Politecnico di Milano

Master's Degree in
Computer Science and Engineering

Software Engineering 2

TrackMe
Design Document

Authors
Alberto Archetti
Fabio Carminati

Reference professor
Elisabetta Di Nitto

v.0 - November 27, 2018

Contents

1 Introduction

1.1 Purpose

TrackMe wants to develop a software-based service that allows individual users to collect health data, called `Data4Help`. This data can be retrived from the system and visualized according to different filters by a user interface.
The system allows third parties registration. Third parties can request access to users'collected data in two ways:

Single user data After a third party makes a request to the system for a single user data sharing, by providing user's fiscal code, the system asks the user for authorization; if positively provided, the third party is granted access to the user's data

Amonymous group data Third parties can be interested in big amounts of data, but not in who are the people providing it; the system, once the request is sent by the third party, checks if the data can be effectively anonymized (it must find at least 1000 people that can provide data matching the third party request's filters) and, if positively evaluated, grants access to the anonymized data to the third party

Third parties can subscribe to new data and receive it as soon as it is collected by the system.

Another service that TrackMe wants to develop is `AutomatedSOS`, built on `Data4Help`. This service analyzes users'data and calls a SOS whenever data exceedes the basic health parameters. For this particular purpose, system performance will be a critical aspect to be taken into account, because even the slightest delay matters in critical health situations.

We will list the project **goals**, described in the RASD document:

`G.U1` Users can collect, store and manage their health data

`G.U2` Users can choose to have their health monitored; if their health is critical, an ambulance will be dispatched

`G.T1` Third parties can ask single users for their health data sharing

`G.T2` Third parties can request access to anonymized data that comes from groups of people

`G.T3` Third parties can subscribe to new data and receive it as soon as it is produced

1.2 Scope

1.2.1 World

Our *world* is composed of two main types of actors: *users* and *third parties*. Users are interested in monitoring their health parameters and third parties are interested in developing services or researches that exploit data gathered from the users. `Data4Help` is the service that acts as a bridge between these actors'needs.
Phenomena that occur in the *world* and are related to our application domain are

- physical conditions of the users
- third parties'projects, researches and interests
- ambulances dispatched by the SOS system

These phenomena exist in the *world*, but cannot be observerd directly by our system.

1.2.2 Shared phenomena

In order to communicate with the *world*, our system needs to share some aspects with it. We will list the aspects controlled by the world, but observable by the machine:

`S.1` physical parameters of the users, gathered through sensors on wereable devices

`S.2` third parties requests to the system for the data they need

`S.3` users'location, acquired through GPS signals

On the other hand, the aspects that occur in the machine, but are observable by the world are

`S.4` interfaces that organize the gathered data that can be filtered according to time or type of data

`S.5` messages for the SOS system, that are sent in case of critical health of a user

`S.6` payment requests

1.3 Definitions

Data Quantitative variables concerning health parameters

- **Aggregate data** See *DataSet*
- **Anonymous data** *data entry* that doesn't contain information about the user from which it was produced; a *data set* is said to be anonymized if it contains only anonymous *data entries* and its cardinality is greater or equal than 1000
- **Data entry** Tuple that corresponds to the user's parameters in a particular moment
- **Data set** Set of *data entries*; depending on the context, it can identify a set of entries all belonging to a single user or or a set of anonymous entries belonging to more that 1000 users; a *data set*, among all *data* that the system is storing, can be identified and constructed according to the filters of a third party request

Request Third parties can ask the system for some data sharing through requests; requests are encoded through filling a form; the system, provided that the request is satisfiable, grants the third party access to the requested data

Third party Actor interested in collecting data from a single user or from an anonymous group of users

Threshold Numerical values related to a particular health parameter; they act as boundaries between the domain of critical health status and normal health status

User Actor interested in his/her health data collecting and managing; a user can also be interested in automating SOS calls whenever his health status becomes critical

Some of these definitions may already be present and further explained in the RASD document.

1.4 Acronyms and abbreviations

API Application Programming Interface

DBMS Database Management System

Data Whenever the context refers to generic groups of *data entries*, the terms *data* and *data set* are interchangeable

System Software product that TrackMe wants to develop; can be interchanged with *S2B*

S2B Software To Be

1.5 Revision history

Version	Log
`v.0`	DD first draft

1.6 Document structure

This document describes architecture and design of `Data4Help` and `AutomatedSOS` systems. The description will start with a top-down approach, in order to make the reader familiar with the overall structure; a bottom-up approach will then be adopted, in order to describe components in a isolated way. This document is devided in

- Section 1 is a brief introduction on the project to be developed in order to make this document self-contained
- Section 2 describes the high-level architecture (high-level components, their interation, runtime views and architextural decisions)
- Section 3 provides an overview on how the user interface will look like
- Section 4 contains mapping between software requirements, described in the RASD document, and design elements
- Section 5 identifies the order in which subcomponents will be implemented, integrated and tested
- Section 6 lists the work sessions that drove this document's development, ordered by date, as the hour counter of effort spent by each group member

2 Architectural Design

2.1 Overview

The architecture is a three-tier architecture (Figure 1): it allows to separate clearly *presentation* layer, *business* layer and *data* layer. These sets of components will communicate through defined interfaces and will be treated as black boxes during their interaction. This modular approach enhances modifiability and extensibility.

