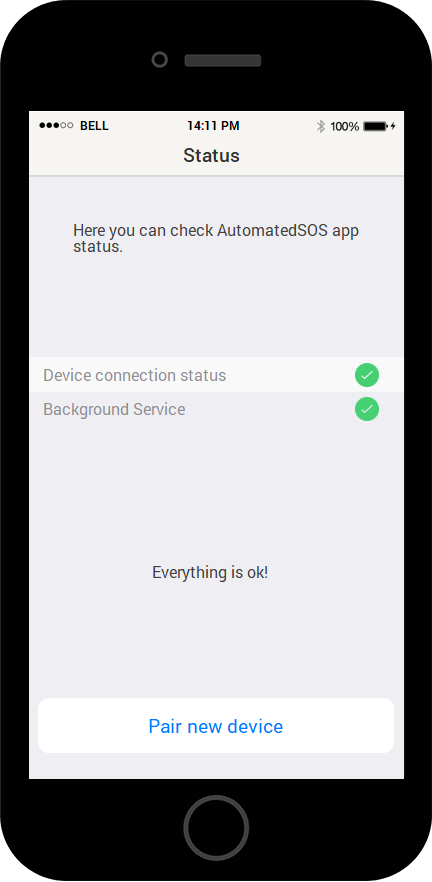


Figure 3.1.8 Status

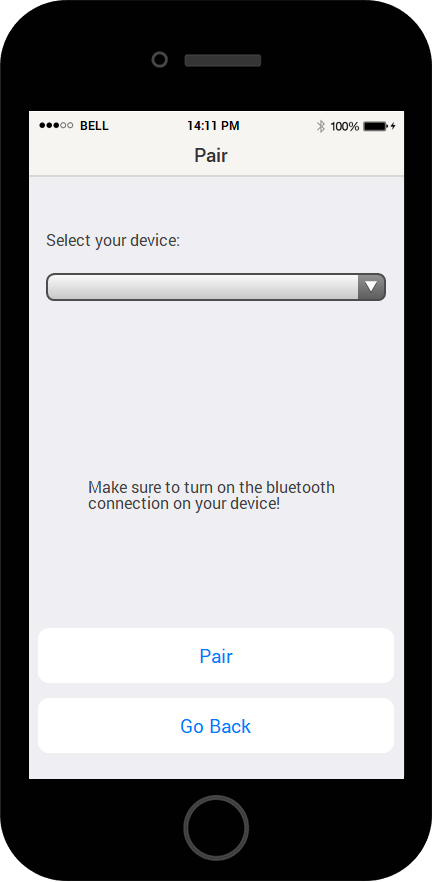


Figure 3.1.9 Pair

**Fig. [3.1.2]**

Welcome screen, the user can login if it was already registered.

**Fig. [3.1.3]**

Sign up screen, here the user can register to the system.

The user can see a detail of privacy, data shared and conditions of use clicking the underlined link.

**Fig. [3.1.4]**

Log in screen, the user must insert his credentials to log in.

**Fig. [3.1.5]**

Home screen, this is the main entry point to all app’s functionalities.

**Fig. [3.1.6]**

Account screen, here the user can review his own data.

**Fig. [3.1.7]**

Health screen, here the user can review his own heart rate statistics ordered by time.

**Fig. [3.1.8]**

Status screen, the user can view the app status to see if the app is properly working and if the health monitoring device is connected.

**Fig. [3.1.9]**

Pair screen, the user can pair a new health device with the application.

**N.B.**

The end user, at first time access, must complete a tutorial step in which the app will guide him/her to connect to the health monitor device he/she owns.

**Software Interface**

The system will use the following programming, markup and representation languages:

* JavaScript
* HTML
* CSS
* Python
* SQL
* JSON

**Communication Interface**

Communication between client and server happens through HTTPS by TCP protocol using the standard port 443.

The system must preserve user’s privacy, all the communications involving sensitive data must be held in a crypted channel.

## **Functional Requirements**

### 

### **3.2.1 Scenarios**

**AutomatedSOS:**

#### Scenario 1

Susan decides to download AutomatedSOS app from the store after a commercial got her eye, out of curiosity she installs it, signs up and tries it out.

She thinks that it would be a great app for her beloved grandfather John, he lives alone and he needs help in case of emergency, so she helps him installing the app and setting it right.

#### Scenario 2

John, Susan’s grandfather, after some time using AutomatedSOS, breaks his smartwatch. He goes to the store, buys a new one and pairs the new device with the app.

#### Scenario 3

Mark is a good athlete, he trains really hard every day. Unfortunately, his father’s family has heart attack history, even though Mark is perfectly healthy, he wants to monitor his health anyway. He hears from a close friend about AutomatedSOS, so he goes to the store, downloads and installs it.

After some time, he wants to check his health statistics, so he goes into his account information and checks it.

