PEOPLE'S DEMOCRATIC REPUBLIC OF ALGERIA
MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH
UNIVERSITY MOHAMED BOUDIAF - M'SILA

FACULTY: Mathematics and Computer Science
DEPARTMENT: Computer Science
N°:……

DOMAIN: Mathematics and Computer Science
BRANCH: Computer Science
OPTION: RTIC

Dissertation submitted to obtain Master degree

By: MR. ZAHAR Youcef

SUBJECT

Mobile Application for Medical Assistance

Supported before the jury composed of:

…….	University of M'sila	President
Mr.Akhrouf Samir	University of M'sila	Supervisor
…….	University of M'sila	Examiner
…….	University of M'sila	Examiner

Academic year: 2018/2019

Dedications

To my dear parents.

To my small and large family.

To my friends and relatives.

MR. Zahar Youcef

ACKNOWLEDGEMENTS

I am honored to write these lines to express my thanks and gratitude to anyone who helped me directly or indirectly from afar or from near.

First, my greetings and thanks to my father and my mother who were with me throughout my studies.

My greetings and thanks to the framed professor: Akhrouf Samir.

I would also like to thank the members of the jury for having devoted a part of their time reading this memoir and for their interest in this work.

I thank very much my friends who helped me: Geusri Tariq and Chaker Rachid.

Finally, without exception, I thank all my friends and all who helped me from near.

Table of content

Table of content i

Figures table iii

List of tables iv

General Introduction 1

1. General information about mobile applications. 3
1.1. Introduction…… 3
1.2. Mobile Applications…….. 3
1.2.1. Definition…… 3
1.2.2. Categories and types of mobile applications…….. 3
1.2.3. Client/Server Application…… 4
1.2.4. Mobile operating systems…….. 5
1.3. Operating system Android……. 6
1.3.1. Android definition…….. 6
1.3.2. Android versions……. 6
1.3.3. Android System Architecture……. 7
1.4. Subject presentation…… 8
1.4.1. Medical emergency……. 8
1.4.2. Problematic……. 8
1.4.3. Solution in a mobile application image…….. 9
1.5. Conclusion……. 9
2. Analysis and design. 10
2.1. Introduction…… 10
2.2. Initial expression of needs……. 10
2.3. Modeling needs…….. 10
2.3.1. Functional requirements……. 11
2.3.2. Non-functional requirements…….. 12
2.3.3. Specification of requirements according to use cases…… 12
2.3.4. Detailed specification of requirements……... 16
2.3.5. Sequence Diagrams……. 21
2.4. Class Diagram…… 26
2.5. Conclusion……. 27
3. Implementation. 28
3.1. Introduction……... 28
3.2. Development environment…… 28
3.2.1. Hardware environment…….. 28
3.2.2. Software environment…… 28
3.2.2.1. On the server side……... 29
3.2.2.2. On the client side…… 32
3.3. Implementing the database…… 34
3.4. Presentation of the application…….. 36
3.4.1. Tree view of our application…….. 36

3.4.2. Presentation of the interfaces…… 37
3.4.3. Security…….. 45
3.5. Conclusion……. 45

General conclusion **46**

Bibliography **47**

Figures table

2.1 Use case diagram associated with administrator…… 14
2.2. Use case diagram associated with user……... 15
2.3 Use case diagram associated with civil protection……. 16
2.4 Sequence diagram for authentication……. 22
2.5 Sequence diagram for add profile/medical file…….. 23
2.6 Sequence diagram for SOS request…… 24
2.7 Sequence diagram for Blood request……. 25
2.8 Sequence diagram for find hospitals……. 26
2.9 Class diagram our application…… 27
3.1 REST API……... 29
3.2 Getting Google Maps API Key…….. 33
3.3 Permissions of application…….. 34
3.4 Tree view of our application…….. 37
3.5 Splash Interface……... 38
3.6 Icon of application……... 38
3.7 The Front interface…….. 39
3.8 Create a new account interface……... 39
3.9 Create profile/medical file interface……... 40
3.10 Profile interface……. 41
3.11 notifications interface…… 41
3.12 medical file interface…… 42
3.13 Settings interface…….. 42
3.14 SOS Request interface…….. 43
3.15 Location and Nearby Hospitals interface……. 43
3.16 Alarm with sound and light interface…….. 44
3.17 Blood Request interface……... 44
3.18 Token form……. 45

List of tables

2.1 Textual description of the use case Authentication…….. 17

2.2 Textual description of the Use case create profile and medical file……. 18

2.3 Textual description of the Use case request SOS……. 19

2.4 Textual description of the Use case request blood…… 20

2.5 Textual description of the Use case identify the nearby hospitals…… 21

General Introduction

Today, our lives have become easy in the era of technology, where people are using them in all of their daily activities. Among them is the smart phone, which has become popular for its usefulness and importance. The phone is now no longer used for calls and messages only, but takes a different dimension. It offers many services such as running games, watching movies, watching news, navigation and running smart applications ..., The integration of the Internet with smart phones allowed to create chat applications and applications of social networking and server / client applications that contributed to the development of all areas and facilitate life, especially in the medical field, which provided great additions.

Within the medical field, especially medical assistance and medical ambulance. Helping people in serious situations and preserving their lives is of great importance. In accidents of all kinds, the life of the victims is in serious danger and requires good medical care. In such cases, assistance is requested by contacting the emergency numbers. However, it is difficult for paramedics to locate the victims and when reaching the victims, it is difficult to identify them and to know their medical file in order to act quickly. After the transfer of the victims to the hospital, it is difficult to inform the families of the lack of information. In some cases, the victim needs blood and is not available and there are many other problems. These problems may prevent the paramedics from saving the victims and preserving their lives.

In this context, we are motivated to design and develop an application that works on smart phones works as a server / client. The main objective is to facilitate the provision of medical assistance and to make additions in this area. We try to use them to solve the problems that medical professionals face in providing medical care which may increase the success of rescue cases and avoid the large number of victims. We chose the android platform to run the application for the great abundance of smart phones that use this platform and is open source and free. The server will also be compatible with various devices and operating system.

This dissertation is divided into three chapters describing the steps to be taken mobile application for medical assistance. In the first chapter we talk about smart applications and the most important platforms, explain the type of server / client applications. We look at medical assistance and its problems and our way of solving these problems. In Chapter II. We analyze

the needs and identify the actors that will play a role in the management tool for implementation. We define the functions and usage cases of these actors through the use of case graphs and sequences. In the third chapter we mention the development environment we chose from the server side and the smart application ,followed by a presentation of the application screens and how to use them. Finally, we conclude with a general conclusion summarizing the main points of our work and the degree to which we achieve our goals.

CHAPTER ONE
GENERAL INFORMATION ABOUT MOBILE APPLICATIONS

1.1. Introduction:

Mobile phone customers today want to take advantage of the applications on their phones. They use these applications to complete routine tasks such as taxi request, home booking, news viewing, GPS usage, social networking sites and more.

In this section we talk about mobile applications and their types, about client-server-type applications, as well as more known mobile frameworks. Finally, take a look at the area of medical assistance (its problems and importance) and speak to solve its problems in the form of a mobile application.

1.2. Mobile Applications:

A mobile app is a product that is explicitly developed for use in small computers, for example, smartphones, advanced tablets, smart cars, and more, rather than desktop or laptop computers. Mobile apps are designed with consideration for the demands and constraints of the devices and also to take advantage of any specialized capabilities they have [1]

The mobile application contains extensive uses such as communication, messaging, browsing, chat, social networking, audio, video, games, etc. In a large number of mobile applications, some have already been installed in the phone and others can download and install on the mobile phone. Technically, different mobile applications run on different managed platforms like the iPhone, BlackBerry, Android, Symbian, and Windows, some virtual machines [3].

1.2.2. Categories and types of mobile applications:

1.2.2.1. Types of mobile applications**:** Mobile applications are categorized as [2]:

1. NATIVE APPS:

   One of the most popular mobile phone applications is the " NATIVE " for a particular device or platform where it is developed for a single operating system. Applications developed for platforms like Android, Windows, iOS, BlackBerry, Symbian, and Windows phone only work on their platforms.

2. HYBRID APPS:

Applications developed using multi-platform technologies are called as Hybrid apps. For instance, CSS, JavaScript, and HTML5 are used together to develop hybrid apps. These are web site applications concealed in a native package.

3. WEB APPS:

Web applications are mainly software applications having behavioral tendencies almost similar to native apps. They use a single browser to run and are built in CSS, JavaScript, or HTML5. With the help of these apps, you can redirect users to your site URL and offer the option to install by just creating a page bookmark.

1.2.2.2. Categories of mobile applications**:**

While Apple's App store categorizes apps under 24 categories and Google Play categorizes mobile apps in 33 types, only 7 types of mobile applications have managed to successfully reach the users**.** So, know about these 7 categories of apps that have been able to make it to the market [2]:

- Gaming applications
- Business applications
- Educational applications
- Lifestyle applications
- Entertainment applications
- Utility applications
- Travel applications

1.2.3. Client/Server Application:

The client-side application and access to the remote server for information and services are called the client / server application. User interaction with the server is always through a client-side interface or application, sending requests and responding or rejecting the server. the client uses the network to send and receive communications about its order, or request. the server will take the request and make sure that the request is valid. if everything checks out okay, then the server will fetch the request and serve the client. the server can make a request from the client as well. it may want to check up on the status of the client, or ask if it has received any

security patches, or if it still needs resources from the server. if not, the server will close the connection in order to free up network traffic.

1.2.3. Mobile operating systems:

A mobile phone like a computer uses an operating system to run mobile applications, we mention the most Mobile operating systems [4]:

1. Android:

September 20, 2008 was the date Google launched the first Android operating system called "Astro", and Google adopted the trend of naming Android versions with candy names. After entering the smartphone and tablet market, Android gained popularity because of its beautiful appearance and effective work, many new features that have played an important role in Android success, Samsung, HTC, Motorola and many other top Android manufacturers are using their devices. Android is currently one of the best operating systems and is a serious threat to iPhone.

2. Apple IOS:

IOS was introduced in 29th June 2007 when the first iPhone was developed. Since then IOS has been under gone many upgrades. Apple has still not allowed any other manufacturer to lay hands on its operating system. Unlike Android, Apple has more concentrated on the performance along with appearance. This is the reason that the basic appearance of IOS is almost the same as it was in 2007. Overall it is very user-friendly and is one of the mobile best operating systems in the world.so far IOS has been used in all iPhones, iPod & iPad.

3.Blackberry OS:

Blackberry OS is the property of RIM (Research in Motion) and was first released in 1999, RIM has developed this operating system for its Blackberry line of smartphones. Blackberry is much different from other operating systems. The interface style, as well as the Smartphone design, is also different having a trackball for moving on the menu and a QWERTY keyboard. Like Apple, Blackberry OS is a close source OS and is not available for any other manufacturer. It is a very reliable OS and is immune to almost all the viruses. Some of the smartphones operating on Blackberry OS are Blackberry Bold, Blackberry Curve, and Blackberry Torch [4].

4.Windows OS:

All of you will be familiar with the Windows operating system because it is used in computers all over the world. Windows is also used on mobile phones, but ordinary mobile users have difficulty running it, but at the same time it was very popular among people who used to run it. This was the case until Nokia and Microsoft cooperated to work together. Through its colorful and easy-to-use interface, Windows gives a new life. [4].

1.3. Operating systems Android:

we talk about the Operating systems Android

1.3.1. Android definition:

Initially founded by Andy Rubin in October 2003, and later acquired by Google on August 17, 2005. Android is a free Linux based platform and is an open software stack with an operating system and middleware. It was originally developed by Google and released on November 5, 2007, for mobile platforms. The T-Mobile's G1 phone (HTC Dream) is the first phone to be released to the public with Google Android on September 23, 2008. Android is a strong rival to the Apple IOS. [5].

1.3.2. Android versions:

Below is a listing of each of the different major Android versions [5]:

Android version 1.0 released on September 23, 2008.

Android version 1.5 (**Cupcake**) released on April 27, 2009.

Android version 1.6 (**Donut**) released on September 15, 2009.

Android version 2.0 released (**Eclair**) on October 29, 2009.

Android version 2.2 (**Froyo**) released on May 20, 2010.

Android version 2.3 (**Gingerbread**) released on December 6, 2010.

Android version 3.0 (**Honeycomb**) released on February 22, 2011.

Android version 4.0 (**Ice Cream Sandwich**) released on October 8, 2011.

Android version 4.1 (**Jelly Bean**) released on July 9, 2012.

Android version 4.4 (**KitKat**) released on October 31, 2013.

Android version 5.0 (**Lollipop**) released on November 3, 2014.

Android version 6.0 (**Marshmallow**) released on October 5, 2015.

Android version 7.0 (**Nougat**) released on August 22, 2016.

Android version 8.0 (**Oreo**) released on August 21, 2017.

Android version 9.0 (**Pie**) released on August 6, 2018.

1.3.3. Android System Architecture:

The Android software stack generally consists of a Linux kernel and a collection of C/C++ libraries that is exposed through an application framework that provides services, and management of the applications and run time [6]:

1.Linux Kernel**:** Android was created on the open source kernel of Linux. One main reason for choosing this kernel was that it provided proven core features on which to develop the Android operating system

2.Libraries**:** Running on the top of the kernel, the Android framework was developed with various features. It consists of various C/C++ core libraries with numerous of open source tools.

3.Android Runtime**:** It is the third section of the architecture. It provides one of the key components which is called Dalvik Virtual Machine. It acts like Java Virtual Machine which is designed especially for Android. Android uses its own custom VM designed to ensure that multiple instances run efficiently on a single device. The Delvik VM uses the device's underlying Linux kernel to handle low-level functionality, including security, threading and memory management.

4.Application Framework**:** The Android team has built on a known set proven library, built in the background, and all of it these is exposed through Android interfaces. These interfaces warp up all the various libraries and make them useful for the Developer. They don't have to build any of the functionality provided by the android.