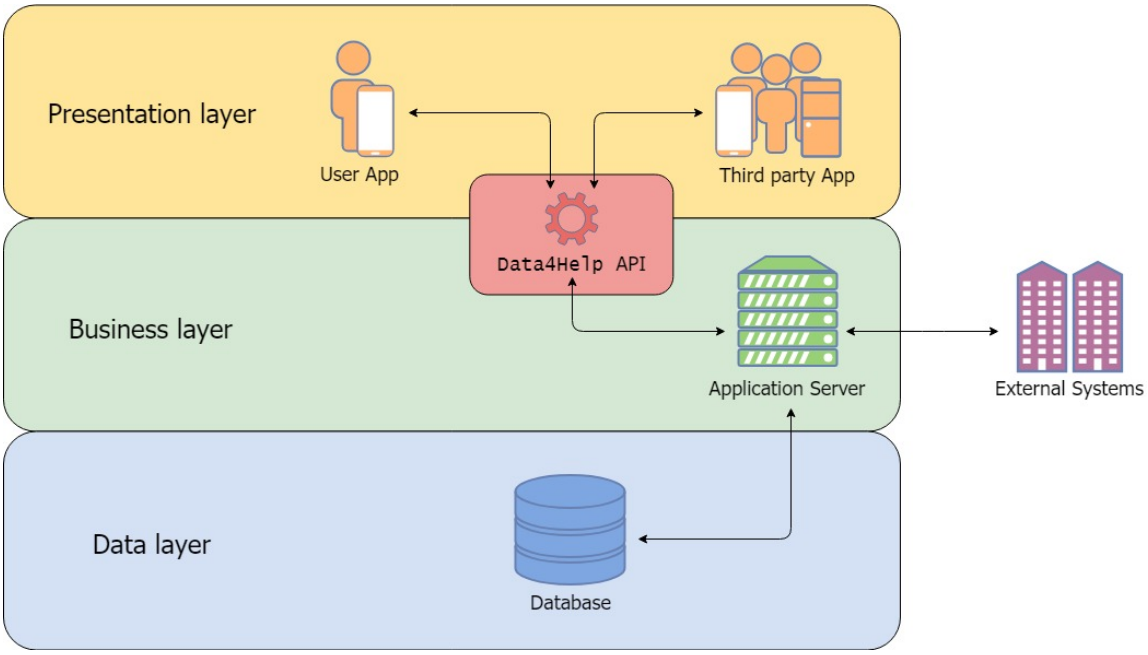

Figure 1: Overall architecture of the system

The main components of the system are

App Application installed on users'devices that communicates with the system; its purpose is to show data to the user and forward his/her requests to the Application Server; we will focus on the smartphone app for Andriod or iOS systems, as it is the main front-end application that our clients need

Application Server Back-end component on which the logic of the application takes place; it elaborates the requests it receives and interacts with external services and the data layer; we will focus mainly on this component, as it shall handle all the information dispatching from different layers

Database Component responsible for data storage; it shall grant ACID properties (Atomicity, Consistency, Isolation and Durability) and shall provide a management service that handles query parallelization and optimization, as data access policies from different accounts

External Systems Systems that interact with `Data4Help` or `AutomatedSOS`; they handle functionalities not internally developed in the system, such as payment handling and ambulance dispatching

2.2 Component view

In this section we will analyze every high-level component in terms of its subcomponents and provide the main interface interaction between different components. For details on component interfaces see Section 2.5.

2.2.1 App

The application component is the front-end of the system. Our clients will interact with the system through the front end. We will provide

- A smartphone application, capable of exploiting all of the system functionalities: it shall render data, provide forms for the clients (users and third parties) and communicate with the Application Server
- An API that allows more experienced users or other developers to automate communication with our system; the API is particularly useful when third parties need to analyze huge quantities of data that a smartphone graphical interface cannot render

It is important to note that the smartphone application exploits the API for communication with the Application Server. Every `Data4Help` or `AutomatedSOS` service can be required by API communication.

2.2.2 Application Server

The Application Server holds the application logic. It is the only component of the *business* layer, but it is the most crucial component of the system. Its role is to coordinate the information flow between the user layer and the data layer and to incorporate external systems'services.

In the architecture the Application Server is the only link to the database. External systems or clients cannot directly access persistent data of our system.

The Application Server is also the only link to the *presentation* layer, as the Application Server coordinates the user-external system interation.

Subcomponents of the Application Server are

`AccountManager` This module handles creation, authentication and management of users and third parties'accounts; before exploiting our system's functionalities users and third parties need to be authenticated by this module after providing their credentials

`DataCollector` This module communicates with users'application and periodically receives data entries, as soon as they're collected by users'wearables

`EmergencyDetector` This module is in charge of automatically analyze data entries inserted in the system if their owner subscribed to `AutomatedSOS`; it is separated from the `DataCollector` because emergency detection can be exploited in many ways, depending on the medical literature on the topic; this feature should be independant and isolated from the rest of the architecture

`EmergencyDispatcher` This module builds emergency messages and forwards them tothe SOS system

`FilterManager` This module composes filter constraints on data entries that can be fetched from the database

`NotificationManager` This module shall dispatch notifications between user and third party accounts; notifications from users to third parties contain information about user's responses concerning third parties'requests; notifications from third parties to users corcern third parties'single user requests

`PaymentGateway` This module is in charge of communicating with the external payment system is order to process payments between third parties and TrackMe

`RequestManager` This module is in charge of composing, verifying and elaborating third parties'requests, both of single-user type and anonymous-group type; it communicates with `FilterManager` to properly identify which type of data is required and with `NotificationManager` to keep every client involved in the request updated on its status

`SetBuilder` This module generates data-oriented queries for the database, given a particular filter from the `FilterManager`; queries can be accepted or declined by the database, depending on the account permissions concerning data entries access

2.2.3 Database

The database is the only component of the *data* layer. Queries are managed by a DBMS that optimizes and elaborates them in parallel. Data stored in the database is persistent and shall not be lost due to external factors. The database service will not be directly developed by us, but will be bought from the existing ones.

The *data* layer is only accessible from the Application Server. It won't implement any application logic, except from DBMS functionalities: it will just respond to queries and passively store data.

An important factor for `Data4Help` is the data access policy: Data Entries should be available only to the users that produced them, when inserted in the database. If a Data Set is shared to a third party, that third party shall be allowed to retrive Data Entries that belong to that Data Set from the database. Therefore the access policy shall be dinamic and shall consider `Data4Help` different accounts.