**Examples of interactions with other third-party software:**

#### Scenario 4

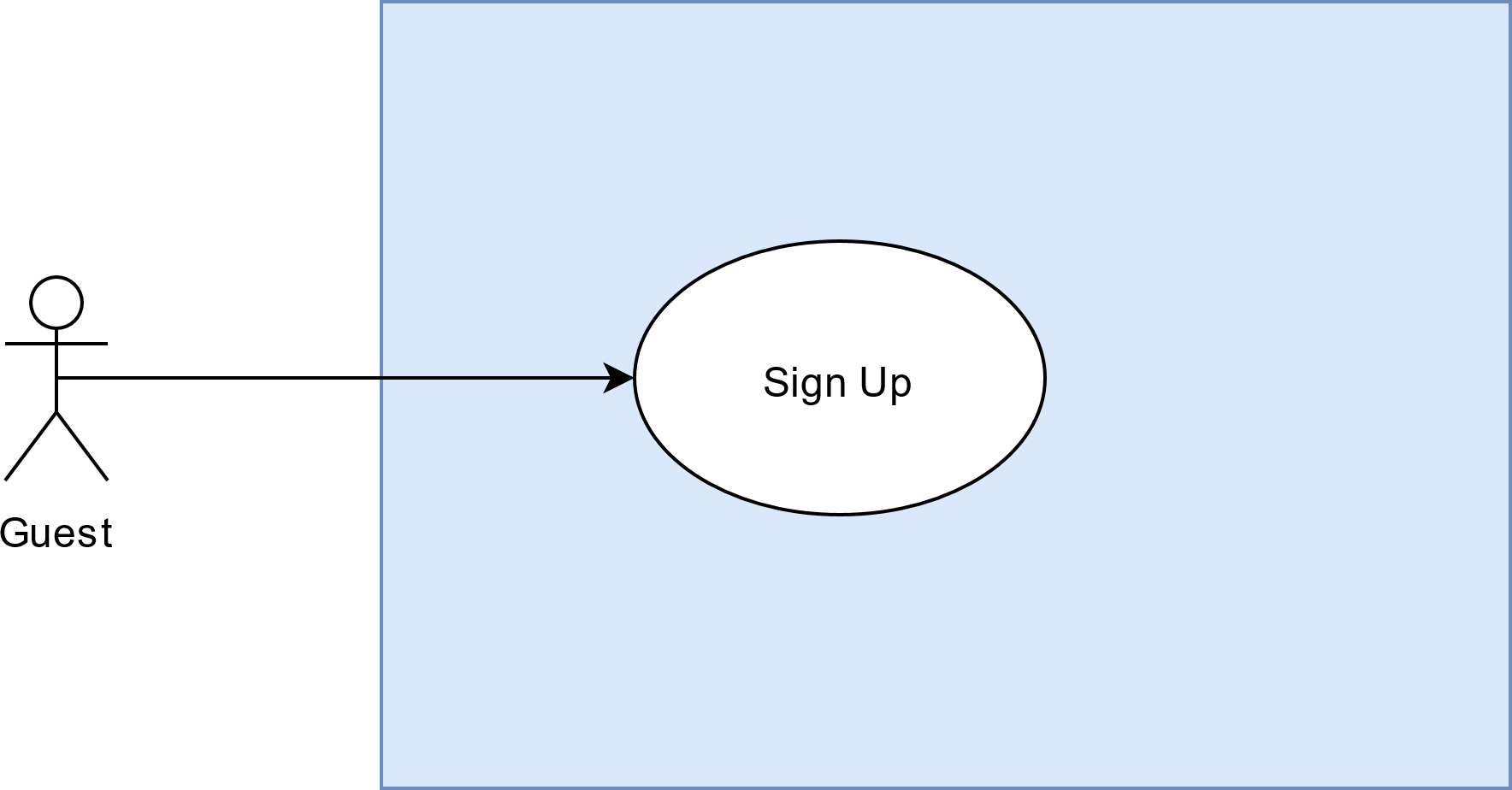
Track4Run is a third-party software connected with Data4Help, it’s capable of tracking athletes participating in a run, it allows organizers to define the path for the run, participants to enrol to the run and spectators to see on a map the position of all runners during the run.

#### Scenario 5

LiveStatistics is another third-party software connected with Data4Help, it can track health data across the globe giving live estimations for each country.