5.Applications**:** Android applications can be found at the topmost layer. At application layer we write our application to be installed on this layer only. Examples of applications are Games, Messages, Contacts etc.

1.4. Subject presentation:

In this section, we talk about our object.

1.4.1. Medical emergency:

The purpose of any emergency department is to save lives. An emergency is any medical problem that may cause death or permanent injury if not treated quickly. In some cases, severe pain may also be a medical emergency, such as bloody, broken, heart attack... [7].

Some examples of medical emergencies are:

- Some examples of medical emergencies are:
- Chest pain, shortness of breath or feeling that your heart is beating irregularly or very fast.
- Acute bleeding does not stop after 15 minutes of direct pressure.
- fainting.
- Broken or displaced bones.
- The poison swallows.
- Burns.
- Effects of car accidents and accidents at work.

In an emergency case we must go to the emergency department, call emergency numbers or call an ambulance.

1.4.2. Problematic:

In Emergency Medical Assistance (Civil Protection, Samu ...),in the event of accidents (at the street, home or work place ...) or sudden cases that are in a critical condition that need good care, the reason may be to save many victims and reduce dead accidents, but the paramedics face many problems Such as late reporting time, difficulty in acting quickly, inability to accurately locate the scene, road congestion problems, lack of information on the victim (health status, blood type, consumables, information about relatives ...), problems Relating to the preparation of the victim reception in the hospital, Based on the above, what is the solution to the problems facing paramedics. Or at least to reduce these problems.

1.4.3. Solution in a mobile application image:

Because we are in the era of technology and smart phones, we suggest problematic solving in the form of a mobile application for emergency medical assistance, this application works on the Android operating system, we provide the following in the application:

- Enable emergency services officers' instant access to information about victims to effectively take care of them at the scene of the accident.
- Have the opportunity at the scene of the accident to complete a medical form for the preparation of the emergency service of the nearest hospital.
- To make calls to blood donors and locate them geographically.
- Give the possibility to anyone to launch SOS directly from this application to the services concerned in order to quickly locate the victim (s).
- Can easily identify victims to access their medical records and inform their relatives.
- Ask for help from the phone even in lock mode.

1.5. Conclusion:

The medical domain in these days has become digitized and this has made us deal with our problem through a mobile application for its benefits and use smart phones to save people's lives, in this chapter we first mentioned smart applications and their types, then we talked about the famous mobile operating systems, We dedicated a side to talking about the Android operating system, Finally, we have raised a problematic about emergency medical assistance care and the way we solve this problems. The next chapter will be devoted to the need's analysis and the design of our mobile application for medical assistance.

CHAPTER TWO:

ANALYSIS AND DESIGN

2.1. Introduction:

The first stage of design is to analyze the situation to take account of constraints, risks and everything related to the development of the system that serves the user. This chapter allows us to identify all the features of our future application for both the user and the manager by identifying descriptive needs, and allows for the creation of a list of requirements translated by non-functional requirements, this will be done by identifying the actors and defining all the needs that will be modeled by a use case diagram for each entity, followed by their sequence diagrams and finally the class diagram.

2.2. Initial expression of needs:

We will have a mobile phone application in the future as a goal to ensure good management of medical assistance in cases requiring assistance, facilitate the process of requesting help and transfer victims to hospitals. The application should make it easy to manage and access the medical records of the victims so that they can be treated quickly and inform their families. Users of the application will find many privileges in terms of ease and speed of seeking help in serious cases, save their health information for cases of ambulance, in cases of need for blood can be requested from other users, and can search for the nearest hospital from their location, the application will also offer the advantage of asking for help even when locking the phone, in case of fainting and getting help from someone else. So far to ensure better assistance of the emergency of victims by paramedics.

2.3. Modeling needs:

In this section, we identify the features of our future mobile application. these features are then modeled by appropriate UML diagrams.

2.3.1. Functional requirements:

We will have a mobile phone application to collect all the functions needed for medical assistance, request SOS, access to medical files, blood request, GPS and the most important files provided by the application.

- Server Management:

There will be a server administration that works with the application, by receiving user information and storing it in the database in the form of profiles and medical files. At SOS request, the server receives the location information from the device and, based on the identification number, extracts the victim's medical files and sends them to the nearest civil protection center, and then sends a notification to his relatives. After a period of time the server sends the relatives to ask about the need for blood. If the answer is yes, he will do the blood request protocol. When the blood is requested, the server receives the information of the hospital and the location, based on the identification number used to extract the blood group and then searches the database for the nearest users and the same blood group to send them alerts.

- SOS Request :

The user will be able to request SOS service through a button on the application home page or by shortening the application in the closed screen, so that when pressing the button, the app sends accurate location information with the user ID to the server, the server then executes the protocol corresponding to the request.

- Blood Request:

The user will be able to request blood by pressing the button on the home page after entering the hospital information and location, the application sends the information to the server, the server then executes the protocol corresponding to the request.
In another case, after a period of SOS service, the server intervenes and asks relatives about the victim's need for blood.

- Search for the nearest hospital:

In case the user can go himself to the hospital, the application will provide a search page for the nearest hospital from where it is located.

- Profile and medical file:

Each user will have their own profile and medical file, so they can change their information, view or send their medical information to their doctor.

- Signal light and sound:

In the event that the victim is in a place that is difficult for the paramedic to identify or is deaf, unable to speak, the application will provide him with a button that activates an alarm sound with light signals from the telephone so that the medics can reach him and locate him.

2.3.2. Non-functional requirements:

We divide non-functional requirements into two categories: the quality requirements and performance requirements.

2.3.2.1. Quality requirements:

- Attractive and efficient ergonomics:

Interface design should allow immediate recognition of its various elements to allow the user to intuitively access what he is looking for, directly from the first use. Priority must be given to the important elements of the application.

- Intelligible forms:

Forms should contain only the minimum number of fields to be filled in. We should strictly restrict the user to the information we want from him, and should therefore be taken care of in particular.

2.3.2.2. Performance requirements:

- User information must be confidential and secure and only used under the agreement.
- Mistakes in communication between application and server should be avoided.
- The priority of the application should be to save the victims.

2.3.3. Specification of requirements according to use cases:

Actors and use cases are the fundamental UML concepts for requirements specification.in this section, we will identify them from the initial expression of the needs of our case study.

We will structure, link and then classify these use cases and elaborate associated UML graphics representations.

2.3.3.1. Identification of actors:

The actor represents every role played by an external entity (human user, device, or other system) that directly interacts with the system being studied.

In the case of our system, we basically identified three interactive actors:

- The user**:** Means the user of the application who has powers after the registration process and then uses the application in cases that are victim and needs help.

- The administrator**:** He is the person responsible for maintaining the application and managing the user accounts and responsible for communicating with the authorities responsible for the rescue. It ensures the proper functioning of the server and the data server and its integrity.

- Civil Protection**:** Means the party to whom the application works in medical assistance operations.

2.3.3.2. Identification of use cases:

A use case represents a set of action sequences that are realized by the system and which produce an observable result of interest to a particular actor.

Let's take the three representatives and list the different ways to use the system in the future:

1. Administrator:
    - Login.
    - System Management.
    - view and manage profiles.
    - Add and manage services.

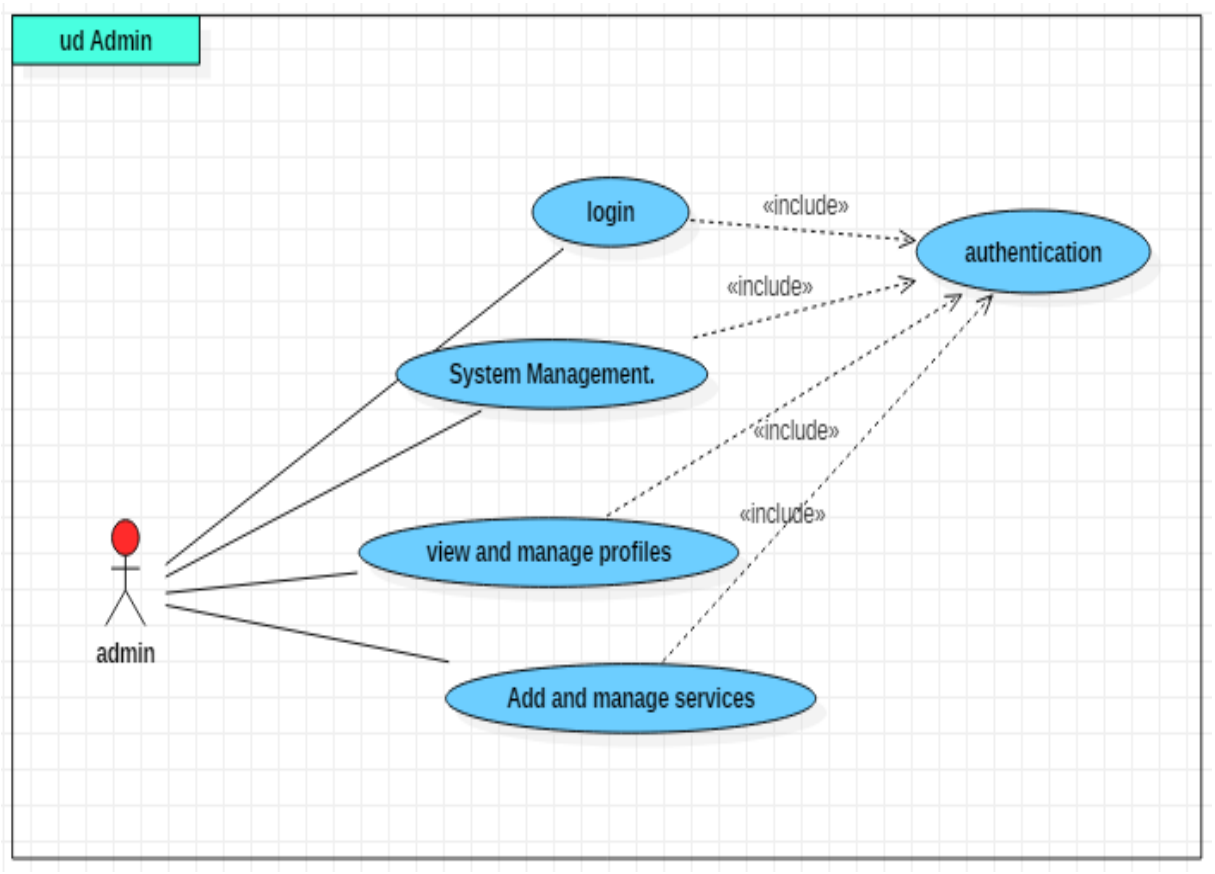

Figure2.1 Use case diagram associated with administrator.

2. User:

- Create a user account.
- Login and Logout.
- See profile and medical file.
- Search for the nearest hospital.
- Sent SOS.
- Blood request.

The authentication use case is a case that must be realized in order to allow each actor to execute his own use cases. These use cases are shown in the figure 2.2.

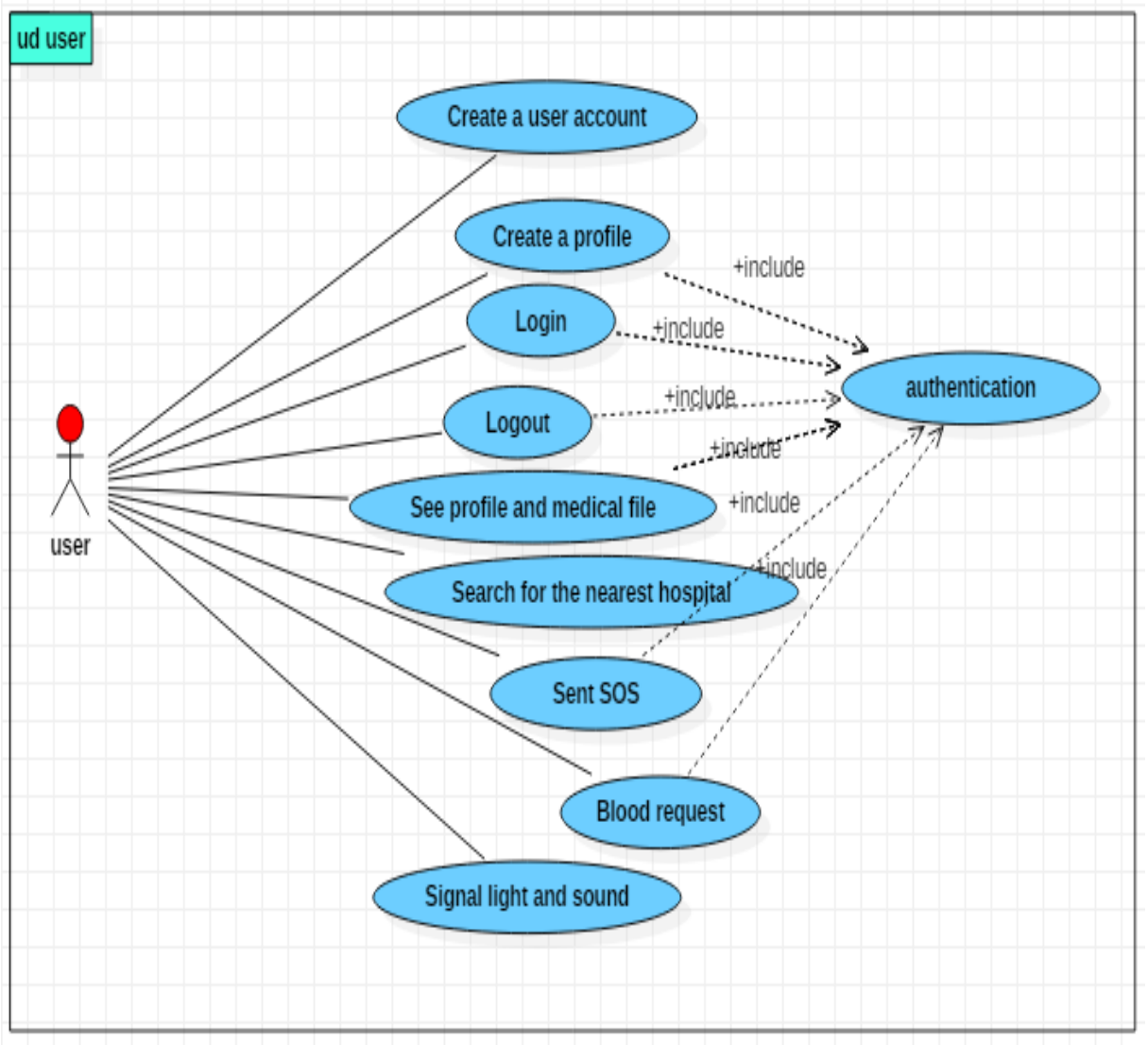

Figure2.2 Use case diagram associated with user.