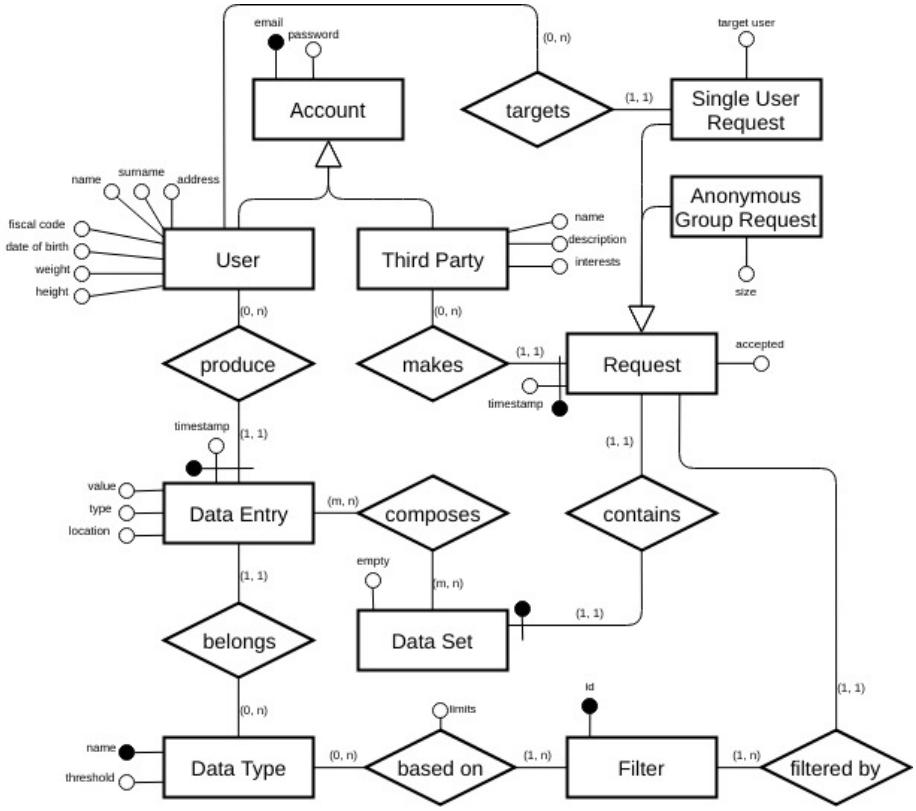

Figure 2: Entity-Relation diagram

2.2.4 External Systems

In this section we will present the main external systems that interact with the Application Server.

`Data4Help` relies on an external payment handler. The Application Server, once has composed a third party request, evaluates its price and asks third party for payment, by exploiting the external payment handler service. The service manages the effective payment from the third party to TrackMe and signals errors occurred during the procedure.

`AutomatedSOS` relies on an external SOS system. The SOS system dispatches ambulances and handles health emergencies by accepting automated calls. `AutomatedSOS`, on the Application Server, detects health dangers as soon as they're collected from the front-end components forwards an emergency message to the SOS system.