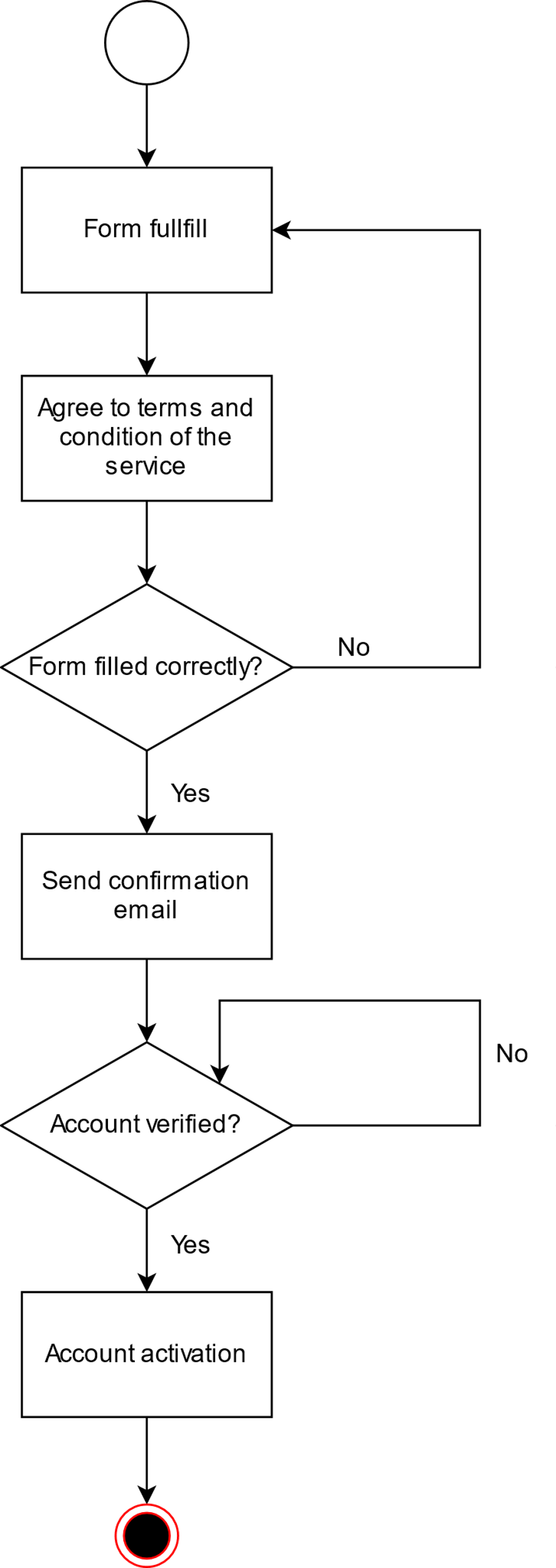
### **Use Case**

**UC[1] Sign Up:**



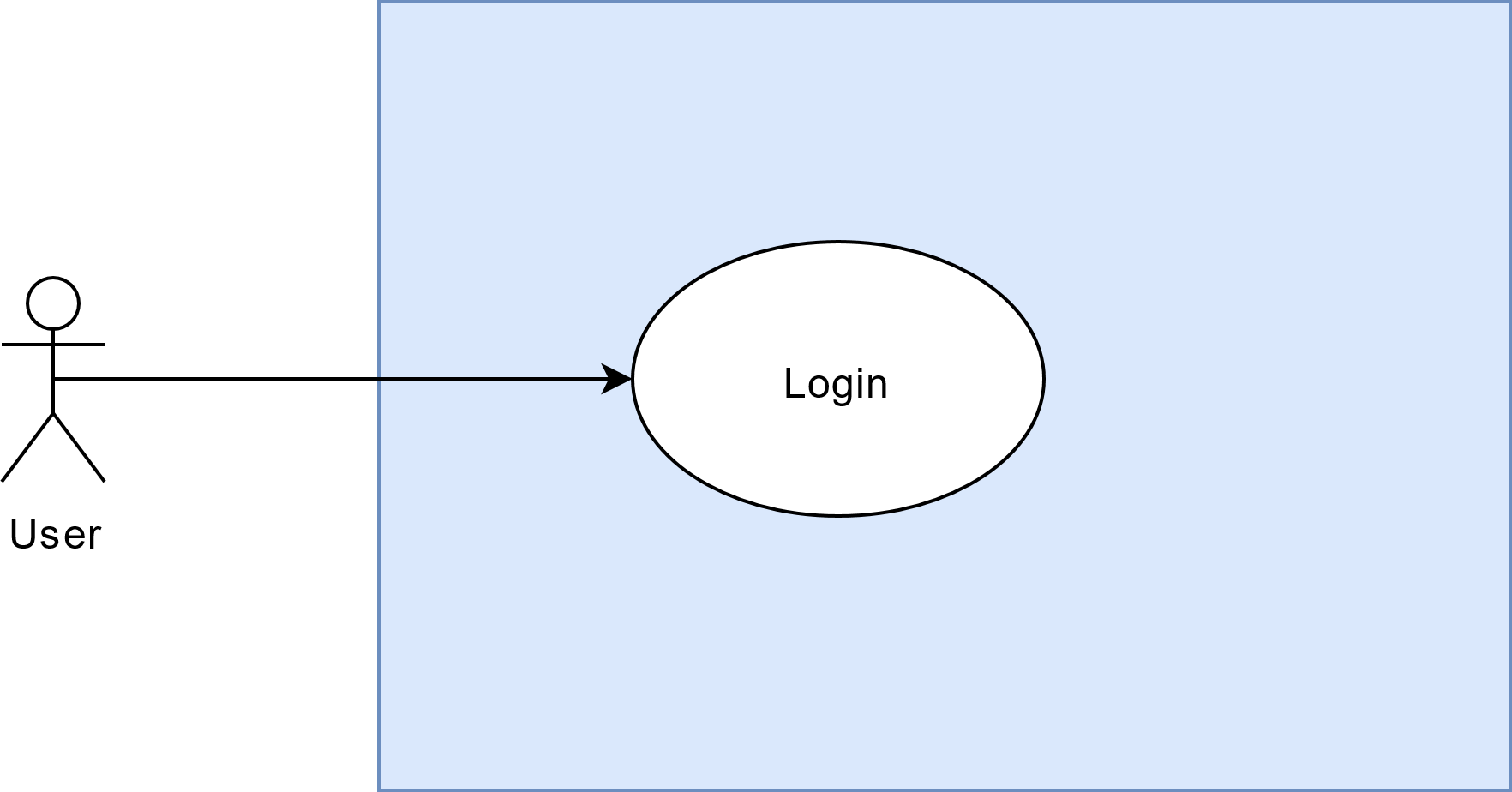
UC[1] Sign Up

|  |  |
| --- | --- |
| Name | Sign Up |
| Actor | Guest |
| Entry Condition | The actor hasn’t got any account on the system |
| Goal |  |
| Event Flow | 1. The guest downloads and installs the application 2. The guest opens the application 3. The guest clicks the “Sign Up” button (See Fig.3.1.2) 4. The guest fills the provided form (See Fig.3.1.3) and click the “Sign Up” button 5. The guest opens the confirmation email and clicks on the attached link |
| Exit condition | The actor becomes a user of the system and can use the application |
| Exceptions | Some form fields get incorrectly filled or left empty, the system notifies the user on the problem and highlights what’s wrong, the user needs to fill or correct the selected items.  If the user provides a wrong email address, he/she must redo the sign-up process from the beginning. |