3. Civil Protection:
    - SOS reception from the server.
    - Answer by accepting or rejecting the rescue mission.
    - Inform the server with information about the mission.

These use cases are shown in the figure 2.3.

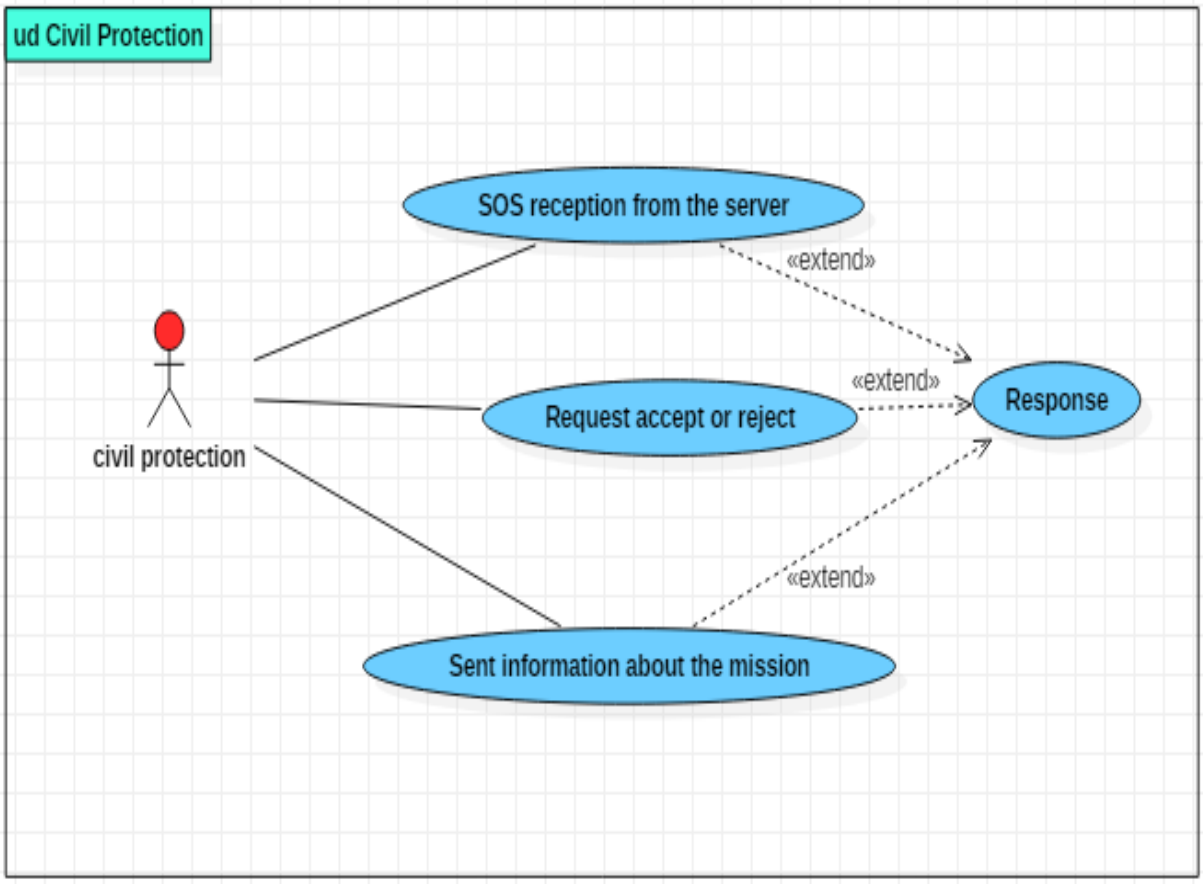

Figure2.3 Use case diagram associated with civil protection.

2.3.4. Detailed specification of requirements:

In what follows, we will describe in detail some of the previously identified use cases by listing in a textual way all the interactions between the actors and the system.

2.3.4.1. Use case Authentication:

This use case is shown in the table 2.1.

Identification Summary		
Use case title	Authentication	
Resume	Authentication provides access to reserved features to a given type of user.	
Actors	User, Administrator.	
Description of scenarios		
Preconditions	Accessible application.	
Nominal scenario	1.The user accesses the authentication page. 2. application displays the form authentication. 3.The user enters his email and his password. 4.The application sends the information to the server.	5.The server checks the existence Account. 6.The server sends a token to the application. 7.The application stores the token and enters the user to the home page.
Error Chains	-No account matching the couple email / password indicated: The system shows an error, the use case ends with failure.	
Post conditions	The user is authenticated and accesses the features that are dedicated.	

Table 2.1 Textual description of the use case Authentication.

2.3.4.2. Use case create profile and medical file**:**

This use case is shown in the table 2.2.

Identification Summary		
Use case title	Create profile and medical file	
Resume	Creating a profile / medical file allows each user to identify specific information and medical information. This information is helpful in the medical assistance process.	
Actors	User.	
Description of scenarios		
Preconditions	Accessible application.	
Nominal scenario	1.The user creates an account for authentication. 2. The user enters his or her login information email/password. 3. The server checks the exist once account and sends a token. 4.The application shows the page form to create the profile/medical file.	5.The user enters the identity and health information. 6.the application sends the information with the token to the server. 7.The server stores information in a database on the cloud. 8. The application enters the user to the home page.
Error Chains	-No account matching the couple email / password indicated: the system shows an error; the use case ends with failure. -Expiration of token: the use case ends with failure.	
Post conditions	A new medical file and profile is added to the list.	

Table 2.2 Textual description of the Use case create profile and medical file.

2.3.4.3. Use case Request SOS:

This use case is shown in the table 2.3.

Identification Summary		
Use case title	Request SOS	
Resume	SOS is required when the user is a victim of an accident or a case that requires assistance. The user then uses this service to request help.	
Actors	User.	
Description of scenarios		
Preconditions	Accessible application. The user is authenticated.	
Nominal scenario	1.The user presses the SOS icon and confirms the demand. 2. The application extracts the site information with token and sends it to the server. 3. From the token, the server extracts the victim's information and sends it to the nearest civilian protection center from the victim's site 4.After a period of time if the server does not receive an answer, a request is sent again to the nearest center	4.The Civil Protection Center responds to the receipt and acceptance of the application. 5.The server sends a message that the help is coming to the application. 6.After the task is completed, the center sends a report about the process to the server.
Error Chains	-The phone is unable to extract location information: the system shows an error; the use case ends with failure. -Civil Protection Center refused to request: the use case ends with failure.	
Post conditions	A new medical assistance operation.	

Table 2.3 Textual description of the Use case request SOS.

2.3.4.4. Use case Request Blood:

This use case is shown in the table 2.4.

Identification Summary		
Use case title	Request Blood	
Resume	Blood request is a service to the user when in hospital and needs blood	
Actors	User.	
Description of scenarios		
Preconditions	Accessible application. The user is authenticated.	
Nominal scenario	1.The user presses the blood icon and Enter information about the hospital. 2. The application sends the information with token to the server.	3. From the token, the server extracts the victim's blood type and sends a notification to everyone who has the same blood type and be addressed near the hospital.
Error Chains	-No one has the same blood type or no one is close to the hospital: the use case ends with failure.	
Post conditions	A new blood donation operation.	

Table 2.4 Textual description of the Use case request blood.

2.3.4.5. Use case Identify the nearest hospital**:**

This use case is shown in the table 2.5

Identification Summary		
Use case title	**Identify the nearest hospital**	
Resume	Service The user is able to identify the nearest hospital from its place in case it was able to go alone	
Actors	User.	
Description of scenarios		
Preconditions	Accessible application. The user is authenticated.	
Nominal scenario	1.The user enters the maps page. 2. The user clicks on the find Hospitals icon.	3. The application shows nearby hospital locations on the map
Error Chains	-AGPS device failed to locate: the use case ends with failure.	
Post conditions	Identify the nearest hospital to the user.	

Table 2.5 Textual description of the Use case identify the nearest hospital.

2.3.5. Sequence Diagrams:

The purpose of sequence diagrams is to represent the interactions between objects. This representation can be realized by use case considering the different associated scenarios. In what follows, we represent the sequence diagram of a scenario representative of each of the use cases described above.

2.3.5.1 Use case Authentication**:**

The first scenario for the user is to authenticate with the system. The chronological of this scenario is shown in Figure 2.4.

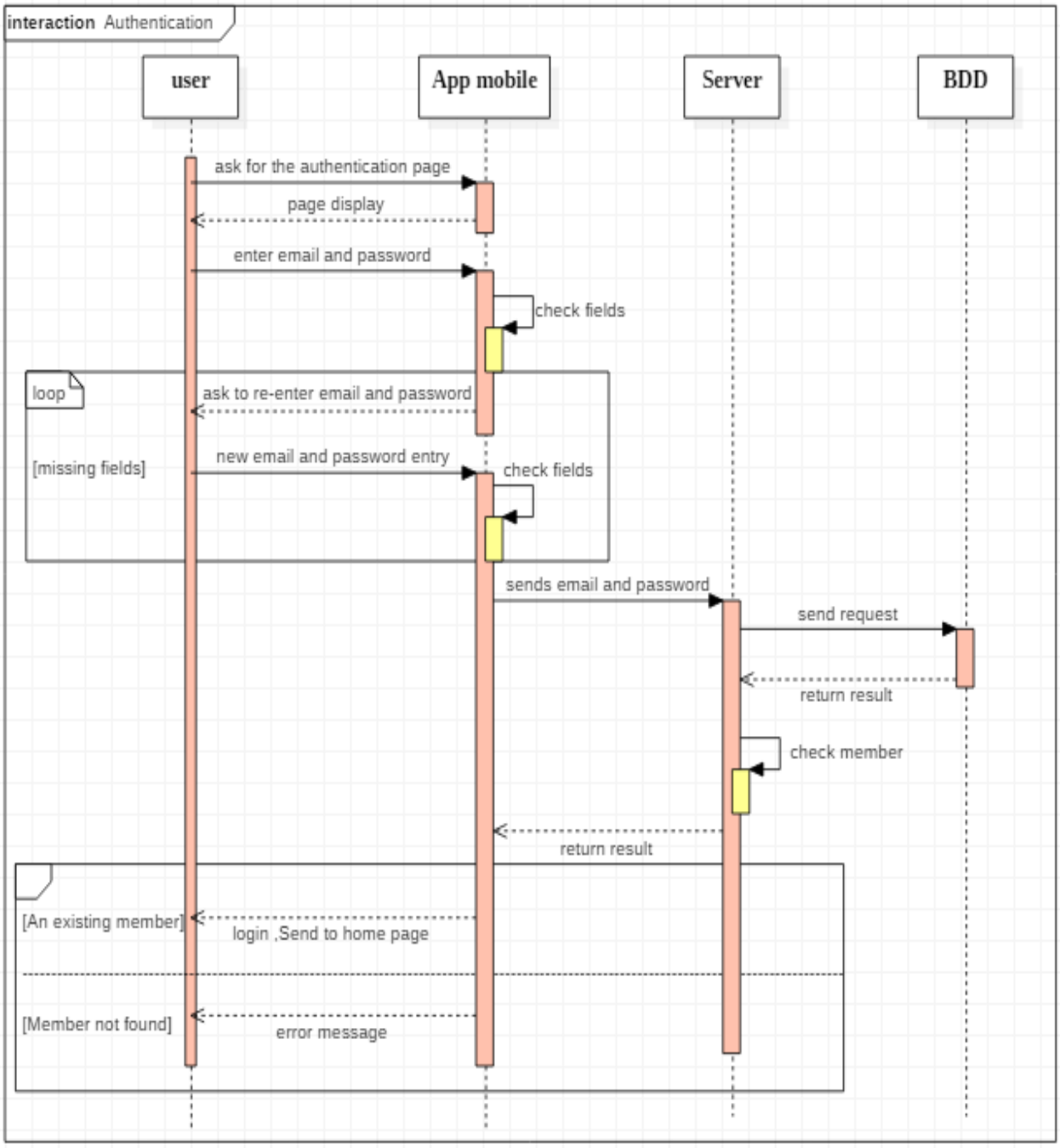

Figure 2.4. Sequence diagram for authentication.

2.3.5.2 Use case Add a profile/medical file**:**

The scenario of adding a medical file/profile is done according to the chronology represented by the 2.5.

