POLITECNICO
MILANO 1863

TrackMe
Software Engineering 2 Project
DD Document

Stefano Martina, Alessandro Nichelini, Francesco Peressini

A.Y. 2018/2019
Version 1.0.0

December 5, 2018

Contents

1 Introduction

1.1 Purpose

1.2 Scope

1.3 Definitions, Acronyms, Abbreviations

1.4 Revision history

1.5 Reference Documents

1.6 Document Structure

2 Architectural design

2.1 Overview: High-level

The system is going to be implemented with a three tier architecture. Tiers are as briefly described by the following schemas.

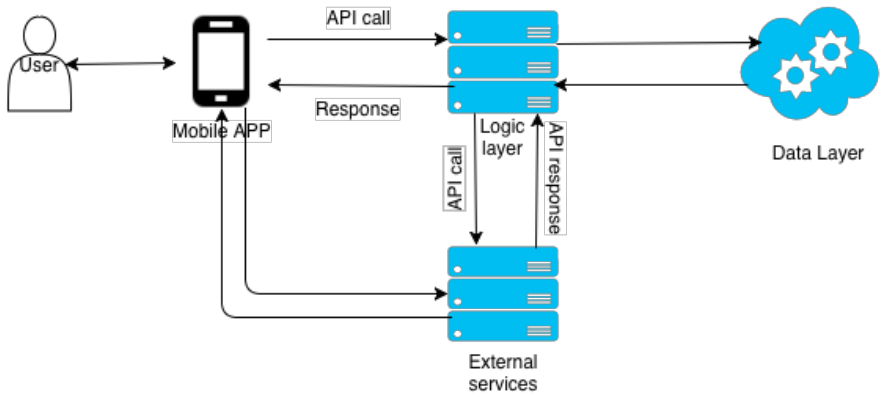

Figure 1: High level view of the system's architecture

The decision of this kind of architecture has been taken in order to build the system in the most modular possibile way. Here are described layer organisation:

- Presentation layer:
  - *Mobile clients*: users will be given with a iOS application which will be a view of the entire system.
  - *Third parties*: third parties will be given with a light web interfaces to register/manage API access and they will be authorised to communicate with the system.

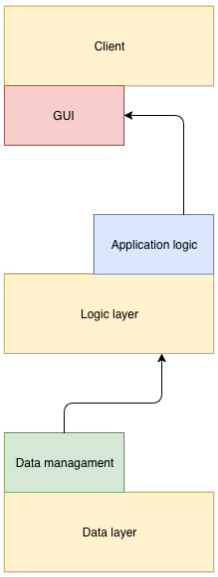

Figure 2: Distribution of application's function among the tier

- *Logic layer*: logic layer will implement all the logic of the entire system and will handle communications between clients' app and the data layer.
- *Data layer*: data layer will be implemented in third party's cloud system and will keep persistent users' data.

The idea is to keep as separate as possibile the logic layer from the data layer in order to let the system grow in a modular fashion and let us change cloud data provider as the system's dimension grow with the minimum effort.

2.2 Component view

We now provide a high level view of system's components. The whole system can be seen as two main component *WebInterface* e *Mobile Application* that consumes services offered by the three main important subsystem: *HealthSharing Manager*, *SOS Manager*, *RunEvent Manager*. We are going to provide further details of each subsystem.

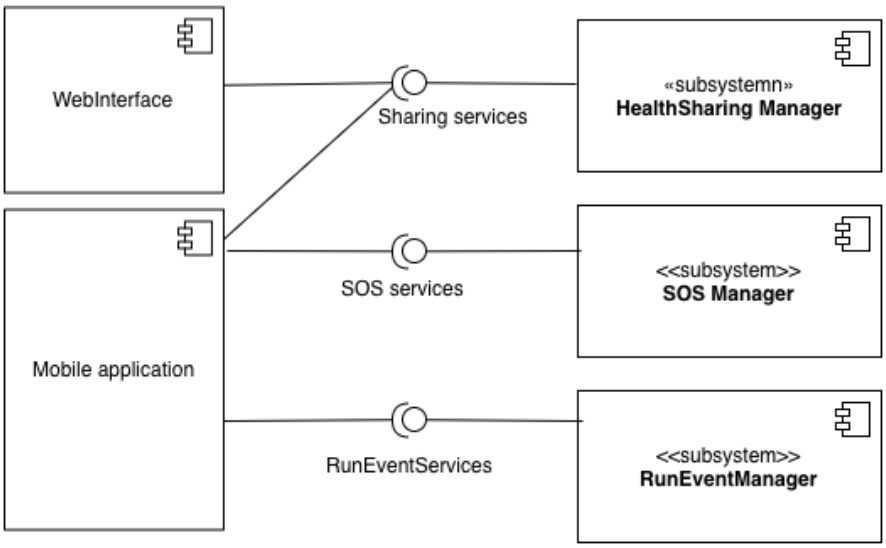

Figure 3: High level overview of system components.