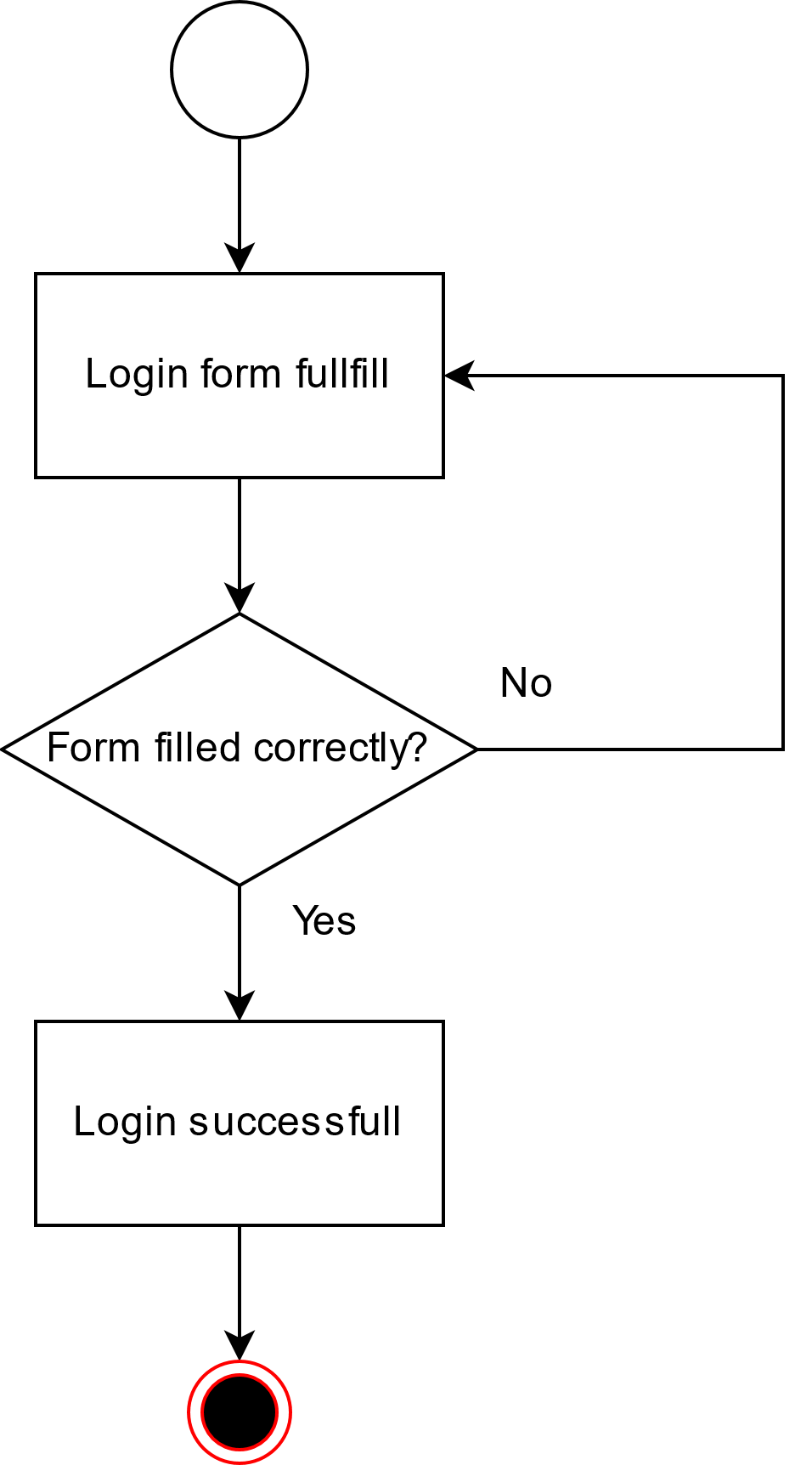
Activity Diagram UC[1]