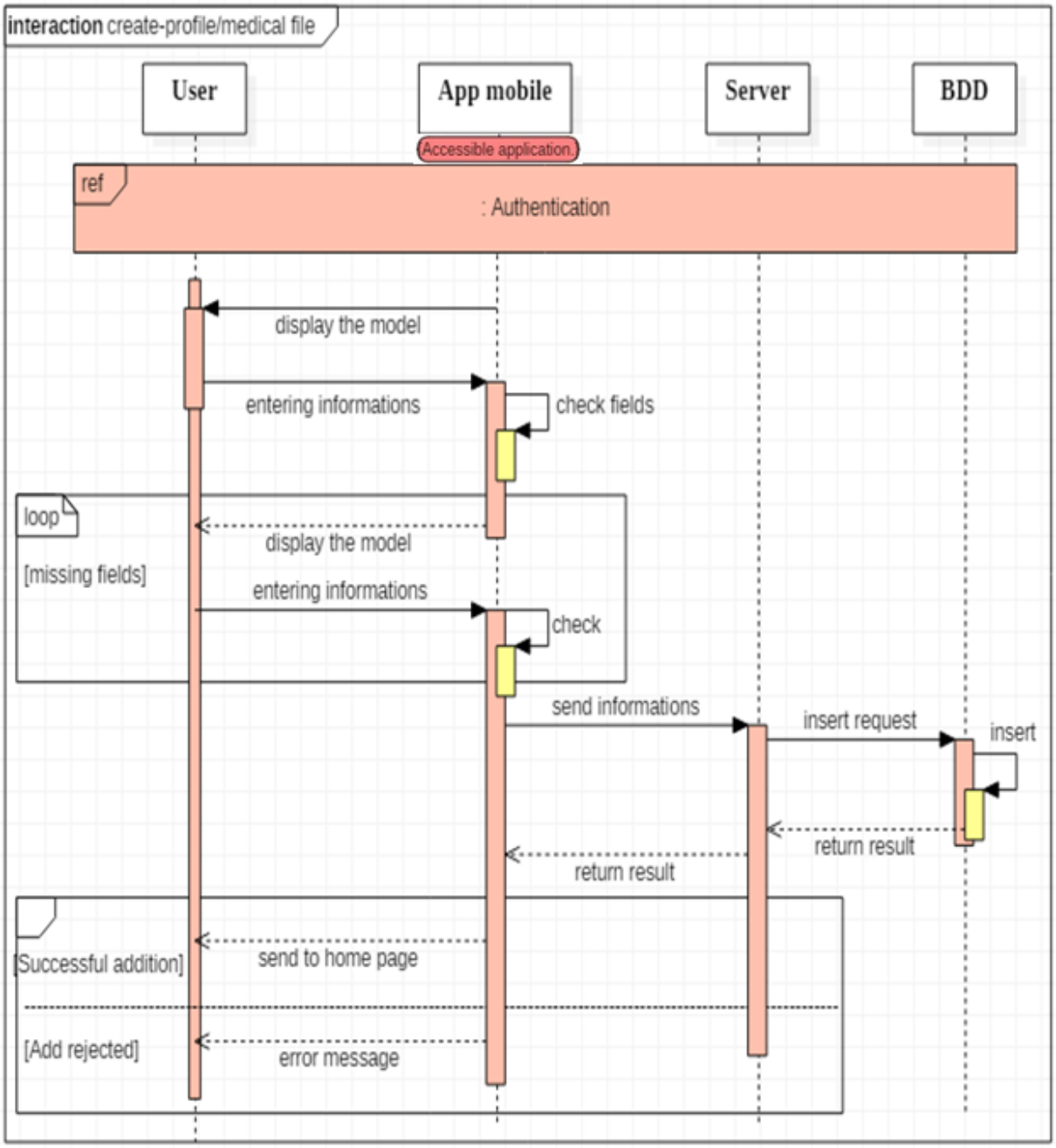

Figure 2.5. Sequence diagram for add profile/medical file.

2.3.5.3. Use case SOS request**:**

The scenario of SOS request is done according to the chronology represented by the 2.6.

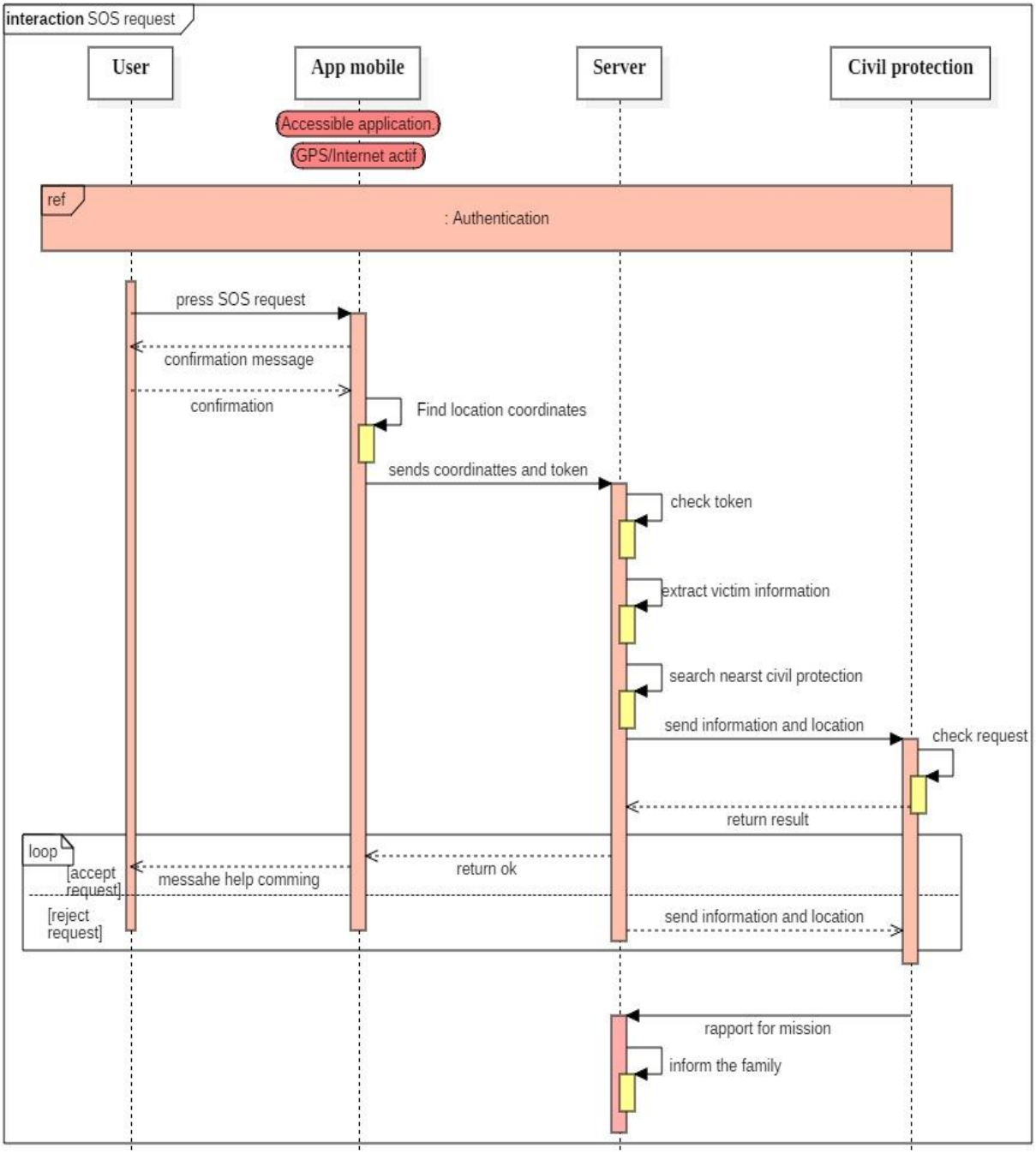

Figure 2.6. Sequence diagram for SOS request.

2.3.5.4. Use case Blood request:

The scenario of Blood request is done according to the chronology represented by the 2.7.

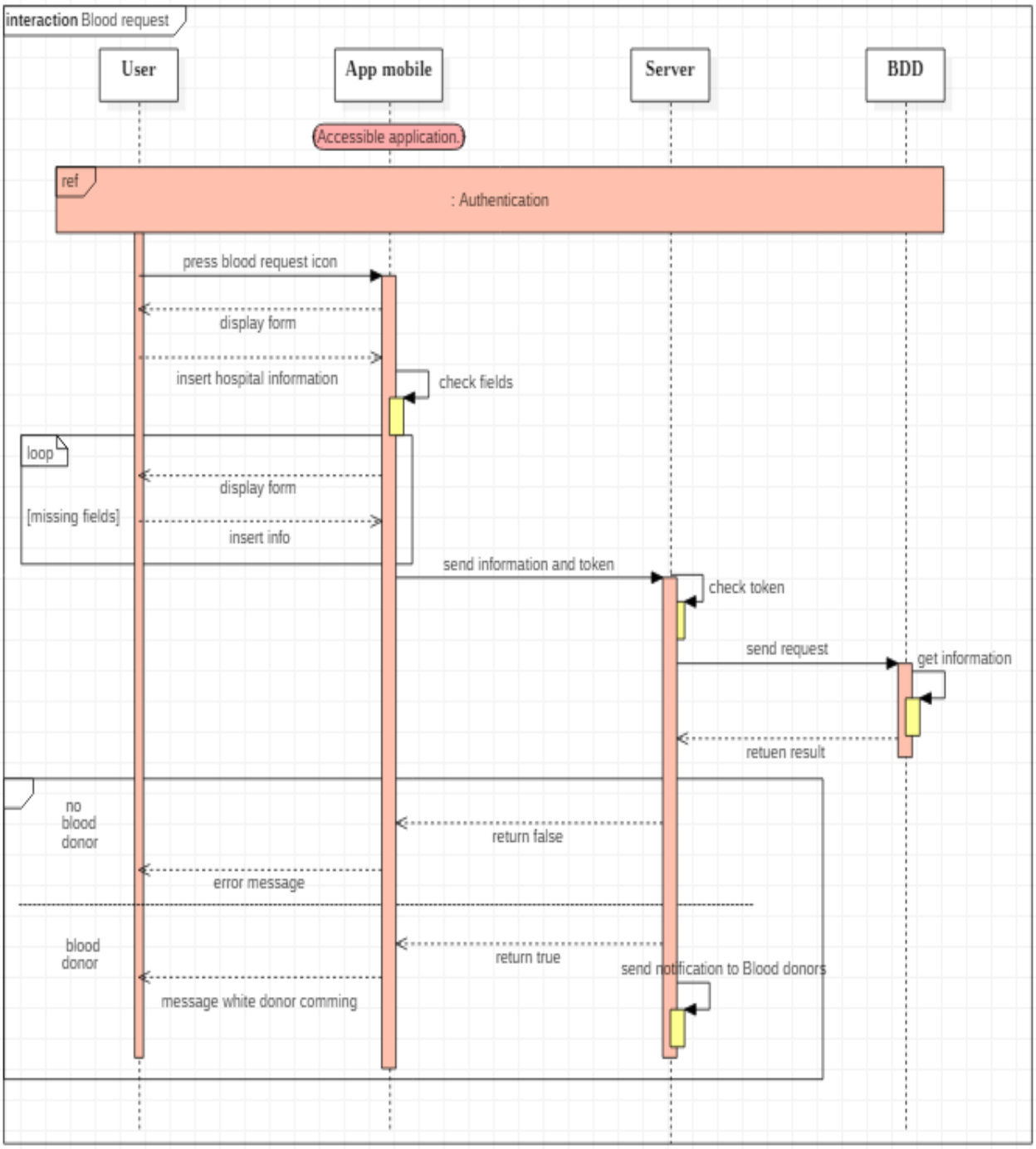

Figure 2.7. Sequence diagram for Blood request.

2.3.5.5. Use case find a hospital**:**

The scenario of Blood request is done according to the chronology represented by the 2.8.

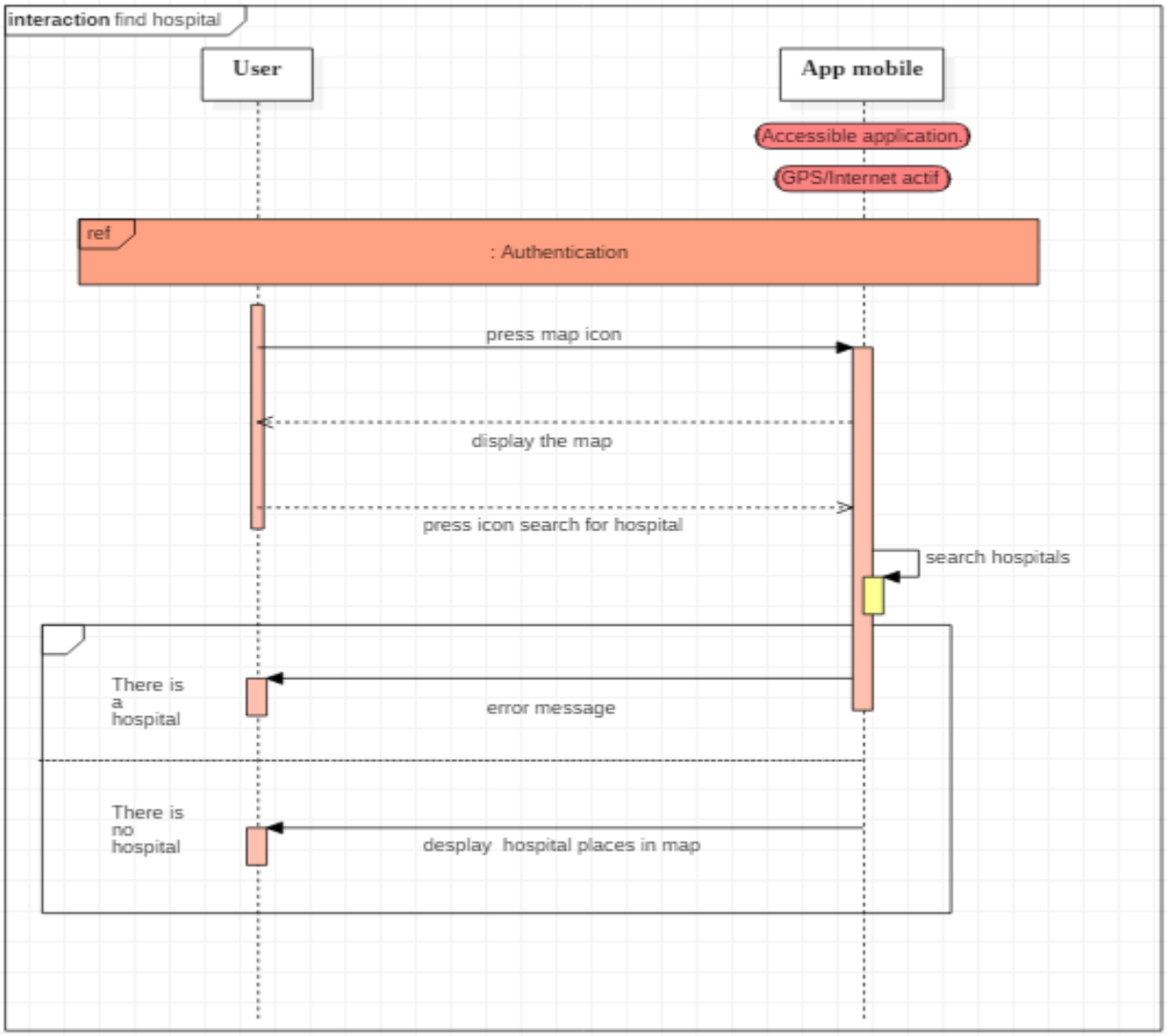

Figure 2.8. Sequence diagram for find hospitals.