2.3 Deployment view

2.4 Runtime view

2.4.1 User Account

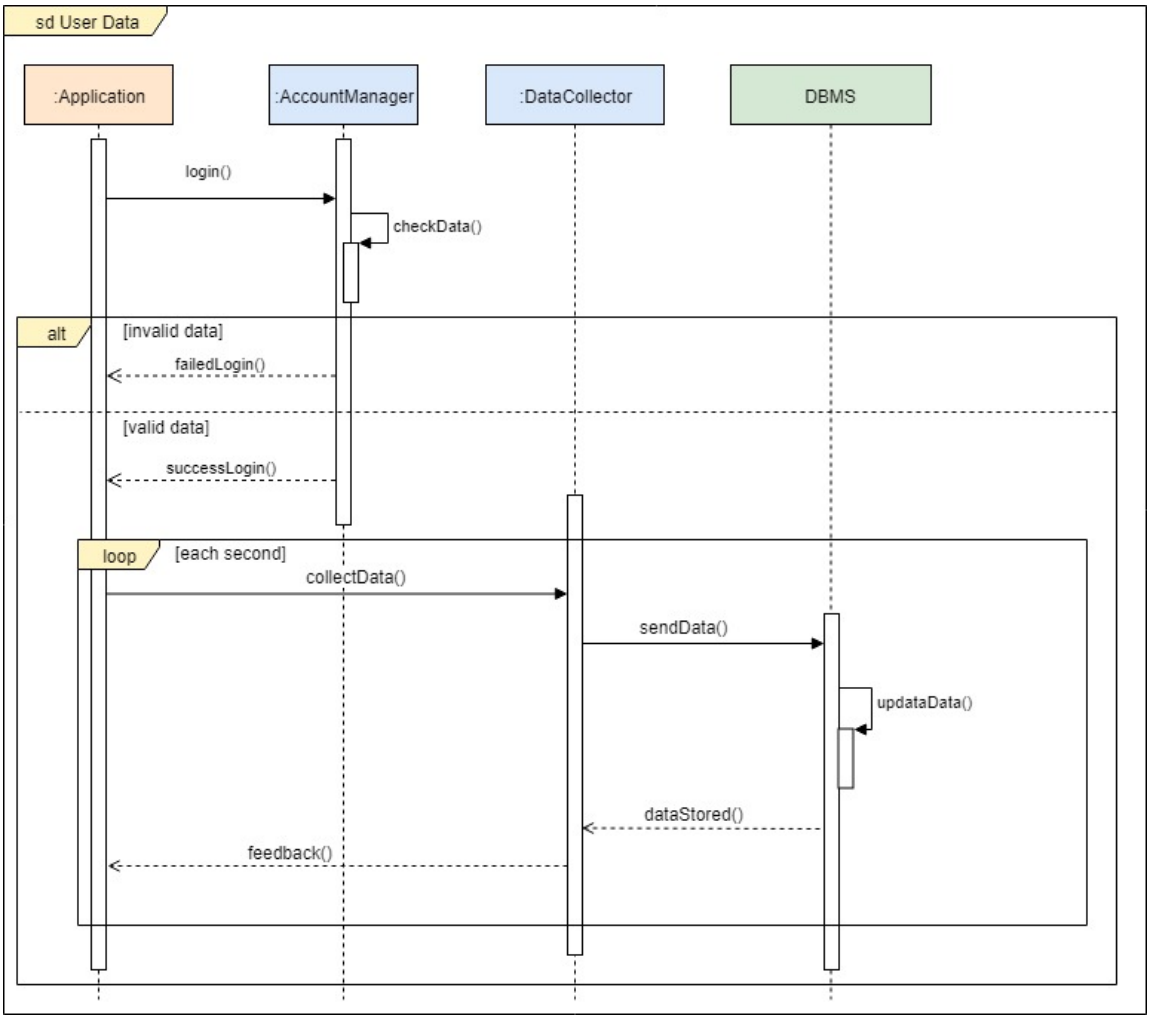

Figure 3: User account

2.4.2 Filtering

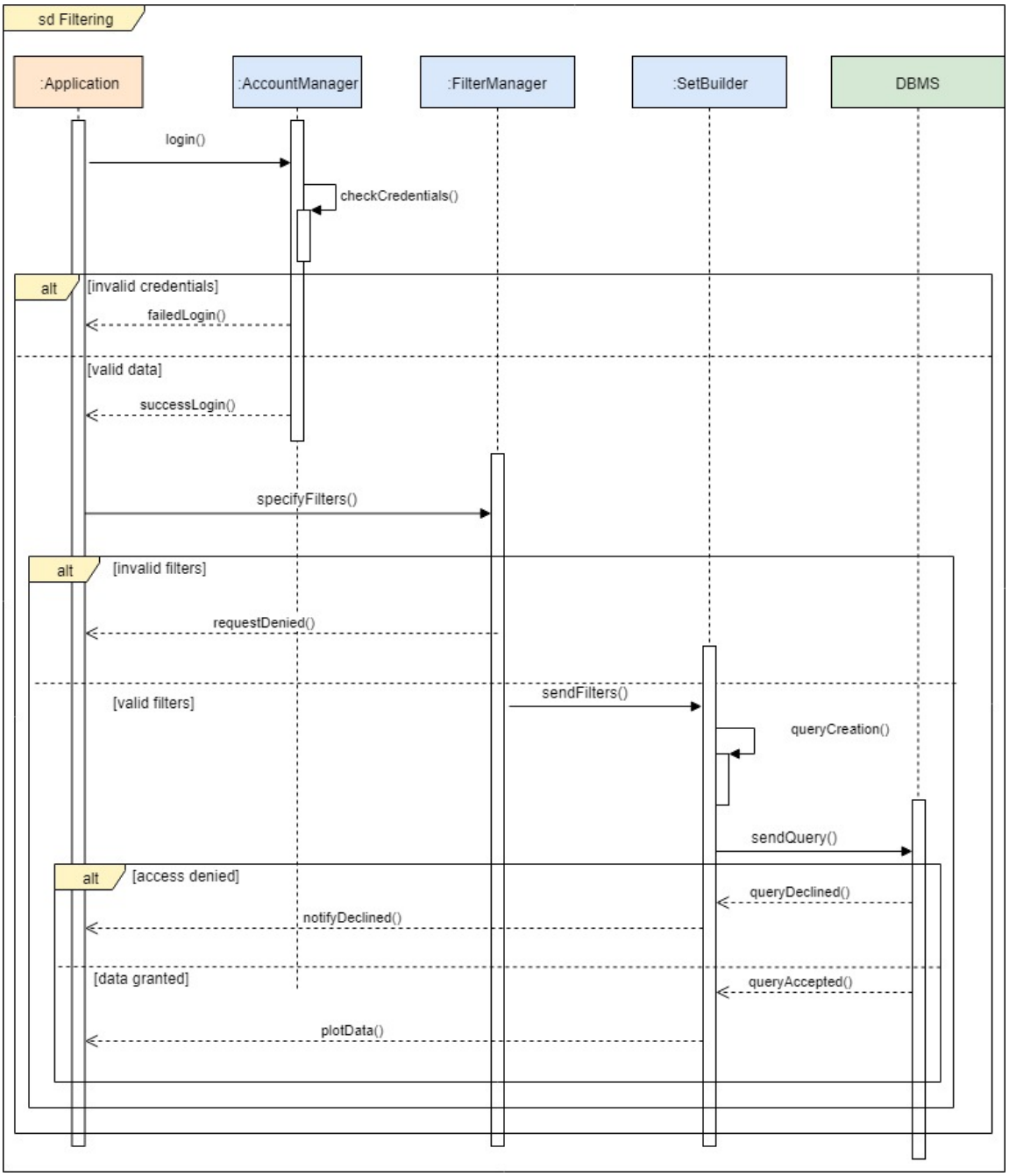

Figure 4: Filtering

2.4.3 AutomatedSOS

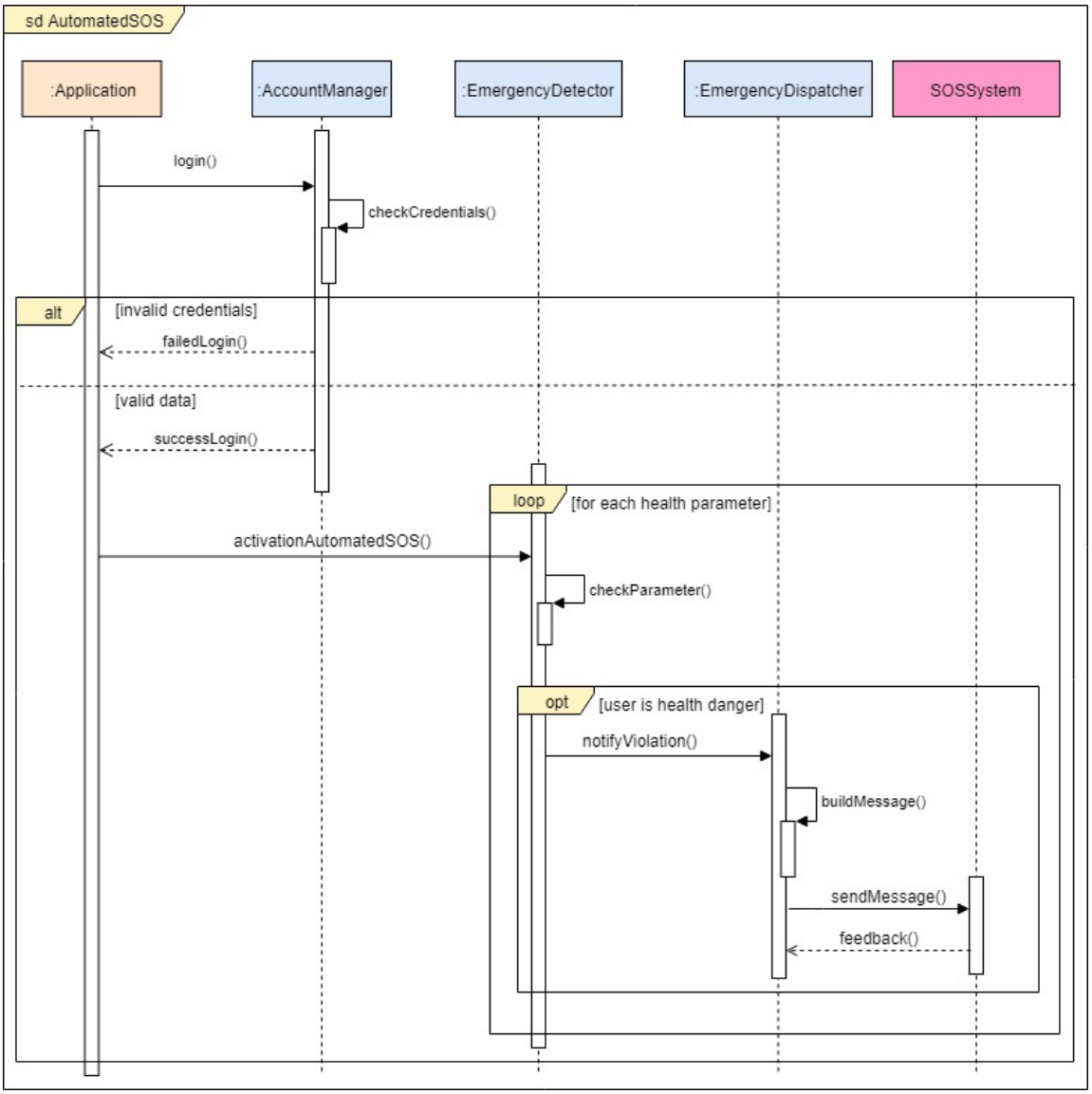

Figure 5: AutomatedSOS

2.4.4 Single User request

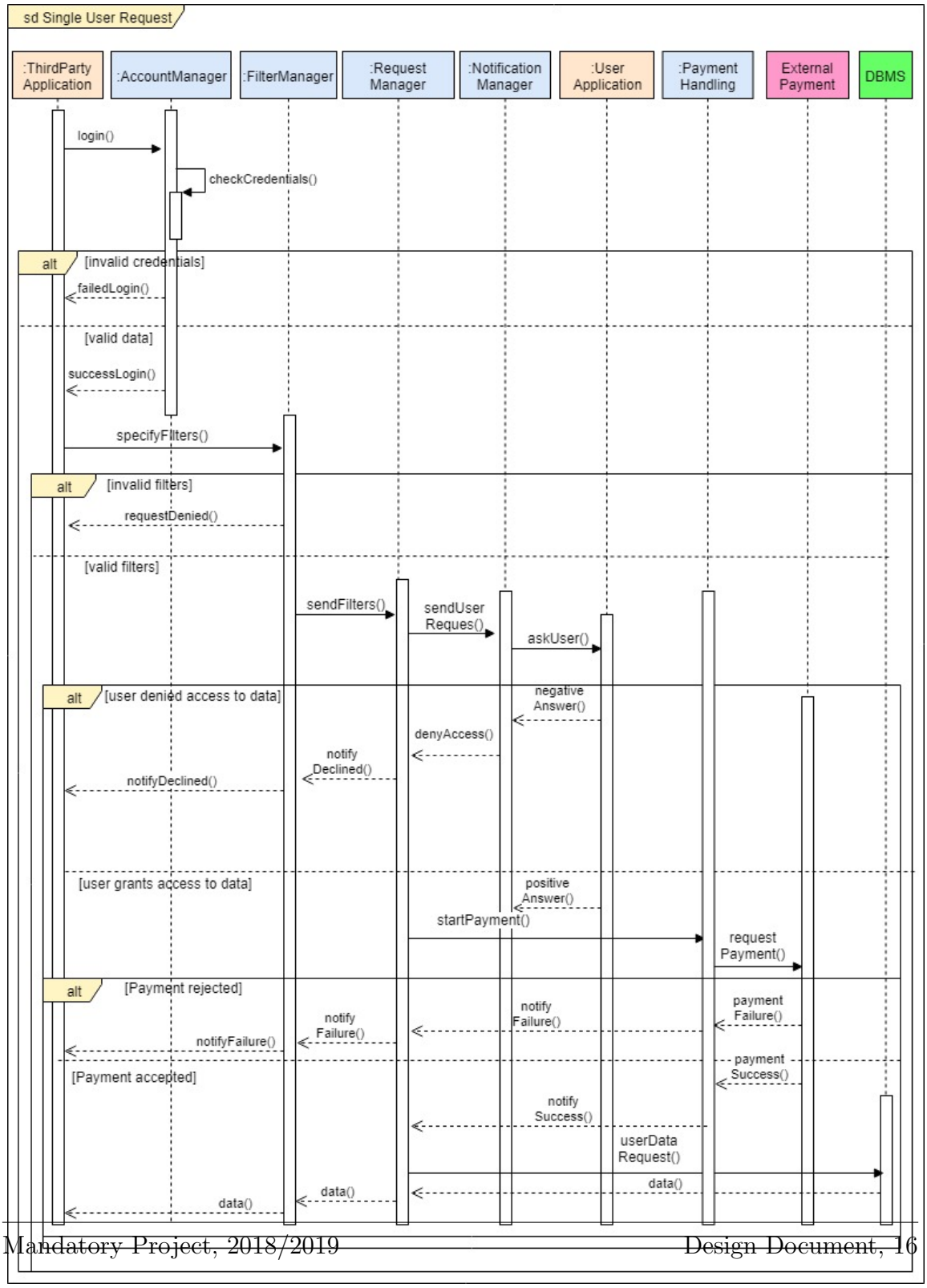

Figure 6: Single User request

2.4.5 Anonymous Group request

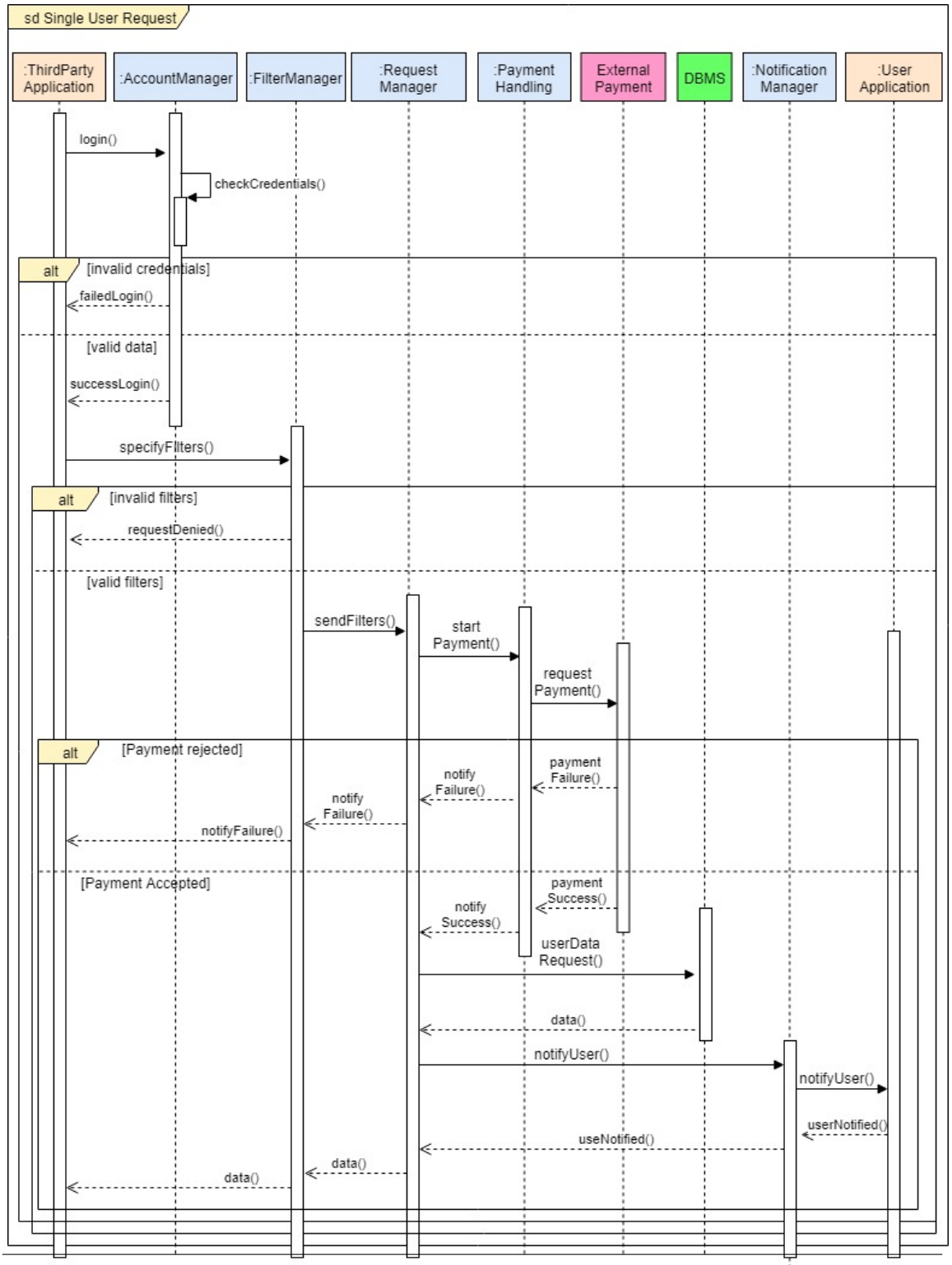

Figure 7: Anonymous Group request

2.5 Component interfaces

In this section we will present the details concerning the interfaces of the components defined in Section 2.2. Every component will offer or require some functionalities through interface methods in a way that, once all components are assembled, the system shall not have uncovered functionalities and shall satisfy RASD requirements through offered functionalities.

We will present first the interfaces of the Application Server subcomponents.

2.5.1 `AccountManager`

`createUserAccount` Generates a new user account, provided email, password and other valid information, specified in Section 2.2.1 of RASD document; the return value specifies if the procedure ended correctly or if some incorrect information made it abort