**UC[2] Login:**



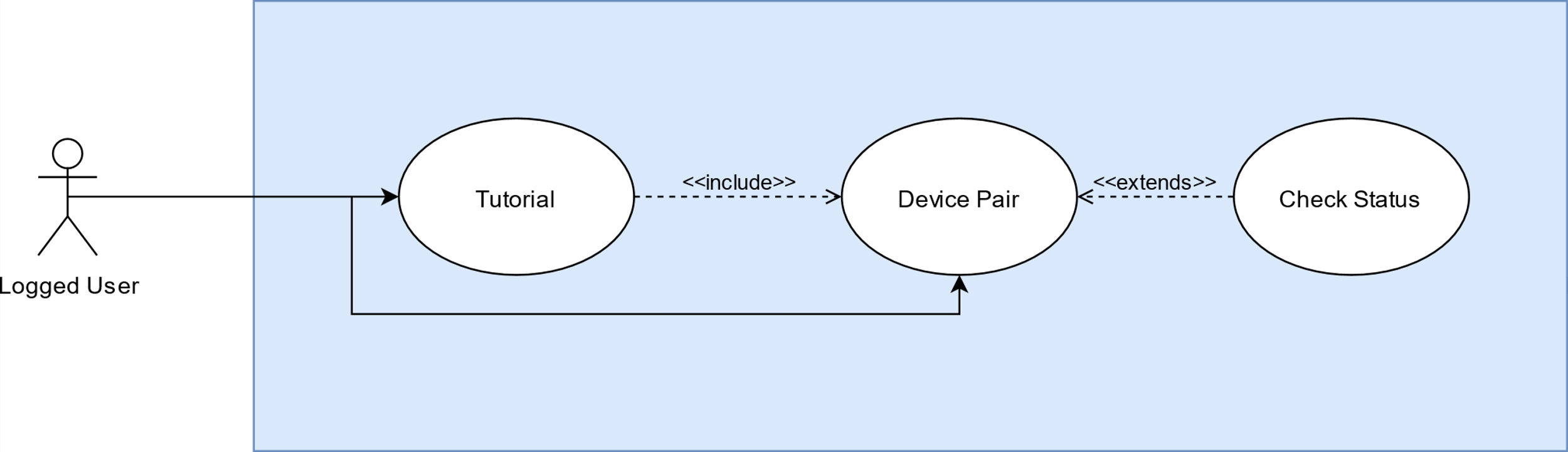
UC[2] Login

|  |  |
| --- | --- |
| Name | Login |
| Actor | User |
| Entry Condition | The actor must be registered |
| Goal |  |
| Event Flow | 1. The user fills up the login form with their credentials 2. The system checks the credential given |
| Exit condition | The user is logged in and he ca use the functionalities of the application. We will call **Logged User** the user that has performed this use case. |
| Exceptions | 1. Some form fields get incorrectly filled or left empty, the system notifies the user on the problem and highlights what’s wrong, the user needs to fill or correct the selected items. 2. Some credentials provided are wrong, the user will be alerted with a “wrong credentials” message. |



Activity Diagram UC[2]