2.4. Class Diagram:

A class diagram is defined as a set of classes containing attributes and operations, connected to each other by relations and have conditions of participation (cardinalities). for this diagram we are using UML Class Diagram. The class diagram of our application is shown in Figure 2.9.

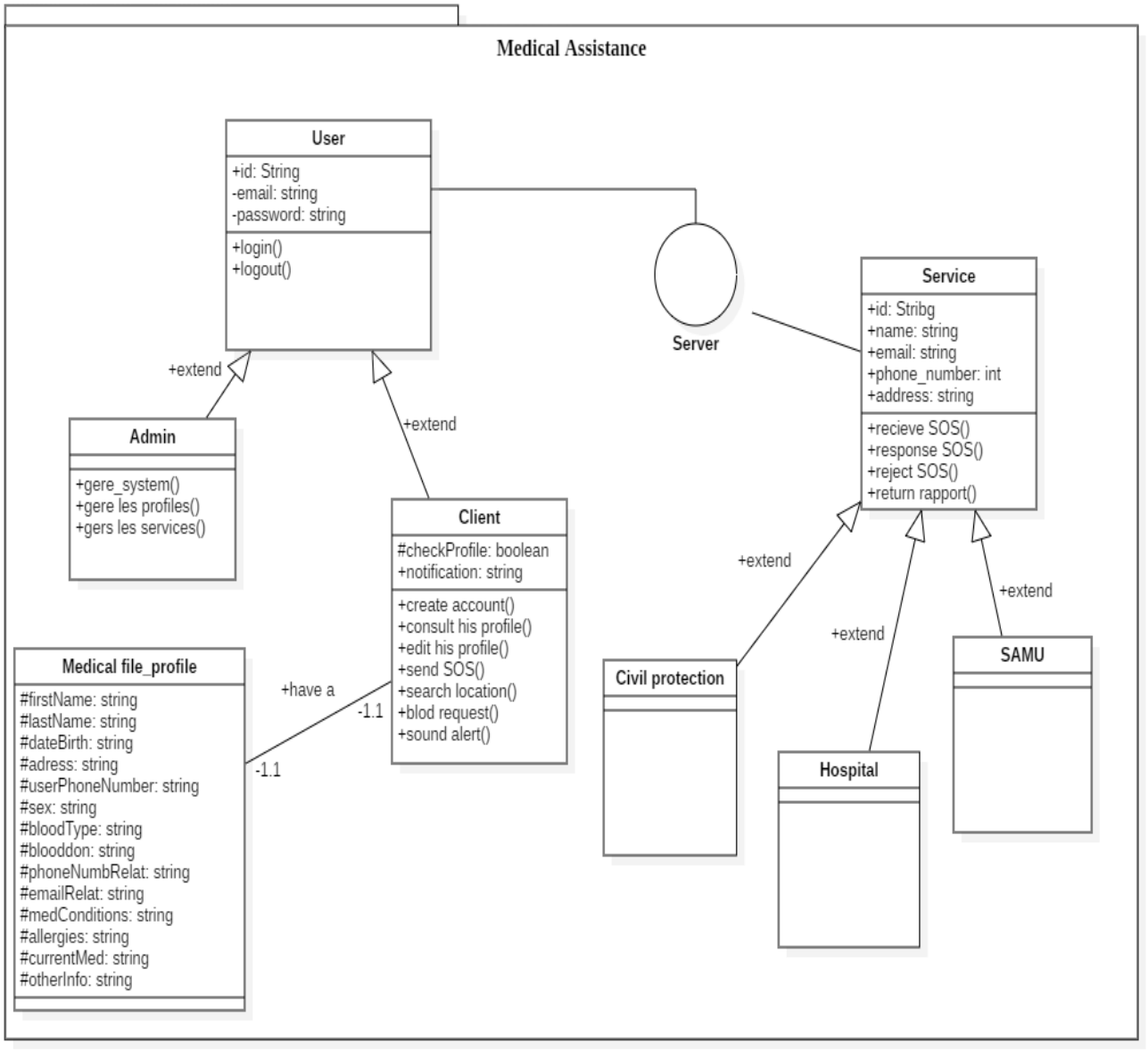

Figure 2.9. class diagram our application.

2.5. Conclusion:

In this chapter, we describe usage cases in detail by inserting in a textual manner all interactions between actors and the system. We have completed this textual description with a graphical representation of UML, we also represented the database by class diagram. Finally, this chapter has allowed us to prepare the phase of realization that will concretize everything that has been presented so far.

CHAPTER THREE:

Implementation

3.1. Introduction:

This chapter contains a description of the development tools used by the server and the client. It includes an explanation of the usage of the application with some images and services available in the application. We speak a bit about the database and also about the technical side and the security aspect of the application. We have chosen open source development tools to avoid legal aspects, contracts, licenses and links. Thus, reducing costs.

3.2. Development environment:

We define the development environment hardware and software.

3.2.1. Hardware environment:

For the realization of our project, we used a HP computer characterized by:

- Operating system: Windows 10.
- Processor: Intel ® Core™ i5-3230M CPU @ 2.60 GHz.
- RAM: 8 GB.
- Hard Disk: 500 GB

In order to install and run the application we used a Samsung smart phone with:

- Name of the device: Samsung Galaxy J5 Prime.
- Android version: 8.0.0
- Connection: LTE, 3G, 2G.
- Wi-Fi: 802.11a.
- GPS: GPS.
- RAM: 2 GB
- Internal memory: 16 GB

3.2.2. Software environment:

In this section, we define the development environment and the most important libraries used by the server and then by the client.

3.2.2.1. On the server side:

we define the development environment and the most important libraries used by the server side.

- REST API:

The principle of REST is to use HTTP for the implementation of a Web Service, not only as a simple transport protocol, but also to define the API of each service, the same definition of messages between clients and server. To interact with the MongoDB database, we need to build a REST API first. An API performs the walk described below. Accept requests in the GET or POST methods. Interact with the database by inserting or retrieving data. Finally give a reply back to JSON format.

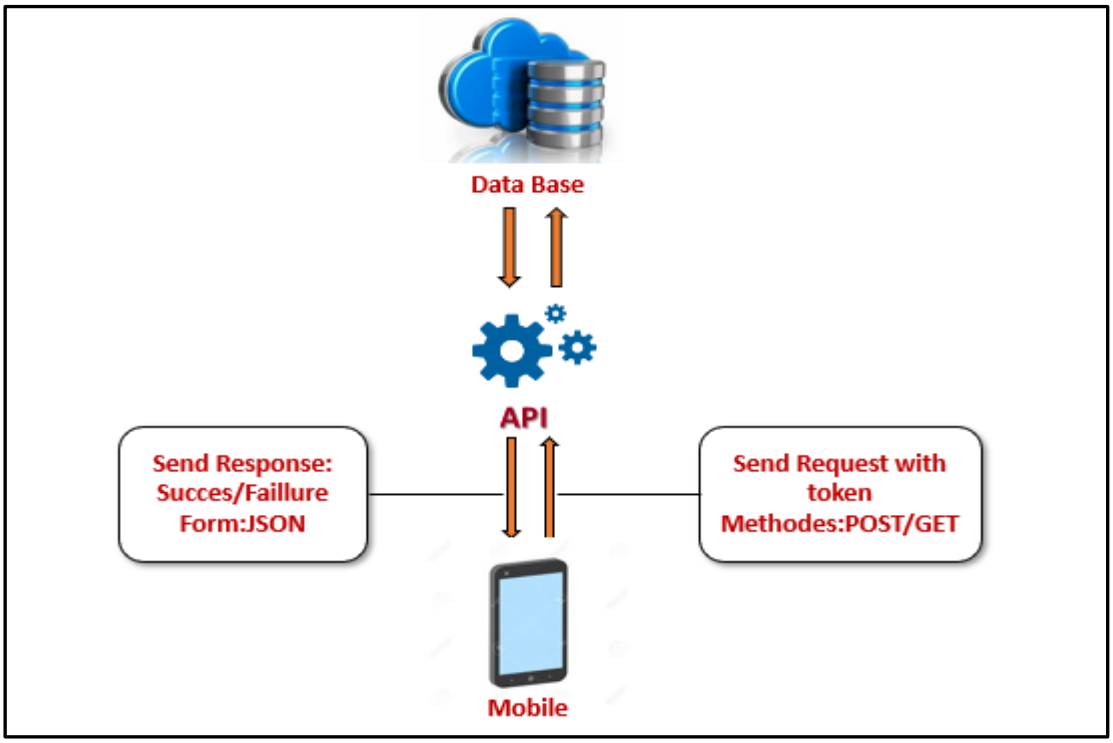

Figure 3.1. REST API.

3.2.2.1.1.Node.js:

Node.js is an open-source and cross-platform JavaScript runtime environment. It is a popular tool for almost any kind of project, Node.js runs the V8 JavaScript engine, the core of Google Chrome, outside of the browser. This allows Node.js to be very performant [9].

3.2.2.2.2. Used libraries:

The libraries we have used in our project are all licensed Open Source.

- Express:

Express.js is a Node JS web application server framework, which is specifically designed for building single-page, multi-page, and hybrid web applications. It has become the standard server framework for node.js [9].

```
constexpress = require("express");
// Init App
constapp = express();

constrouter = express.Router();
```

- Mongoose:

Mongoose is an Object Document Mapper (ODM). This means that Mongoose allows you to define objects with a strongly-typed schema that is mapped to a MongoDB document.

```
constmongoose = require("mongoose");

// Mongoose connection to DB
mongoose.set("useCreateIndex", true);
mongoose.connect("mongodb://localhost/medicalAssistant", {
useNewUrlParser:true
});
letdb = mongoose.connection;
```

- Nodemailer:

library used for sending email.

```
constnodemailer = require("nodemailer");
lettransporter = nodemailer.createTransport({
```

- Bcrypt:

The bcrypt library on NPM makes it really easy to hash and compare passwords in Node. If you're coming from a PHP background, these are roughly equivalent to password_hash () and password_verify ().

```
constbcrypt = require("bcrypt");
// hash user password before saving into database
bcrypt.hash(req.body.userPassword, saltRounds, (err, hash) => {
// function for compare the enter password with password hashing
bcrypt.compare( req.body.userPassword,newUser.userPassword,(err, result) => {
```

- Jsonwebtoken:

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object. This information can be verified and trusted because it is digitally signed. JWTs can be signed using a secret (with the HMAC algorithm) or a public/private key pair using RSA or ECDSA. Although JWTs can be encrypted to also provide secrecy between parties, we will focus on signed tokens. Signed tokens can verify the integrity of the claims contained within it, while encrypted tokens hide those claims from other parties. When tokens are signed using public/private key pairs, the signature also certifies that only the party holding the private key is the one that signed it [10].

```
constjwt = require("jsonwebtoken");
constJwtKey = "secret";
// Create a token with expration in 1h
consttoken = jwt.sign(
{ userEmail:newUser.userEmail,
userId:newUser._id,checkProfile:newUser.checkProfile
},JwtKey,
{ expiresIn:"1h" }
            );
// verify the token received in header
constdecoded = jwt.verify(token, JwtKey, null);
req.userData = decoded;
```

- Body-parser:

Body-Parser is a library that you can use as middleware when handling Node.js GET and POST requests. This module provides the following parsers: JSON, Raw, text and URL-encoded. It essentially makes it easier for us to access the Request Body objects, especially when it comes to our POST requests.

```
// Body Parser Middleware
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended:false }));
```

- Validator:

According to the official website, Express Validator is a set of Express.js middleware that wraps validator.js, a library that provides validator and sanitizer functions. Simply said,

Express Validator is an Express middleware library that you can incorporate in your apps for server-side data validation [11].

3.2.2.2. On the client side:

we define the development environment and the most important libraries used by the client side.

3.2.2.2.1. Android Studio:

Android Studio is the official Integrated Development Environment (IDE) for Android app development, based on IntelliJ IDEA. On top of IntelliJ's powerful code editor and developer tools, Android Studio offers even more features that enhance your productivity when building Android apps, such as: [12]

- A flexible Gradle-based build system
- A fast and feature-rich emulator
- A unified environment where you can develop for all Android devices
- Instant Run to push changes to your running app without building a new APK
- Code templates and GitHub integration to help you build common app features and import sample code
- Extensive testing tools and frameworks
- Lint tools to catch performance, usability, version compatibility, and other problems
- C++ and NDK support

Android studio works with IntelliJ IDEA:

IntelliJ IDEA is a special programming environment or integrated development environment (IDE) largely meant for Java. This environment is used especially for the development of programs. It is developed by a company called JetBrains, which was formally called IntelliJ. It is available in two editions: The Community Edition which is licensed by Apache 2.0, and a commercial edition known as the Ultimate Edition. Both of them can be used for creating software which can be sold. What makes IntelliJ IDEA so different from its counterparts is its ease of use, flexibility and its solid design [13].