`createThirdPartyAccount` Generates a new third party account, provided email, password and other valid information, specified in Section 2.2.1 of RASD document; the return value specifies if the procedure ended correctly or if some incorrect information made it abort

`login` By providing email and password, a client can login into his/her account and exploit system functionalities[1]; the return value is positive if information provided is correct and negative if there's no account that matches given credentials

`resetPassword` This method is called when a client doesn't remember his/her password: it triggers a reset procedure through the client's email, used for registering to `Data4Help`

`editInfo` Updates the client's profile information with the the new set of information passed as parameter; the return value confirms if the procedure ended correctly

`getSharingPermissions` This method returns the set of data types that the target user is willing to share anonymously

[1] the *login* procedure is exploited by assigning a unique token to the client session and providing it as a login succesfull certificate during every API call to the Application Server

2.5.2 `DataCollector`

`acquireNewDataEntry` This **asynchronous** method acquires a new data entry collected on the logged user's application; if the logged user is subscribed to `AutomatedSOS`, it forwards the data entry to the `EmergencyDetector`

2.5.3 `EmergencyDetector`

`analyzeDataEntry` This **asynchronous** method analyzes passed data entry and checks whether all parameters are above or below defined thresholds; if some parameters exceed thresholds, it will call

`generateEmergencyMessage` This method, given a data entry that exceedes thresholds, returns an emergency message that must be delivered to the SOS system

2.5.4 `EmergencyDispatcher`

`sendEmergencyMessage` This method triggers the passed emergecy message to be sent to the SOS system

2.5.5 `FilterManager`

`generateFilter` This method returns a new filter instance with the passed parameters on type and boundaries of data, that can be used to narrow the data domain of interrogation during database queries

2.5.6 `NotificationManager`

`notifyDecline` This method notifies a third party that its single user request has been declined by the target user

`notifyUnfeasibility` This method notifies a third party that its anonymous group request hasn't been accepted by the system due to the impossibility of properly anonymized required data

`notifyPaymentResult` This method sends to a third party the exit status of the payment operation it started

2.5.7 `PaymentGateway`

`pay` This method triggers a payment call to the external payment system that returns a positive or negative exit status, depending on the correct execution of the procedure