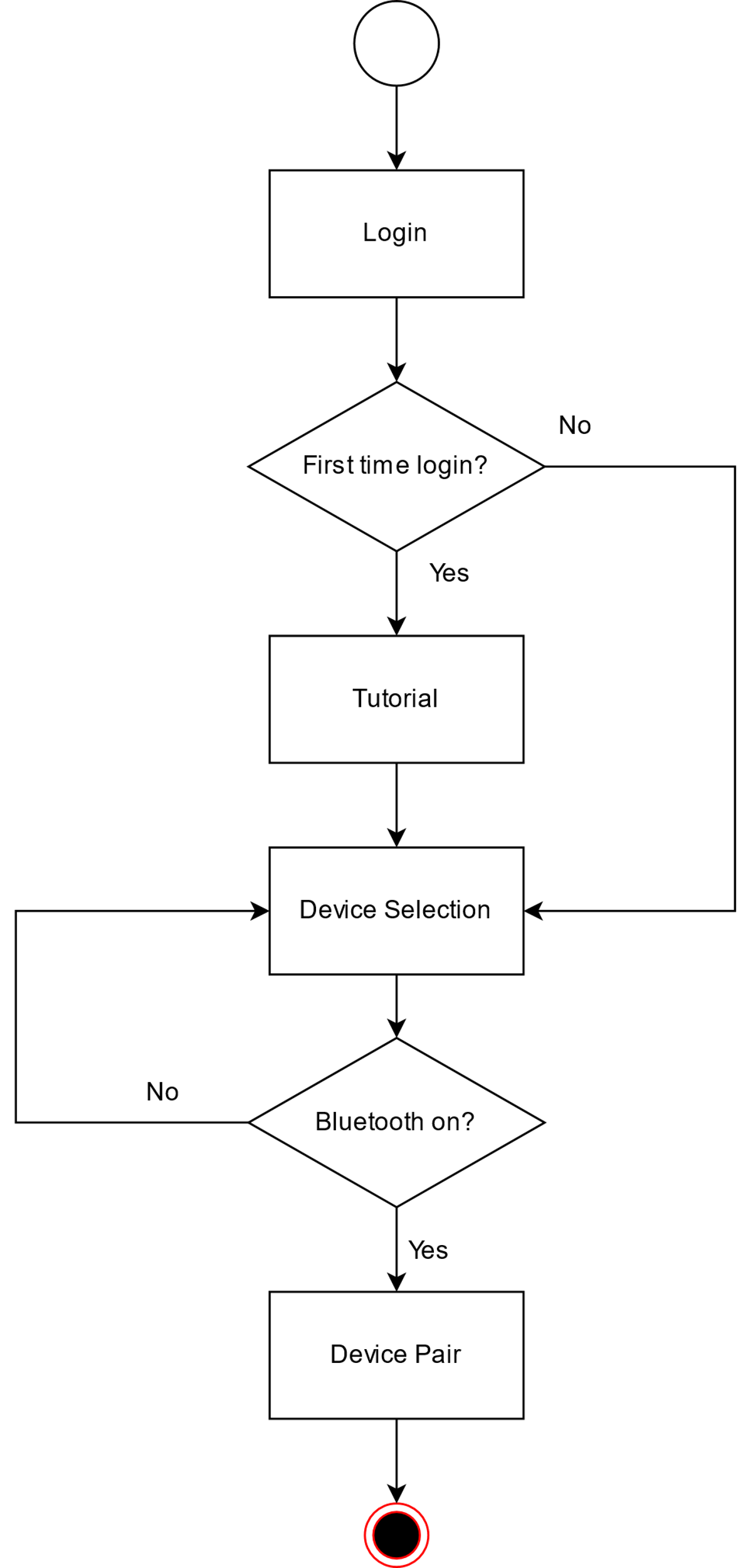
**UC[3] Device Pair:**



UC[3] Device Pair

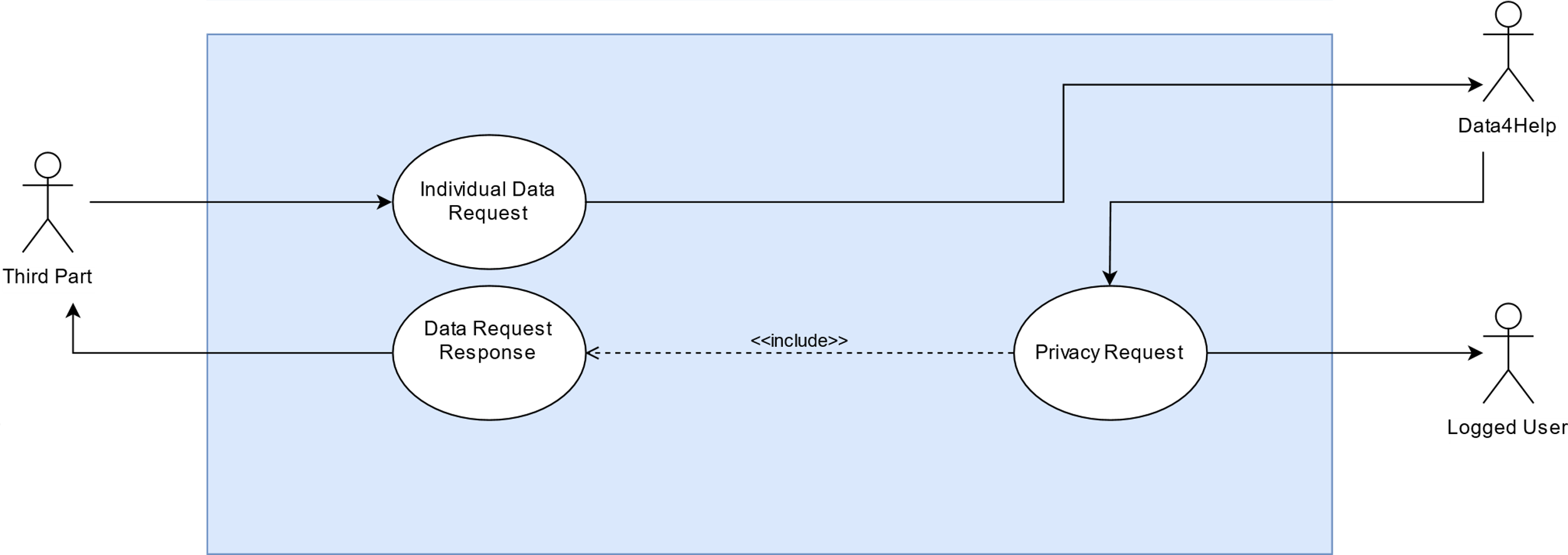
|  |  |
| --- | --- |
| Name | Tutorial |
| Actor | Logged User |
| Entry Condition | The actor must be logged in for the first time |
| Goal |  |
| Event Flow | 1. The user gets prompted with information on the app and functionalities along with a description of the menu 2. The user gets guided through the pair device process described in the next table |
| Exit condition | The user gets guided until the end of pairing process |
| Exceptions | 1. App crashes 2. The user doesn’t complete the tutorial process   In both the exceptions the system provides the tutorial process again |

|  |  |
| --- | --- |
| Name | Device Pair |
| Actor | Logged User |
| Entry Condition | The actor must be logged in |
| Goal |  |
| Event Flow | 1. The user selects his own device from the drop-down menu (see Fig.3.1.9) 2. The user clicks the “Pair” button (see Fig.3.1.9) |
| Exit condition | The user connects successfully a new device with the application |
| Exceptions | 1. Bluetooth turned off, the user must activate it and open again the drop-down menu |



Activity Diagram UC[3]

