POLITECNICO DI TORINO

Master of Science in Computer Engineering

Master of Science's Degree Thesis

Development of a mobile application for patient monitoring after coronary heart diseases

Supervisors	Candidate
Prof. Monica Visintin	Mohamad Almoussa

ACADEMIC YEAR 2019-2020

Acknowledgements

According to a well known quote by Pauline Kael "When there is a will, There is way". That is true, but he never mentioned where the will comes from, not only from your inner self but also from the people that surrounds and loves you.

Before I start discussing this thesis, I want to thank all who helped me to start and finish it.

I would like to thank Professor Monica Visintin for her trust to let me work on this thesis and helping me through it from the beginning till the end.

I wish to express my deepest gratitude to Abinsula, the company that provided this thesis, on top of it Poalo Doz. A special thank to Sara Prete the one that stood by my side in this whole journey in downs before the ups that even a million thanks wont pay her back, the same goes for Gabriele, Silvia, Maria Chiara, Alberto, and the whole Abinsula team or in other words Family whose assistance was a milestone in the completion of this project in which it would be my pleasure to continue and work to offer the same help they gave to me.

I would like to share my love for my parents, Youssief and Nada, because no words can express how thankful I am. I am indebted to both of you for all the support, endless love, patients, sacrifices, encouragements, and above all trust that you gave me since day one. The only way to repay you, besides love and care, is by making you proud of me and believe me that is the main purpose of my life.

Thanks to Rola, Sally, Maya, and Ali, my beloved sisters and brother for being by my side in every step in my life supporting and encouraging to make the best of me. I hope to be there for all of you the same way you have been for me.

Thanks to Zein, Zayat, and Eva, for being my dearest friends, second family, and comfort zone that I enter filled with depression and leave it with nothing but happiness and love.

I would also like to thank Hassan, Abbas, and Giath, my friends, for not letting me down and always stick by my side whenever needed. You made the journey much easier.

Thanks to my uncles, cousins, friends, roommates, and university colleagues, for all the help and support.

Summary

As a general overview of the project and with the help of definitions presented in previous thesis work [1] " CardioFilo project aims to create an healthcare model for patients affected by atrial fibrillation, myocardial infarction and/or undergoing coronary angioplasty or other revascularisation procedures.". Its main purpose is to create a smartphone application to keep track of the patient behavior and a web application for the doctor.

After discharging the patient from the hospital, cardiological visits are limited to 3-6 month check-ups which may effect controlling habits like blood pressure, weight, or smoking cigarettes, and monitoring exam results, bleeding without help, and/or sleep monitoring. Lack of supervision may lead to some difficulties in managing radical changes of habits and following complex therapies, in addition to the possibility of experiencing a new cardiovascular adverse event.

In this master thesis work we will focus on the mobile application part of CardioFilo's project which aims to provide an additional tool for the secondary prevention of cardiovascular diseases through the smartphone. We will discus who will use it, how to develop it, tools used, future improvements, and of course security measures and precautions applied.

In CardioFilo's mobile application different tools and libraries were used. Android studio as a development tool, java as a programming language, Auth0 for authentication, and Okhttp for server connection. It was implemented jointly with Abinsula's mobile development team, and all the graphics were developed by Abinsula's graphic team. Both teams are located in Sassari and the collaboration with them took place remotely with the help of Gitlab, a web-based DevOps platform that provides Git-repository, and hangouts meet for scheduled meetings.

The mobile application is divided into multiple activities, some of them are read-only, uploaded by the doctor (i.e. personal data, medical reports, overview) which gives the patient the possibility to check his own situation. As for the other activities, they are for the patient to state his records to keep the cardiologist updated (i.e. record data, therapy).