3.2.2.2.3. Used libraries:

The libraries we have used in our project are all licensed Open Source.

- Material design:

Material design is a comprehensive guide for visual, motion, and interaction design across platforms and devices. To use material design in your Android apps, follow the guidelines defined in the material design specification and use the new components and styles available in the material design support library. This page provides an overview of the patterns and APIs you should use. Android provides the following features to help you build material design apps:

- A material design app theme to style all your UI widgets
- Widgets for complex views such as lists and cards
- New APIs for custom shadows and animations

- Google Maps Api:

Service allowing virtual geo-tracking thanks to the built-in GPS of the device which allows to remotely monitor the position and movement of an object and take action if the positioner the displacement deviates from certain values fixed in advance.

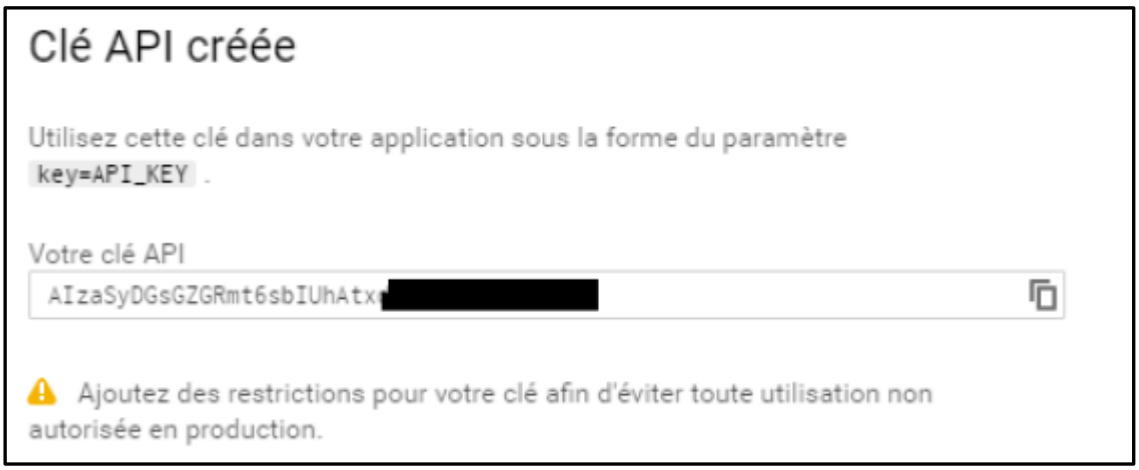

Figure 3.2. Getting Google Maps API Key.

- Retrofit 2:

Retrofit is a type-safe HTTP client for Android and Java. Retrofit makes it easy to connect to a REST web service by translating the API into Java interfaces. This powerful library makes it easy to consume JSON or XML data which is then parsed into Plain Old Java Objects (POJOs). GET, POST, PUT, PATCH, and DELETE requests can all be executed [14].

3.2.3. Permissions of application:

```
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION_" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="com.google.android.providers.gsf.permisson.READ_GSERVICES" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.BIND_NOTIFICATION_LISTENER_SERVICE" />
```

Figure 3.3. Permissions of application.

3.3. Implementing the database:

In order to create the database, we used Mongo dB which uses NOSQL language.

- MongoDB:

MongoDB is an open source database management system (DBMS) that uses a document-oriented database model which supports various forms of data. It is one of numerous nonrelational database technologies which arose in the mid-2000s under the NoSQL banner for use in big data applications and other processing jobs involving data that doesn't fit well in a rigid relational model. Instead of using tables and rows as in relational databases, the MongoDB architecture is made up of collections and documents [15].

- How MongoDB works [15]:

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JavaScript Object Notation objects but use a variant called Binary JSON (BSON) that accommodates more data types. The fields in documents are akin to the columns in a relational database, and the values they contain can be a variety of data types, including other documents, arrays and arrays of documents, according to the MongoDB user manual.

- How Server connect with MongoDB:

```
// Mongoose connection to DB
mongoose.set("useCreateIndex", true);
mongoose.connect("mongodb://localhost/medicalAssistant", {
useNewUrlParser:true
});
letdb = mongoose.connection;

// Check Connection
db.once("open", function() {
console.log("Connected to MongoDB");
});

// Check for DB errors
db.on("error", function(err) {
console.log("Mongo DB err:" + err);
});
```

- Our Data Base :

```
> show dbs
admin             0.000GB
config            0.000GB
local             0.000GB
medicalAssistant  0.000GB
> use medicalAssistant
switched to db medicalAssistant
> show collections
profiles
services
users
> db.users.find().pretty()
{
        "_id" : ObjectId("5ce980b9cc1f4e1ec4a0b9b3"),
        "checkProfile" : true,
        "userEmail" : "youcefzahar1995@gmail.com",
        "userPassword" : "$2b$10$9nZppdW5eIpmtEM9BMtXEOqJ.lZ16jWwVBnwnObwughXdL44U8yf2",
        "__v" : 0
}
{
        "_id" : ObjectId("5cebf7254553ba183874e7ed"),
        "checkProfile" : false,
        "userEmail" : "aissaboudr@gmail.com",
        "userPassword" : "$2b$10$i1WetlAfsom6zWL1ZindUuoO.sPDU.Klun2oPaK4r54AuNzSaCWBK",
        "__v" : 0
}
{
        "_id" : ObjectId("5cebf7304553ba183874e7ee"),
        "checkProfile" : false,
        "userEmail" : "amar.nadir28@gmail.com",
        "userPassword" : "$2b$10$rQdB/E2kHcLcYC5DRFnSzeUIlGdUOHk0VUvZ248ot4Ay4Es/VZVJS",
        "__v" : 0
}
{
        "_id" : ObjectId("5cebfa146969b415781a00d6"),
        "checkProfile" : true,
        "userEmail" : "younesgara@hotmail.com",
        "userPassword" : "$2b$10$AbZXN0LHfjhBTPVjFraqfOiB5RNDZ/Uu8wQFOtTZ5SSRkP0KKwkXG",
        "__v" : 0
}
```

3.4. Presentation of the application:

this part of the chapter to the presentation of the main interfaces of our mobile application.

3.4.1. Tree view of our application:

To illustrate the menu offered by our application, Figure 3.4 shows graphically the tree structure of its features.

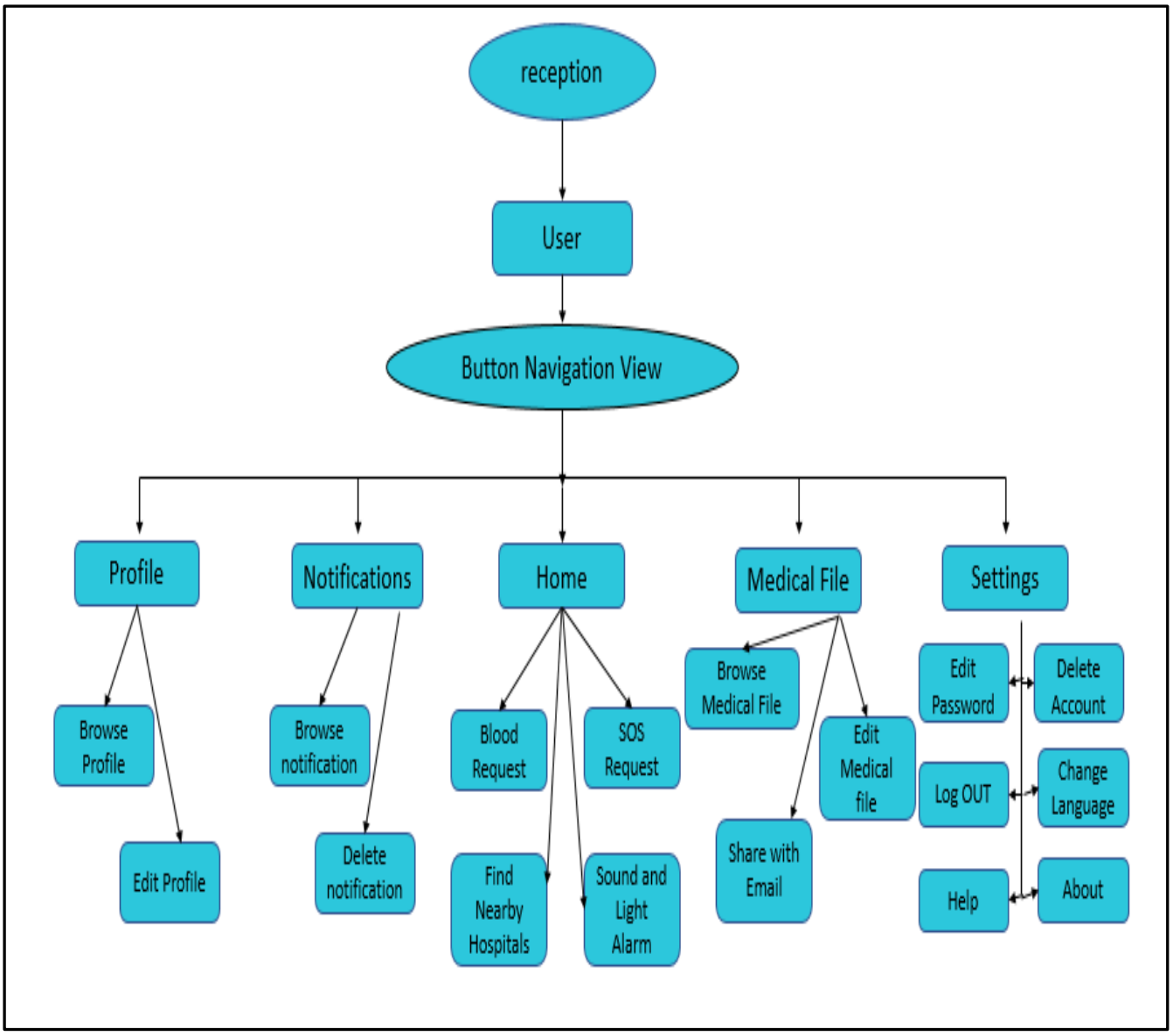

Figure 3.4. Tree view of our application.

3.4.2. Presentation of the interfaces:

In this chapter we will provide all interfaces of our application. Figure3.5 below illustrates the Splash Interface containing the application logo with a loading animation. This page lasts up to two seconds.

Figure 3.5. Splash Interface.

Figure 3.6. Icon of application.

When the splash screen finishes, the application start interface comes in. There are two buttons to browse for usage and the other for the beginning of use. When you press the OK button, the Login and Register interface appear. Figure 3.7 below illustrates the Front interface.

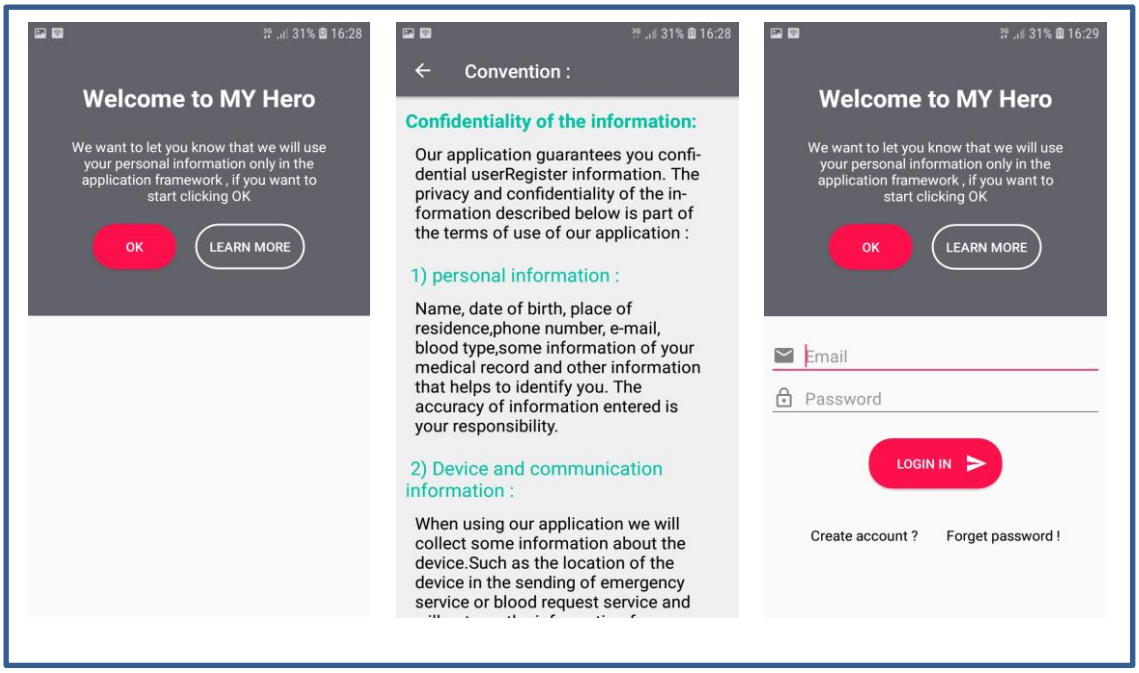

Figure 3.7.the Front interface.

For the first time the user chooses the interface to create a new account. When the information is sent, the account creation result appears. Figure 3.8 below illustrates create a new account interface.