2.2.1 Component view of HealthSharing Manager

The *Access Policy Manager Module* works as a manager of all the policies associated with data sharing. It provides the list of active sharing to users and it let them manage active policies such as accepting new sharing request, or change/delete active policies. Moreover this component is also used by third parties that have to be able to subscribe to users data.
The *Data Manager Module* has to handle all the operations of retrieving/storing data between users' app and databases. It also has to guarente data consistency in the whole system.

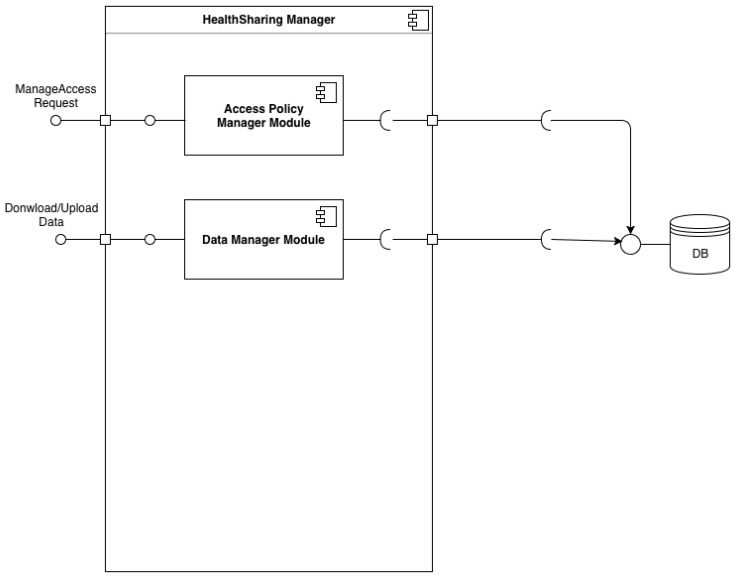

Figure 4: Specific description for HealthSharing Manager component.

2.2.2 Component view of SOS Manager

The *Anomaly Detection Module* has to live read and control heartbeat data from users that have activated SOS functionality in their mobile application.

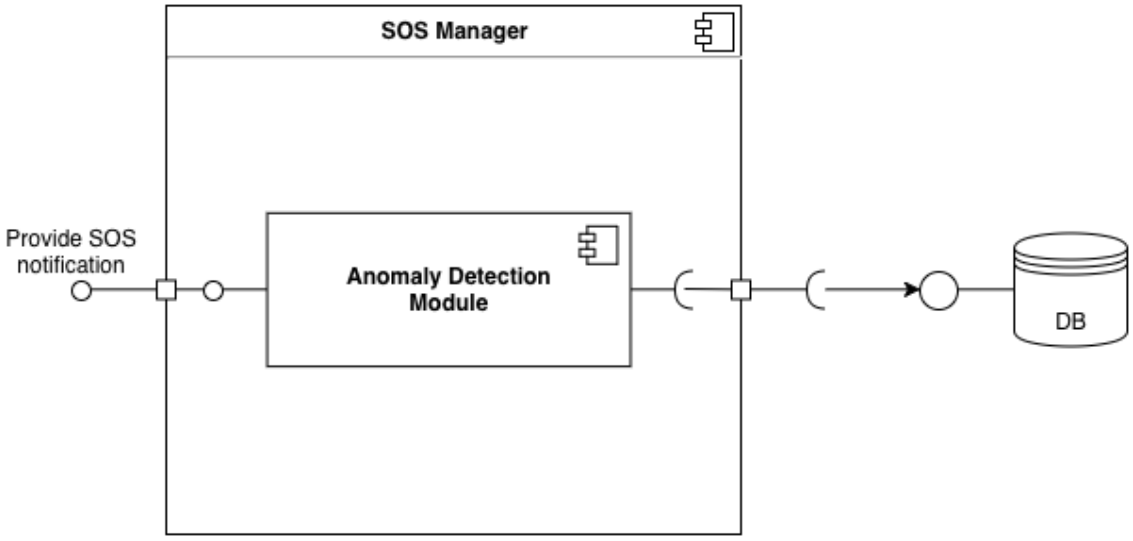

Figure 5: Specific description for HealthSharing Manager component.