Table of Contents

List of Figures

Acronyms

AI
Artificial Intelligence

IDE
Integrated Development Environment

AS
Android Studio

SSO
Single Sign-on

HTTP
HyperText Transfer Protocol

FTP
File Transfer Protocol

OAT
Opaque Access Tokens

HIV
Human Immunodeficiency Virus

BMI
Body Mass Index

DES
Data Encryption Standard

RC

Ron's Code

AES

Advanced Encryption Standard

DSA

Digital Signature Algorithm

RSA

Rivest Shamir Adleman

DH

Diffie-Hellman

Chapter 1

CardioFilo Mobile Project

1.1 Project Overview

CardioFilo's Mobile application is a part of the main CardioFilo's project, a service that aims to generate a technological service between the cardiologist and the patient to optimize cardiac patient home follow-up. It works along side with a Web application that is already implemented by Abinsula Web development team. Both platforms communicate with the server via Internet connecting by Wi-Fi/ADSL, Ethernet or 3G/4G [1] as shown in figure 1.1.

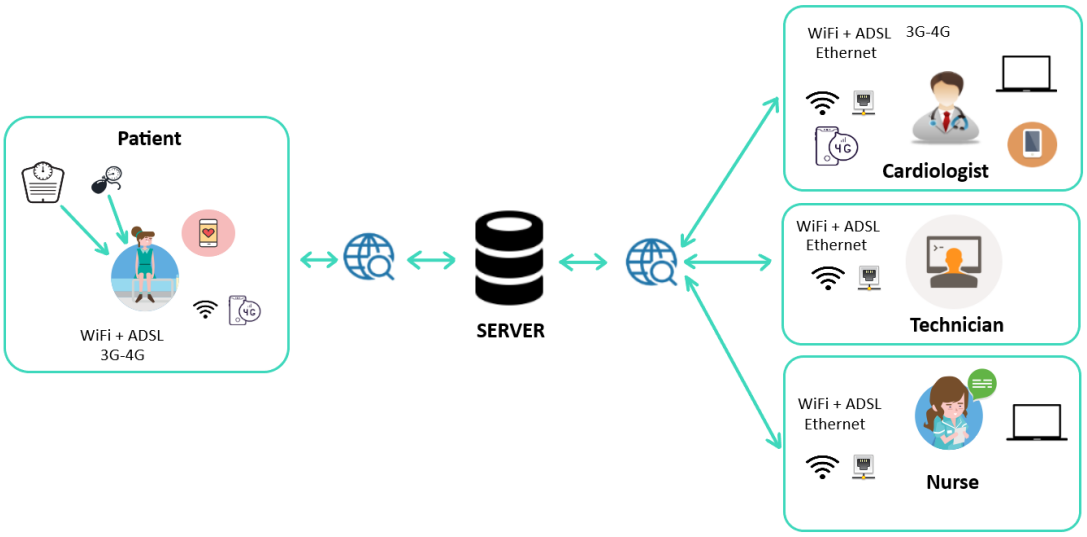

Figure 1.1: Schematic representation of the CardioFilo hardware architecture [1].

The Mobile application is mainly used by cardiac patients after their first

examination by the Cardiologist who will register them, using the Web platform, in the patient's list and create a unique username and password in order to log-in to the app.

1.2 Actors

CardioFilo's mobile application actors are not only those who have direct usage, but also indirect part in the application's workflow as shown in figure 1.2.