2.5.8 `RequestManager`

2.5.9 `SetBuilder`

`getDataSet` This method accepts a filter as parameter and forwards a query based on such filter to the database; the return value is either the set of data entries fetched from the database subject to the filter's constraints or an error message if the query couldn't be performed (the asking user hasn't access permissions or there's not anough data to satisfy the query requirements)

2.5.10 Database

2.5.11 External Systems

2.6 Selected architectural sytles and patterns

2.7 Other design decisions

3 User Interface Design

3.1 Flow graph for Screens Interface

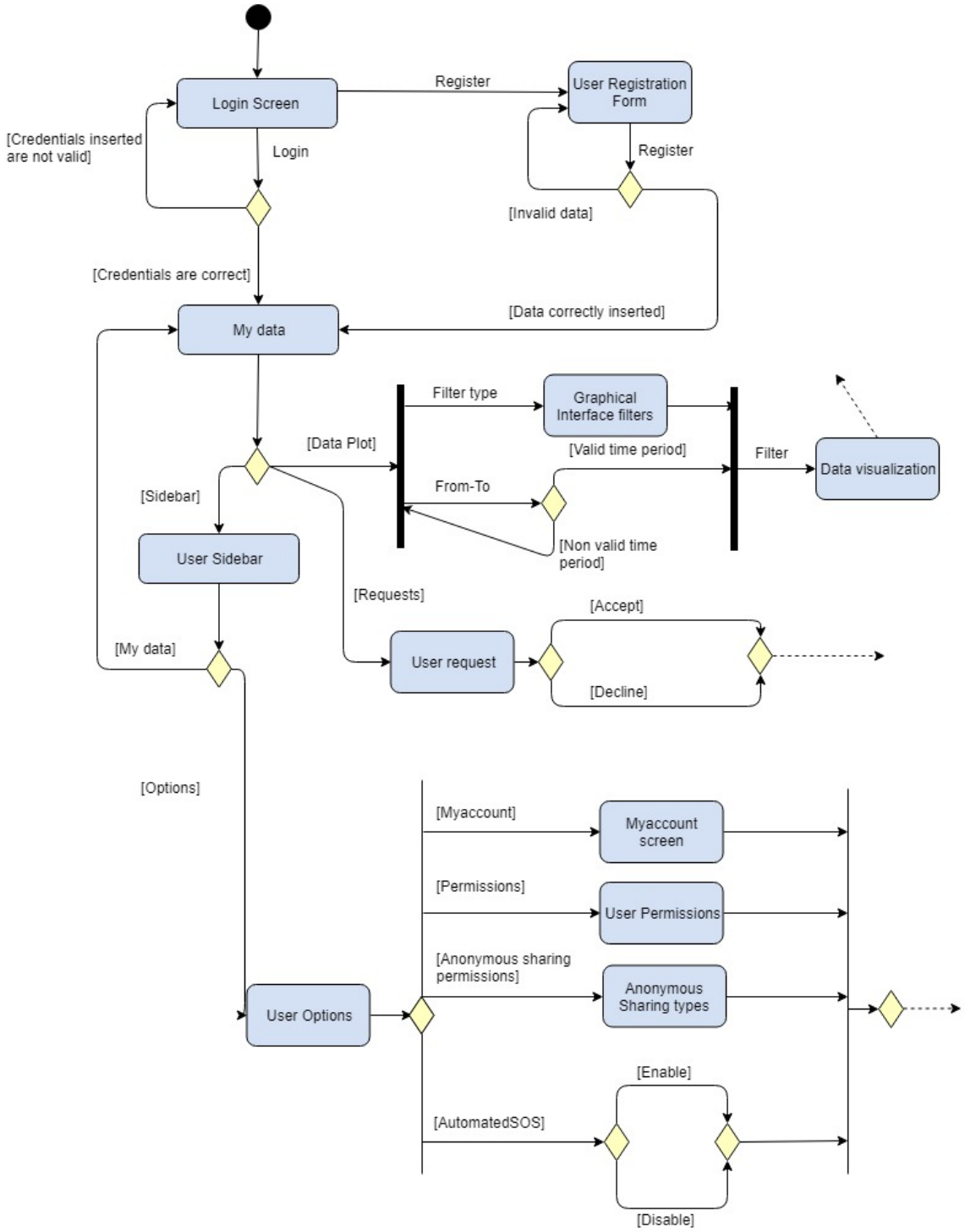

Figure 8: User account

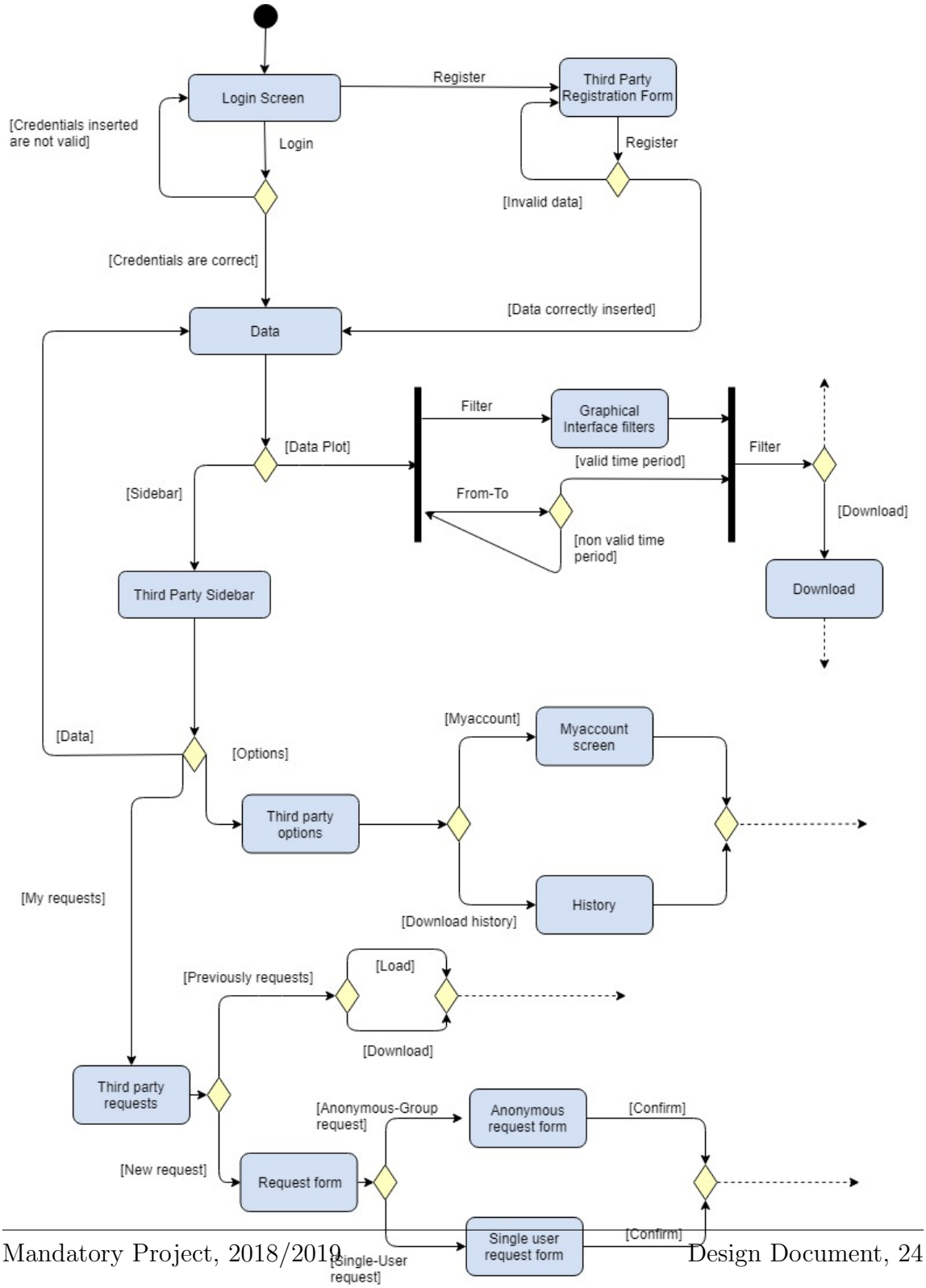

Figure 9: Third Party account

4 Requirements Traceability

In the following tables a description of components which interact with presentation layer is given[2].

4.1 Guest User

Reference ID	Requirement	Component
R.A1	Registration as Data4Help user	Account Manager
R.A3	Distinguish user and third party accounts	Account Manager
R.A4	Different accounts must have different email	Account Manager
R.I1	Provide registration form	Account Manager

4.2 Guest Third Party

Raw ID	Requirement	Component
R.A2	Registration as Data4Help third party	Account Manager
R.A3	Distinguish user and third party account	Account Manager
R.A4	Different accounts have different email	Account Manager
R.I1	Provide registration form	Account Manager

[2]In this section for readibility purpose a brief description of functional requirements is given but for full description see RASD document section 3.2

4.3 Data4Help User

Raw ID	Requirement	Component
R.A3	Distinguish user and third party account	Account Manager
R.A5	Login	Account Manager
R.A6	Only authenticated user can exploit all functionalities	Account Manager
R.D1	System shall encode and store data	Database
R.D2	Retrieve data in case of request	Filter Manager
R.D3	Not erase stored data	Database
R.D4	Data can be collected only when the users are logged in	Account Manager
R.I2	Provide graphical interface to render data graphically	Filter Manager
R.R2	Ask for single-user request directly to user	Notification Manager

4.4 AutomatedSOS User

4.5 Data4Help Third Party

Raw ID	Requirement	Component
R.A3	Distinguish user and third party account	Account Manager
R.A5	Login	Account Manager
R.A6	Only authenticated user can exploit all functionalities	Account Manager
R.S1	Apply to AutomatedSOS	Emergency Detector
R.S2	Monitoring user's health	Emergency Detector