**UC[4] Third-party individual request:**

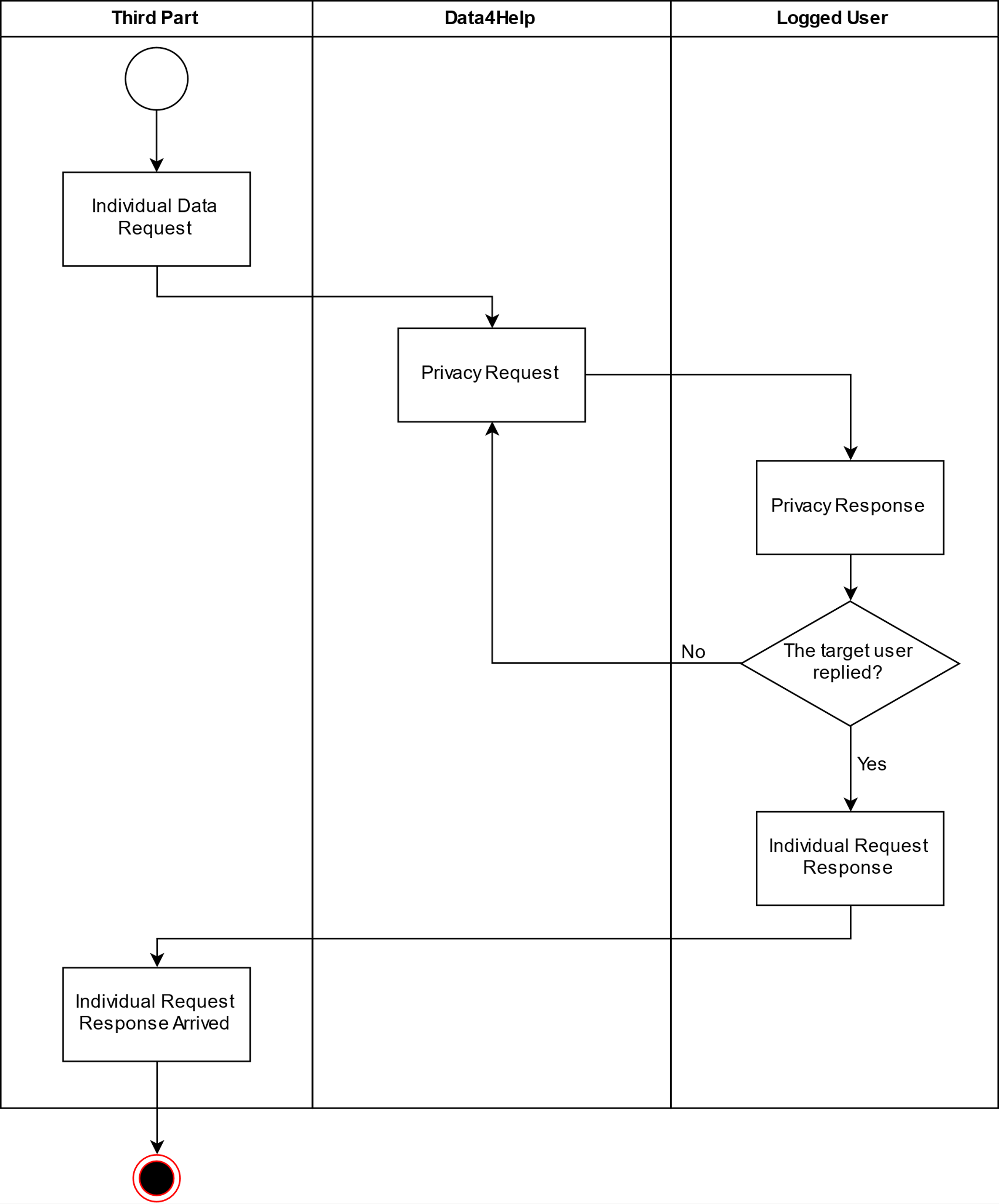


UC[4] Third-party individual request

|  |  |
| --- | --- |
| Name | Individual Data Request |
| Actor | Third Part |
| Entry Condition | The third part must have been added into the system |
| Goal |  |
| Event Flow | 1. The third part sends a request 2. The third part waits a response |
| Exit condition | The third part sends a request and the system receives it |
| Exceptions | 1. Data4Help isn’t reachable 2. Third part system crashed   In both exceptions the third part continues to send the same request until it gets a response |

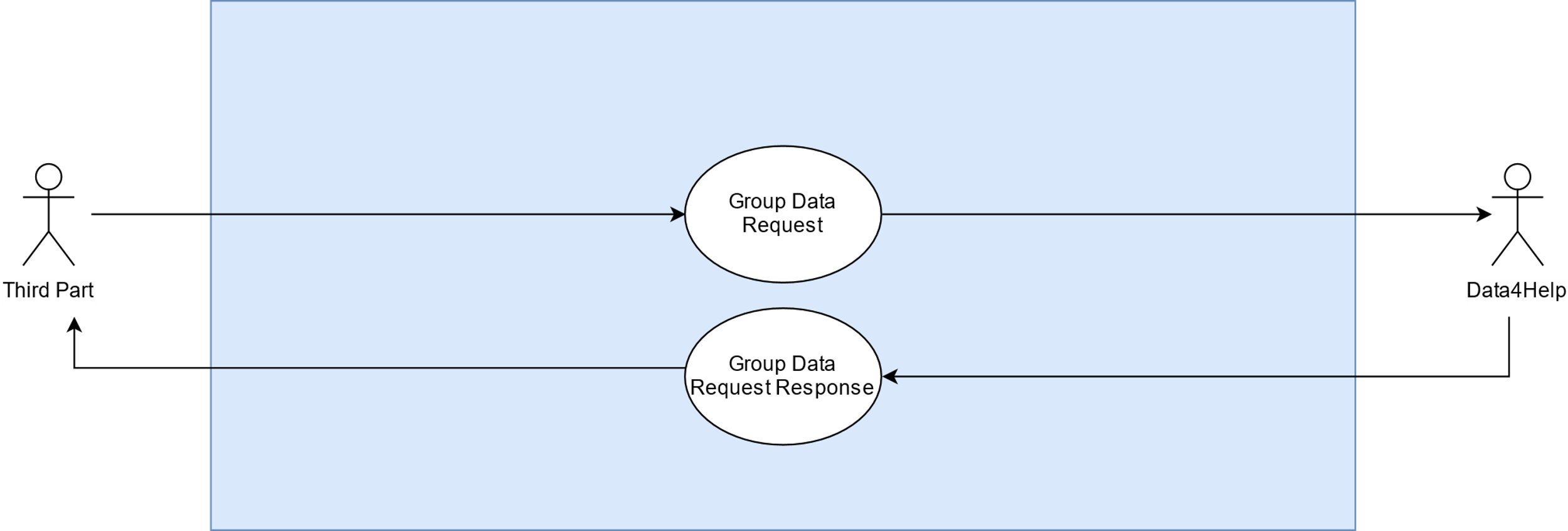
|  |  |
| --- | --- |
| Name | Privacy Request |
| Actor | Data4Help |
| Entry Condition | A data request arrives to the system |
| Goal |  |
| Event Flow | 1. The system sends a Privacy request to the target user 2. The target user accepts or denies the request |
| Exit condition | The user accepts or denies the request |
| Exceptions | 1. The user ignores the request, the system continues to notify the user. After 5 attempts the system considers the request denied |

|  |  |
| --- | --- |
| Name | Data Request Response |
| Actor | Data4Help |
| Entry Condition | The user replied to the privacy request |
| Goal |  |
| Event Flow | 1. The system replies to the third part according the target response |
| Exit condition | The third part gets a response |
| Exceptions | 1. The system crashed 2. Third-party system isn’t reachable   In both exceptions the system continues to send the same response until it arrives |