2.3 Deployment view

The Deployment diagram emphasises the nodes on which the platform runs: **InHouse** box represent the components developed inside the company. The components in the **External box** represent the on demand software.

- **Smartphone** : the application deployed on the iOs Smartphone used by the user. Users are able to retrieve data from the application server and, also, from the external server API directly (e.g. maps by Google Maps)
- **Web Browser API Interface** : A web page for Third Parties registration. It provides a secret to do a request to the API Server.
- **Application Server** : the main logic core of the application. Its the only one access point to the DB but also it communicates with the external server API
- **External Service** : Both the external services are represented as a black-box because we dont know how they are implemented

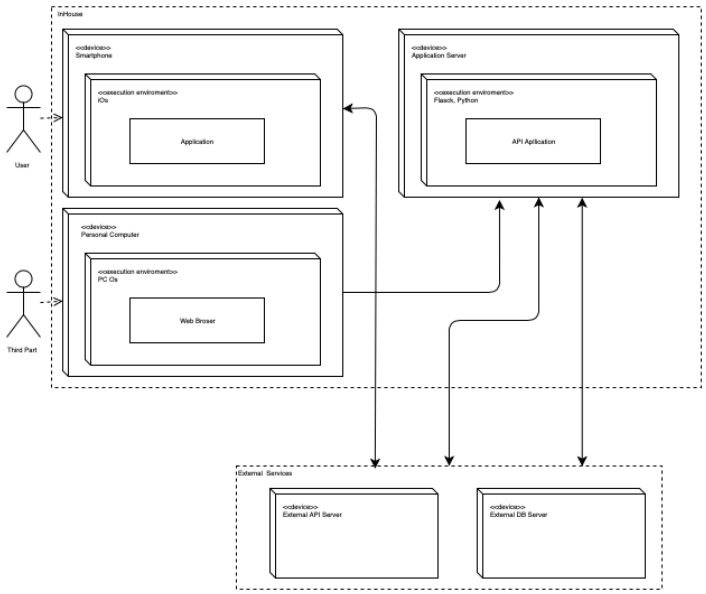

Figure 6: Deployment Dyagram

2.4 Runtime view

2.5 Component interfaces

2.5.1 API structure

All the api system will be implemented referring to a single endpoint www.data4help.cloud. Users' applications and third parties will refer to different subdomain:

- *www.data4help.cloud/api/users* will be the specific endpoint for the application that serves users.
- *www.data4help.cloud/api/thirdparties* will be the specific endpoint for third-parties.

2.6 Selected architectural styles and patterns

Data4Help system is based on a tree tier architecture:

2.6.1 Overall Description

- **Presentation Tier** has to present the data to the user in such a way that they are meaningful for the users perspective. It displays information to the user obtained communicating with the other two tiers. For example, if a user performs a tap on the button create a new run event the communication tier send a query to the Logic Tier that answers with a message requiring for information to create an Event and so on. To be quick to react in case of Anomaly detection, instead of putting the logic of continuous checking of the vital parameters into the Logic Tiers (as the entire logic of the application) the Presentation Tier (the application) contains also this component. Doing this we will be able to speedy react and call the ambulance in the shortest way possible. We are also able to react in narrow situation as absence of field coperture or not working internet network.

- **Logic Tier** has the entire logic needed for the application (excluding the Anomaly detection component as described above). It receives the requests from the Presentation Tier and, if needed, queries the Data tier before then sending the reply.

- **Data Tier** is composed of a DBMS that provide an interface to the DataBase externally located. It contains all the data in a Relational Schema (ER) (view fig XXX)

  The separation between the different layers allow us to upgrade or replace a single tier without affecting the others

2.6.2 Design Patterns

2.7 Other design decision

3 Algorithm Design

3.1 Anomaly Detection

One of the main critical aspect of the platform is Anomaly detection. As described in section 2.6 the logic of this component is inside the application, to avoid issues like network congestion or not server not available. Here is a pseudocode of the component:

```
Threshold = UserPreference Threshold or default value
Emergency = 0

Loop on data feed receiving
    Check if ( BPMReceived < Threshold )
        -- "means that the smartwatch recorded a value under the threshold"

        Check if ( Emergency = 0 )
            Emergency ++

        Check if (Emergency = 3)
            -- "means that three BPM values are under threshold"
            call the ambulance number except if the user taps on 'Its not an emergency'

    Else
        Check if( Emergency != 0 )
            -- "means that the BPM goes under threshold for brief interval"
            Emergency = 0

        continue the data recording
```

4 User interface design

5 Requirements traceability

6 Implementation, integration and test plan

7 Effort spent

8 References