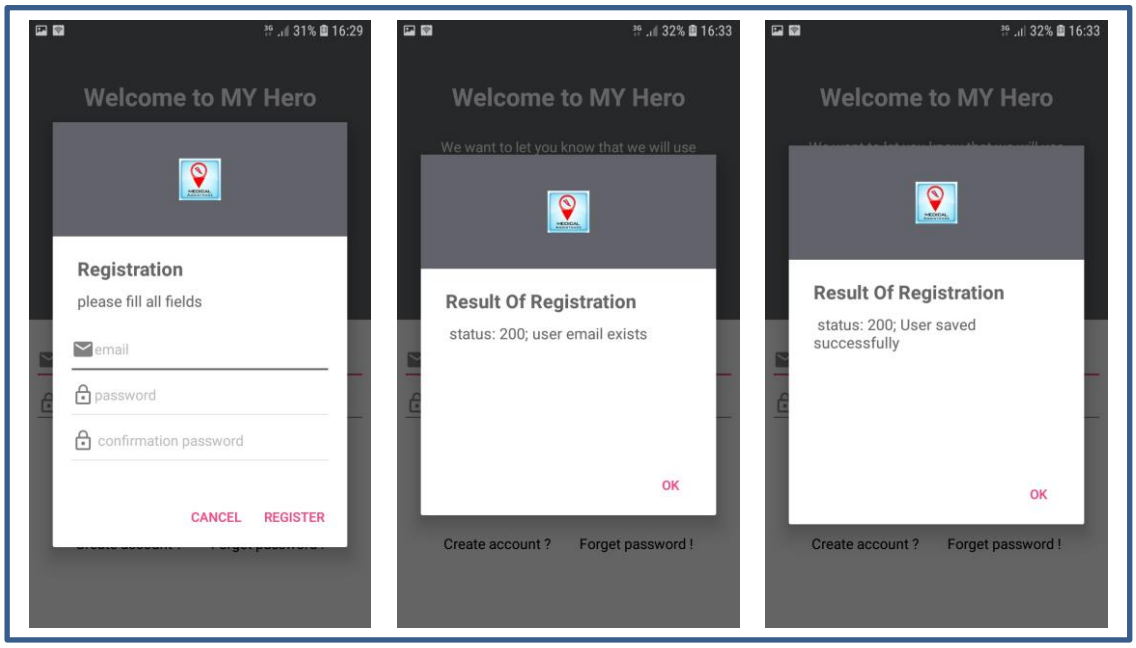

Figure 3.8. Registration interface.

After creating a new account directs the user to create profile and medical file, we searched for information related to the medical file, such as the state of health, allergies, diseases and drugs used. The list was as follows:

- Medical conditions :(ASTHMA, ARTHRITIS, CANCER, ALZHEIMER, DIABETES, Chronic Kidney, Smoke).
- Allergies:(Drug Allergies, Food Allergy, Skin Allergy, Dust Allergy, Eye Allergy, Latex Allergy, Sinus Infection, Mold Allergy).
- Current medications:(Synthroid, Crestor, Ventolin, Nexium, Advair Diskus, Lantus Solostar, Vyvanse, Lyrica, Spiriva Handihaler, Januvia).

Figure 3.9 below illustrates create a profile/medical file interface.

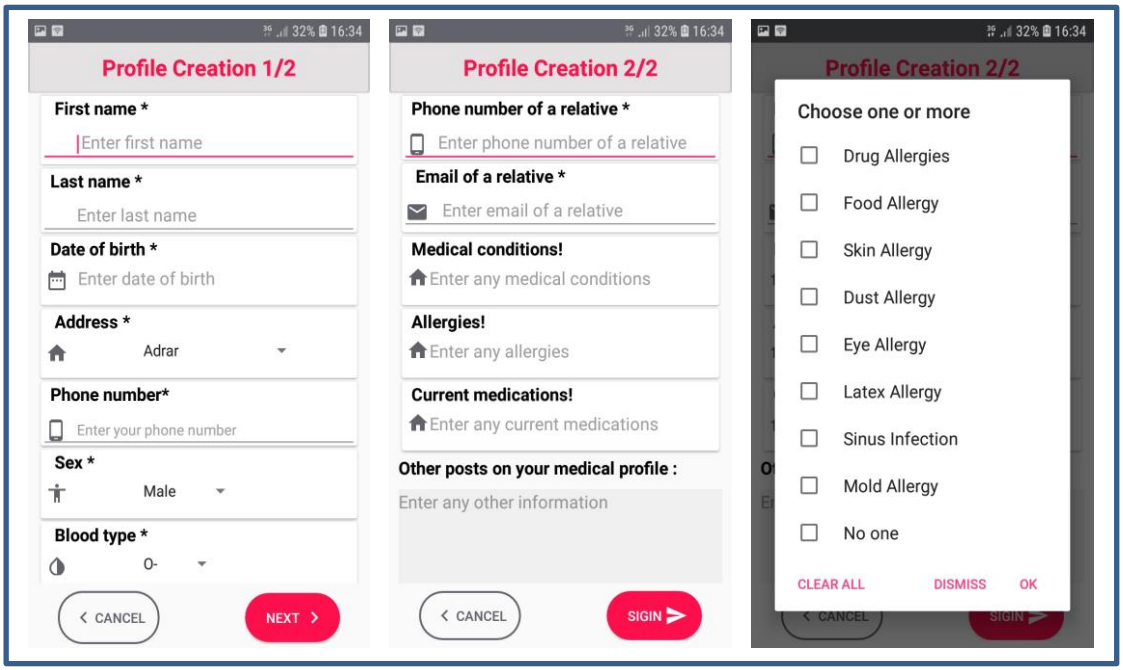

Figure 3.9. Create profile/medical file interface.

The user can see his profile and edit it. Figure 3.10 below illustrates profile interface.

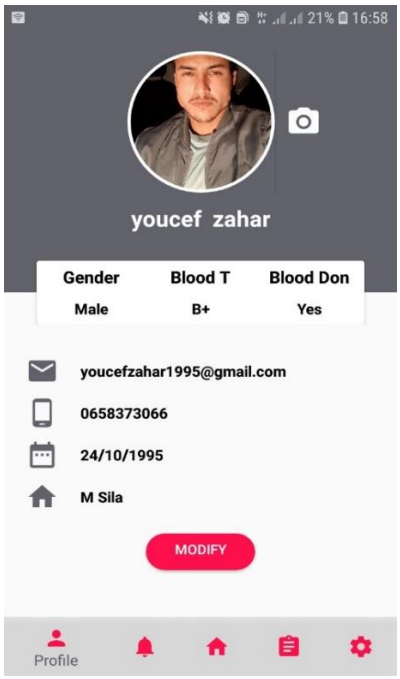

Figure 3.10. profile interface.

When the alerts arrive, the user can see and delete them, Figure 3.11 below illustrates notifications interface.

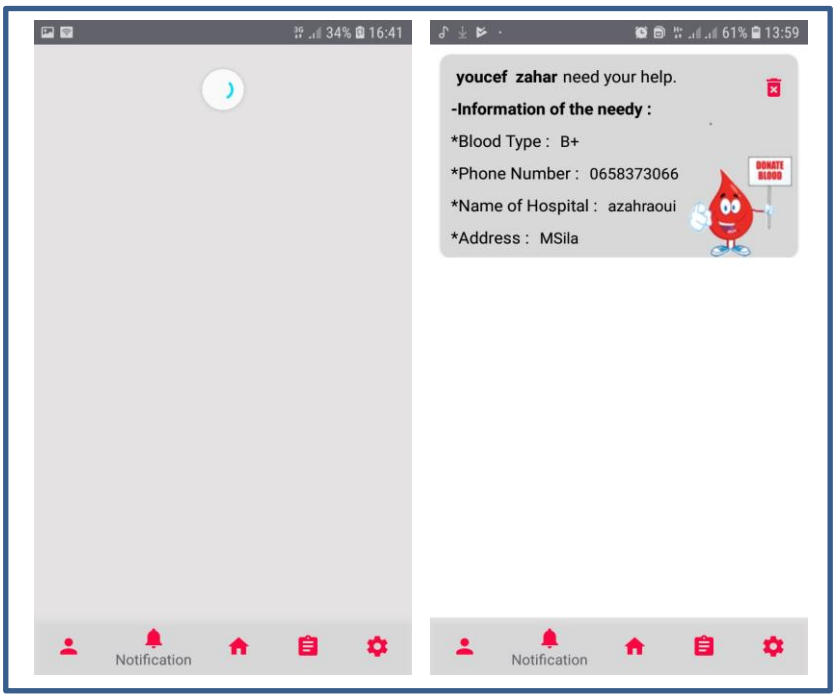

Figure 3.11. notifications interface.

The user can view, change and share his / her medical profile via email. Figure 3.12 below illustrates medical file interface.

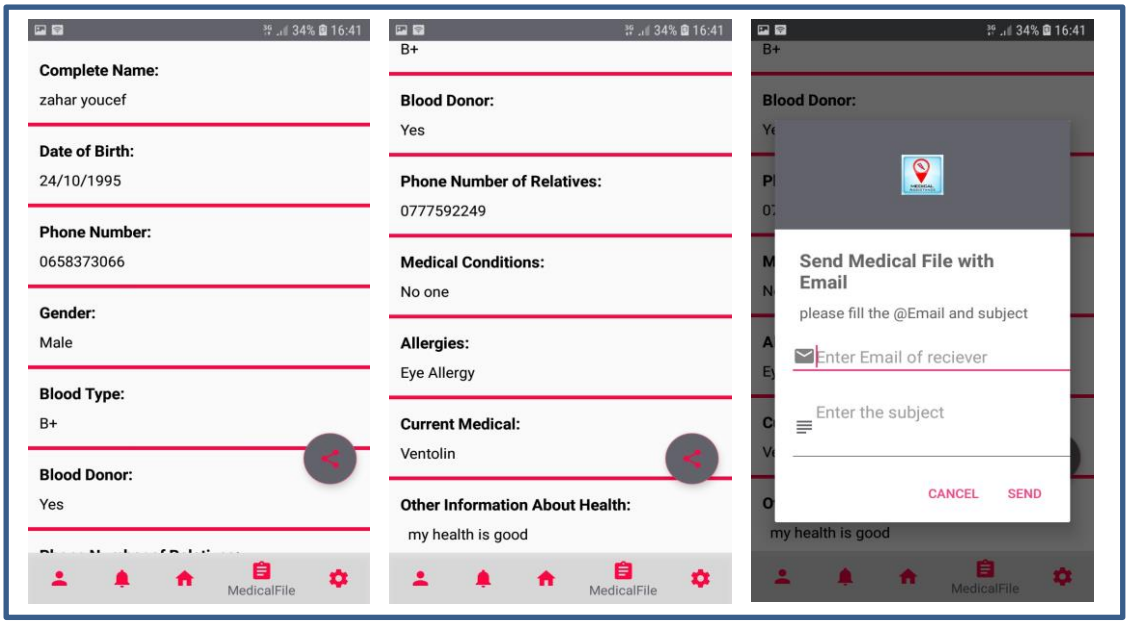

Figure 3.12. medical file interface.

All application settings are located on the settings page where the user can change the password and language and delete the account or exit the application or request information. Figure 3.13 below illustrates settings interface.

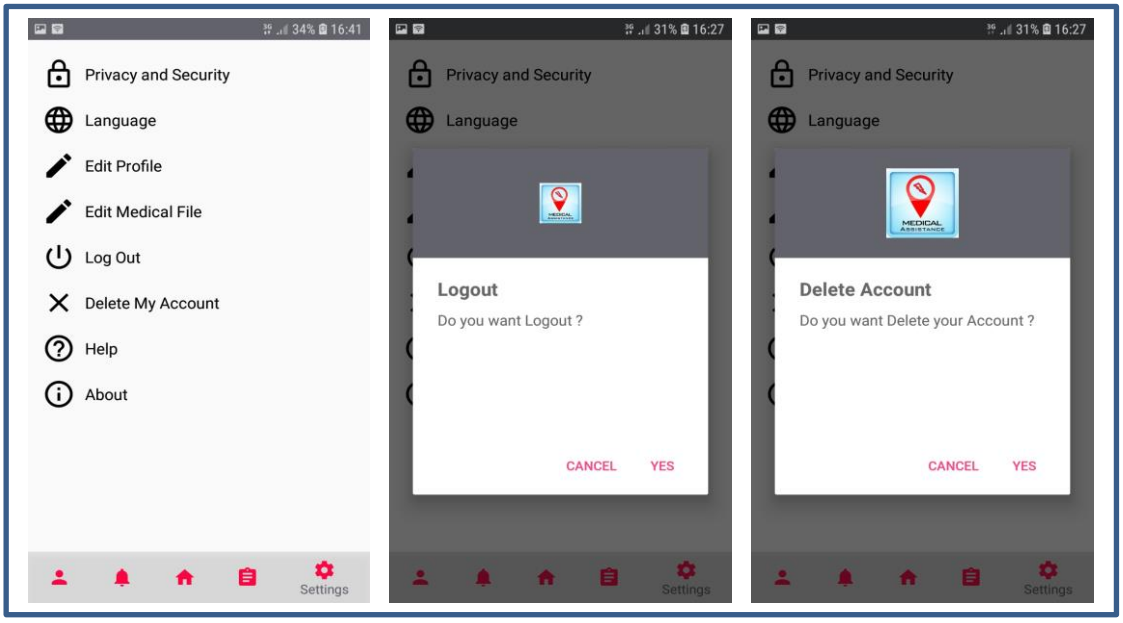

Figure 3.13. Settings interface.

The main services for the app are located on the Home screen. we have a quick SOS application where one-click application extracts location information and sends it to the server. We have a search service for the nearest hospitals. We also have a blood donation

service from donors where the user fills the name of the hospital and the place and sends the information to the server. The request is sent to the donors on the notices page we mentioned earlier There is also a voice and light alarm service for the dumb and to indicate the need for help. Figures 3.14 and 3.15 and 3.16 and 3.17 below illustrates Home interface.