Activity Diagram UC[4]

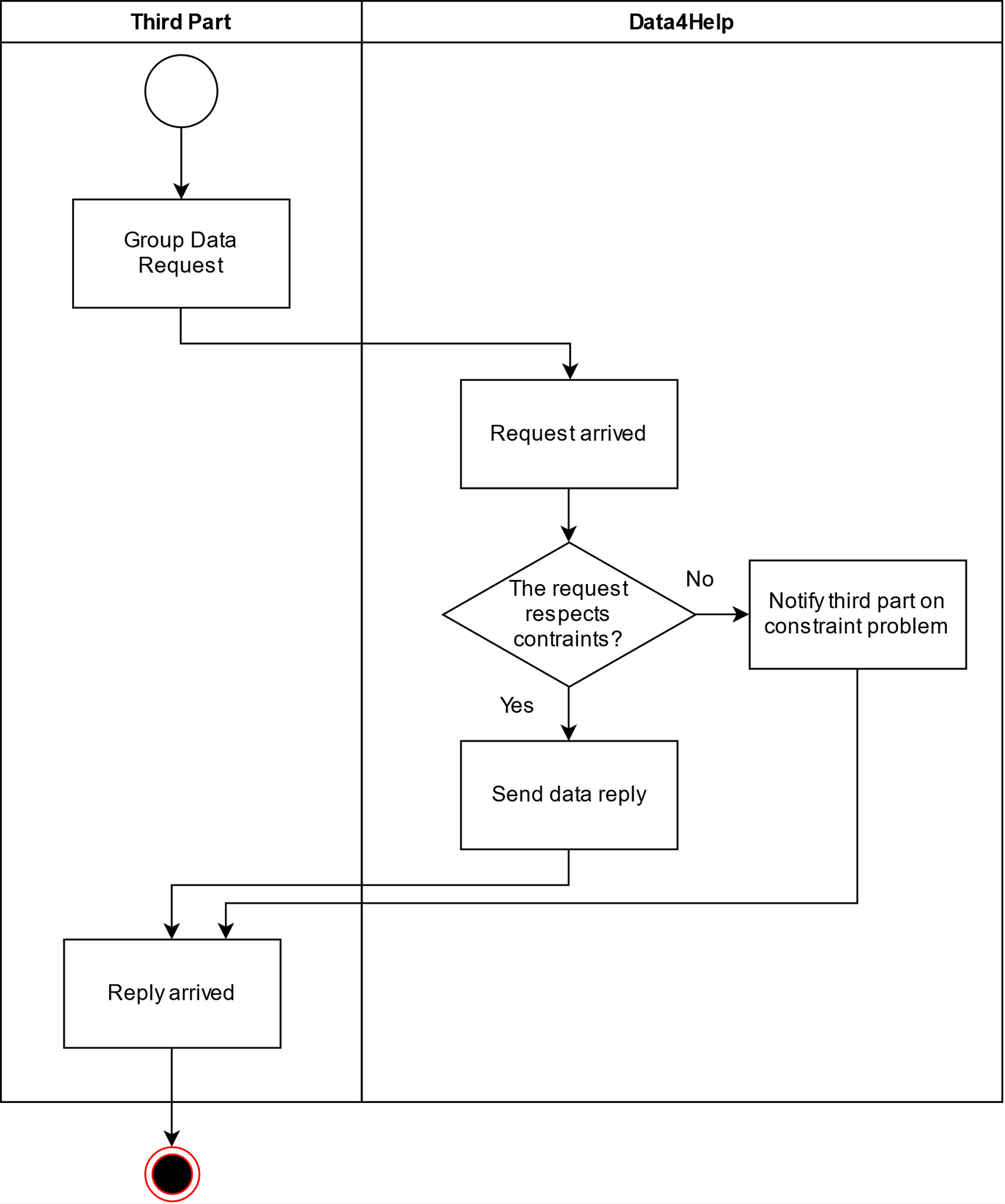
**UC[5] Third-party group request:**



UC[5] Third-party group request

|  |  |
| --- | --- |
| Name | Group Data Request Response |
| Actor | Data4Help |
| Entry Condition | A data request arrives to the system |
| Goal |  |
| Event Flow | 1. The system replies to the third part respecting the cardinality constraint (See constraint 5, section 2.4) |
| Exit condition | The third part gets a response |
| Exceptions | 1. The system crashed 2. Third-party system isn’t reachable   In both exceptions the system continues to send the same response until it arrives |

|  |  |
| --- | --- |
| Name | Group Data Request |
| Actor | Third Part |
| Entry Condition | The third part must have been added into the system |
| Goal |  |
| Event Flow | 1. The third part sends a request 2. The third part waits a response |
| Exit condition | The third part sends a request and the system receives it |
| Exceptions | 1. Data4Help isn’t reachable 2. Third part system crashed   In both exceptions the third part continues to send the same request until it gets a response |



Activity Diagram UC[5]