R.S3	In case of need make an emergency call to SOSSystem	Emergency Dispatcher
R.S4	Provide user's location and health parameters to SOSSyestem	Emergency Dispatcher

Raw ID	Requirement	Component
R.A3	Distinguish user and third party account	Account Manager
R.A5	Login	Account Manager
R.A6	Only authenticated user can exploit all functionalities	Account Manager
R.D1	System shall encode and store data	Database
R.D2	Retrieve data in case of request	Filter Manager
R.D3	Not erase stored data	Database
R.D5	Share data among multiple accounts	SetBuilder
R.R1	Allow to make single-user request	Request Manager
R.R3	Decide whether to share or not the data according to user's answer in case of single-user request	Request Manager
R.R4	Allow to make anonymous-group request	Request Manager
R.R5	Check whether the required data can be properly anonymized in case of anonymous request or not	Request Manager
R.R6	Provide data only if can be properly anonymized in case of anonymous request	Request Manager
R.R7	Provide access to newly produced data if they fit the request	Request Manager
R.I2	Provide graphical interface to render data graphically	Filter Manager
R.I3	Provide a request form	Request Manager

5 Implementation, Integration and Test Plan

6 Effort spent

Date	Archetti Alberto	Carminati Fabio	Activity
12/11/2018	1	1	Introduction sketch
24/11/2018		6	User Interface Design
24/11/2018	3		High-level components
25/11/2018	2		Application Server sub-components
26/11/2018		5	Architectural Design
27/11/2018	2		Component interfaces
27/11/2018		3	Requirement Traceability

References

[1] Mandatory Project Assignment AY 2018-2019

[2] ISO/IEC/IEEE 29148 - Systems and software engineering - Life cycle processes - Requirements engineering

[3] Collection and Processing of Data from Wrist Wearable Devices in Heterogeneous and Multiple-User Scenarios
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5038811/`

[4] Wearable Devices in Medical Internet of Things: Scientific Research and Commercially Available Devices
`https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5334130/`

[5] Google Fit API
`https://developers.google.com/fit/overview`

[6] PayPal API
`https://developer.paypal.com/docs/`

[7] RapidSOS Emergency API
`https://info.rapidsos.com/blog/product-spotlight-rapidsos-emergency-api`

[8] Slides of the course by Prof. Di Nitto
`https://beep.metid.polimi.it/`

[9] LaTeX templates
`http://www.latextemplates.com/`

[10] Draw.io
`https://www.draw.io/`

[11] Diagrams
`https://www.visual-paradigm.com/features/uml-tool/`