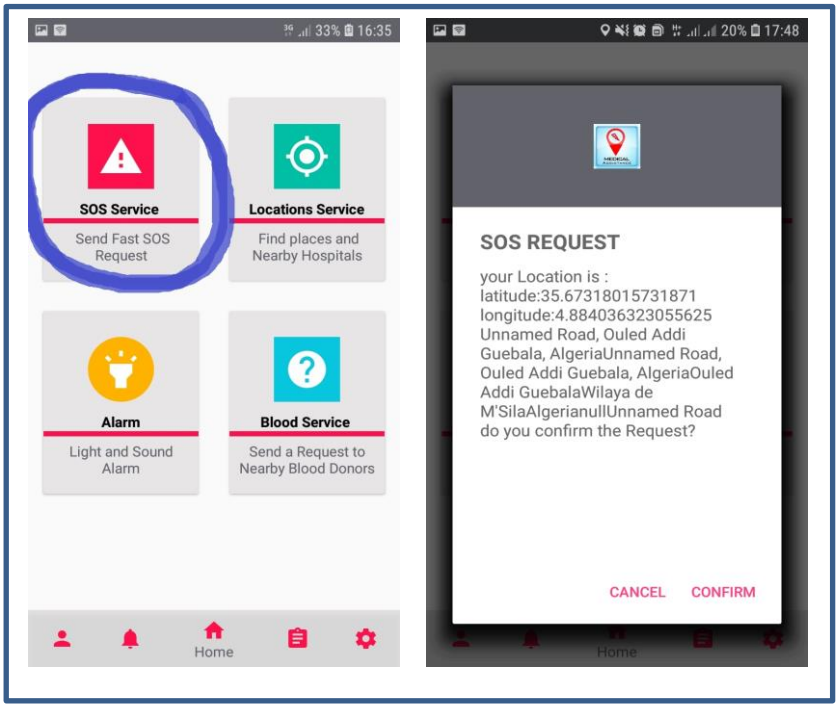

Figure 3.14.SOS Request interface.

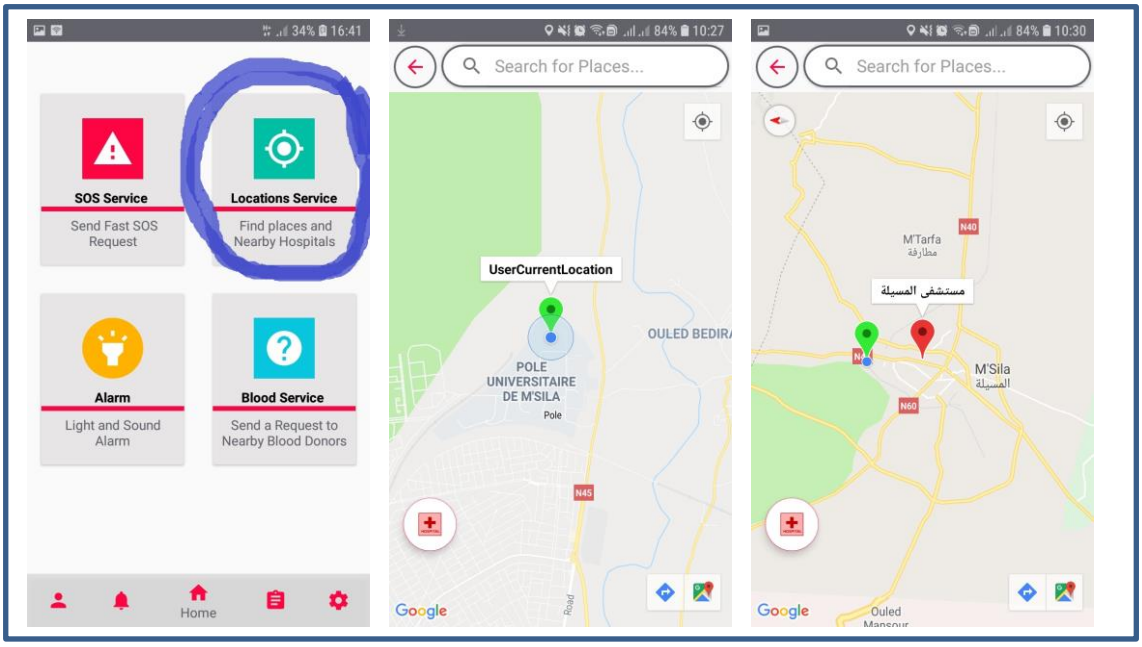

Figure 3.15. Location and Nearby Hospitals interface.

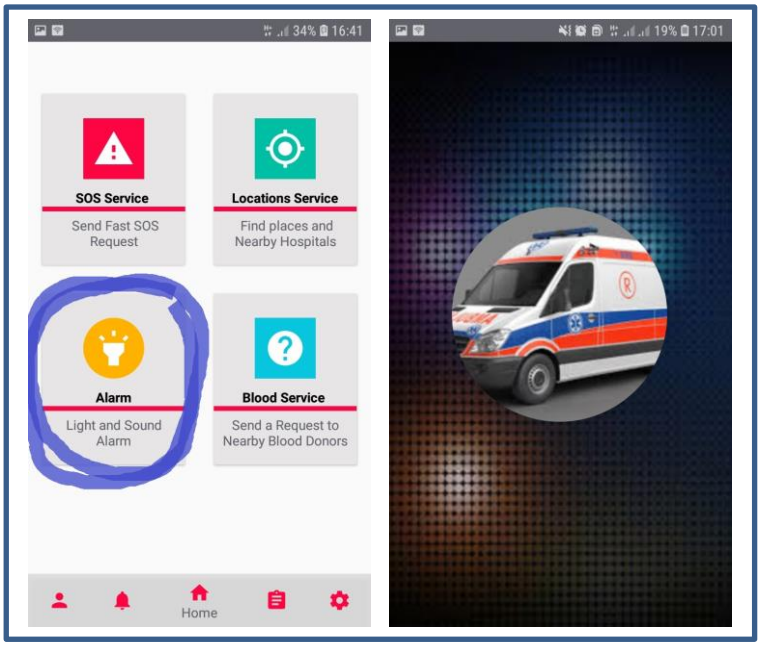

Figure 3.16. Alarm with sound and light interface.

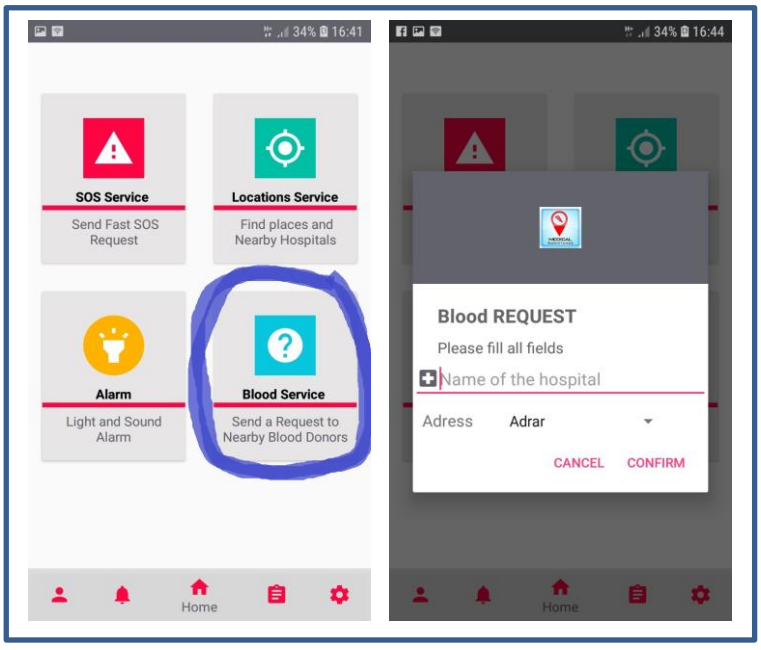

Figure 3.17. Blood Request interface.

3.4.3. Security:

A number of precautions have been taken to ensure the application of the system and the preservation of information in the database. The system isolates the application of the smart phone from direct connection to the database, making it difficult to penetrate the database. we control what the user enters into the fields to avoid any injection. In terms of user definition, we use token to define and to use the application where all requests are monitored and rejected without a valid token. To increase token strength, we have made it repealed every hour and is renewed. Figures 3.18below illustrates token form.


```
{
    "message": "Auth successful",
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJ1c2VyRW1haWwiOiJ5b3VjZWZ6YWhhcjE5OTVAZ21haWwuY29tIiwidXNlck1kIjoiNWNlOTgwYjljYzFmNGUxZWM0YTBiOWIzIiwiY2hlY2tQcm9maWx1Ijp0cnV1LCJpYXQiO
        jE1NjAyODc4MjAsImV4cCI6MTU2MDI5MTQyMH0.CyRIMIRpxiceEiurIzDfd5HXPuiCmPopiH8fYJdL_DI",
    "checkProfile": true
}
```


Figure 3.18. Token form.

3.5. Conclusion:

In this last part of our work, we introduced the development environment and the main tools used, as well as the most important frameworks, which allowed us to achieve our application as we explained the tree that represents the course of the application. We spoke about the interfaces with the explanation of pictures and finally we mentioned the security component and methods used.

General Conclusion

We began this note as a general description of the applications of smart mobile initially applications and types of platforms. Then we put in question the most important problems in the field of medical assistance. We sealed it with a set of solutions in the form of a mobile application connected to a server.In chapter II of the note we have identified the main specifications and features of the application to be achieved Then we introduced the construction stages where we began analyzing needs, we then modeled the use cases and specific functions of the end-user needs of our system using UML form .Either in the third part we moved to the practical side and is part of the mobile application realization that works on the android platform and the realization of the NOSQL data base found in the cloud and create a server that runs the application with the database and we also put pictures of most of the application screens with explanation.

The subject we worked on was good and useful to people's lives. We were very interested in researching and developing our knowledge, we learned how to exploit our gains, with the end of our time we have not reached our goals 100% for the absence of some techniques in the field of medicine in our country and the absence of legal space in the technological aspect. Also, the time is not enough to add an SOS request if the phone is locked, we will seek to add this property and conduct studies to add other characteristics to our application in the future.

Bibliography

[1] https://whatis.techtarget.com/definition/mobile-app , viewed the 09/03/2019.

[2] https://www.ecommerce-nation.com/various-categories-types-of-mobile-applications/ , viewed the 09/03/2019.

[3] International Journal of Engineering & Technology IJET-IJENS Vol:10 No:06: Mobile Application and Its Global Impact: http://www.ijens.org/107506-0909%20ijet-ijens.pdf , viewed the 09/03/2019.

[4] https://www.shoutmeloud.com/top-mobile-os-overview.html, viewed the 13/03/2019.

[5] https://www.computerhope.com/jargon/a/android.htm , viewed the 13/03/2019.

[6] https://www.geeksforgeeks.org/android-system-architecture/ , viewed the 13/03/2019.

[8] https://lexington.wakehealth.edu/What-Is-a-Medical-Emergency.htm/ , viewed the 06/05/2019.

[9] https://nodejs.org/en/docs/meta/topics/dependencies/ ,viewed the 09/06/2019.

[10] https://jwt.io/introduction/, viewed the 09/06/2019.

[11] https://auth0.com/blog/express-validator-tutorial/, viewed the 09/06/2019.

[12] https://developer.android.com/studio/intro , viewed the 09/06/2019.

[13] https://www.techopedia.com/definition/7755/intellij-idea , viewed the 09/06/2019.

[14] https://code.tutsplus.com/tutorials/getting-started-with-retrofit-2--cms-27792 , viewed the 09/06/2019.

[15] https://searchdatamanagement.techtarget.com/definition/MongoDB , viewed the 09/06/2019

Abstract

This dissertation is designed to create a mobile application for medical Assistance In case of danger, Technology is constantly evolving and fast, which is what we need to apply in the medical field and to benefit from it in order to raise the level of medical care. To achieve this goal, we propose designing and building a mobile application on the Android platform that will greatly help to raise the level of medical Assistance and avoid human casualties. The application connects to a server and a database in the cloud, in order to provide various services to the victim, where the user is allowed to apply for a fast SOS service, locate places and nearby hospitals. The application provides the profile and medical file for medical Assistance and facilitates the work of paramedics and doctors, the victim's family will learn as soon as possible, the patient can request blood easily and quickly when needed. All these services are for the best health care. To do the work we used the UML design to ensure the integrity of our system, we used android studio to build the application, NodeJS to build the server, mongo dB to implement the database and some libraries to connect the system.

Keywords: Medical Emergency, Mobile Application, Android, Data Base, Client/Server Application, cloud.

ملخص

تم تصميم هذه الرسالة لإنشاء تطبيق محمول للمساعدة الطبية في حالة وجود خطر. تتطور التكنولوجيا بشكل متواصل وسريع، وهو ما نحتاج إلى تطبيقه في المجال الطبي والاستفادة منه من أجل رفع مستوى الرعاية الطبية. لتحقيق هذا الهدف، نقترح تصميم وبناء تطبيق جوال على نظام أندرويد من شأنه أن يساعد إلى حد كبير على رفع مستوى المساعدة الطبية وتجنب وقوع إصابات بشرية، يتصل التطبيق بخادم وقاعدة بيانات في السحابة، من أجل توفير خدمات متنوعة للضحية، حيث يُسمح للمستخدم بالتقدم للحصول على خدمة"SOS", وتحديد الأماكن والمستشفيات القريبة. يوفر التطبيق الملف الشخصي والملف الطبي للمساعدة الطبية ويسهل عمل المسعفين الطبيين والأطباء ، ستعلم عائلة الضحية في أقرب وقت ممكن عن وقوع الحادثة. يمكن للمريض طلب الدم بسهولة وبسرعة عند الحاجة, جميع هذه الخدمات هي للحصول على أفضل رعاية صحية, للقيام بالعمل استخدمنا تصميم "UML" لضمان سلامة نظامنا، استخدمنا "Android Studio" لبناء التطبيق،"Node JS" لبناء الخادم،و"Mongo DB" لتنفيذ قاعدة البيانات وبعض المكتبات لتوصيل النظام.

الكلمات المفتاحية : طب الطوارئ, تطبيق محمول, أندرويد, قاعدة البيانات, تطبيق خادم/ زبون, السحابة.