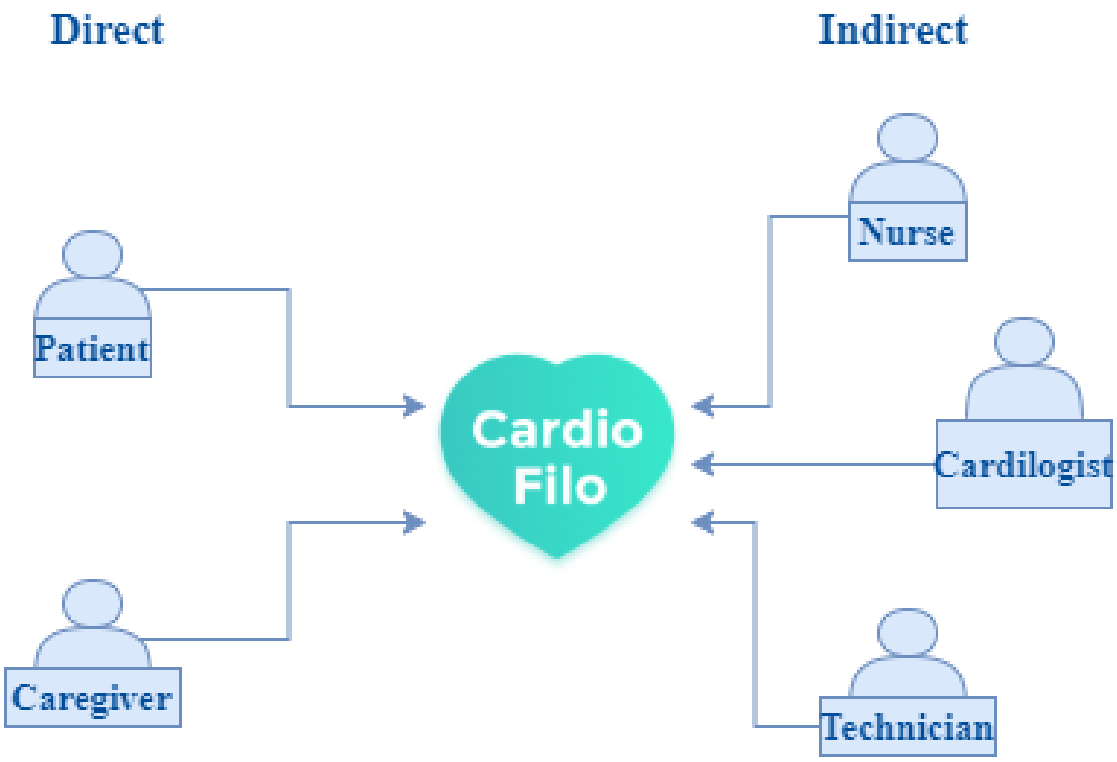

Figure 1.2: CardiodFilo's mobile application actors.

Direct actors:

- Patients: main users of the application.
- Caregiver : substitute of the patients in-case of any difficulties.

Indirect actors: Tasks

- Cardiologist: upload medical records, using the web application, which will be viewed by the patient using the mobile application. In addition to that, monitoring all inserted patient's record data.
- Nurse: Insert patient's personal data.
- Technician: Develop and update the application.

1.2.1 Cardiologist and Nurses

Even-though both the cardiologist and the nurses do not interact directly with the mobile application, they are important actors in the whole system due to the fact that they are responsible for entering all the data that represent the patient.

For each patient, the mobile application shows the personal data which was already inserted by the nurse, using the web platform, before the first visit. Nurses should ask the patient for the required personal data :

- Name and surname;
- Email;
- Data of birth;
- Place of birth;
- Fiscal Code;
- Address;
- Phone Number;

After that, the cardiologist is responsible to enter the results of the examination, using the web platform, in the patient's medical reports and present a suitable therapy. The patient can check them using the mobile application which will show:

- Risk factors, i.e. hypertension, diabetes, previous chemotherapy, previous radiotherapy, HIV in treatment, smoke, severe renal treatment.
- Body mass index, i.e. height, weight, BMI.
- Allergies and active implantable devices.
- Previous acute coronary events.
- Ejection fraction.

- Surgery dates if needed.
- General, cardiovascular, and recent history.
- Suitable therapy plan

Cardiologists can also monitor the situation of the patients on his web platform by viewing the recorded data inserted by the patients.

Advantages of using CardioFilo by the cardiologist:

- Always have updated information on the patient.
- Simple and effective way for collecting data (e.g. : blood pressure, heart rate, weight, adverse events).
- Optimized therapeutic measures.
- Direct communication with the patient in case of changes in the health status.
- Timely identification of the most critical patients.

1.2.2 Patient

Cardiac patients are the main users of the mobile application. Usually 75 years old women and 65 years old men are the most suspected to have this kind of diseases. After the first examination, through username and password authentication, patients can start using the app that already contains all their personal data and medical reports uploaded earlier by the cardiologist. Patients, using the application, can enter daily recorded data and therapy changes (figure 1.3).

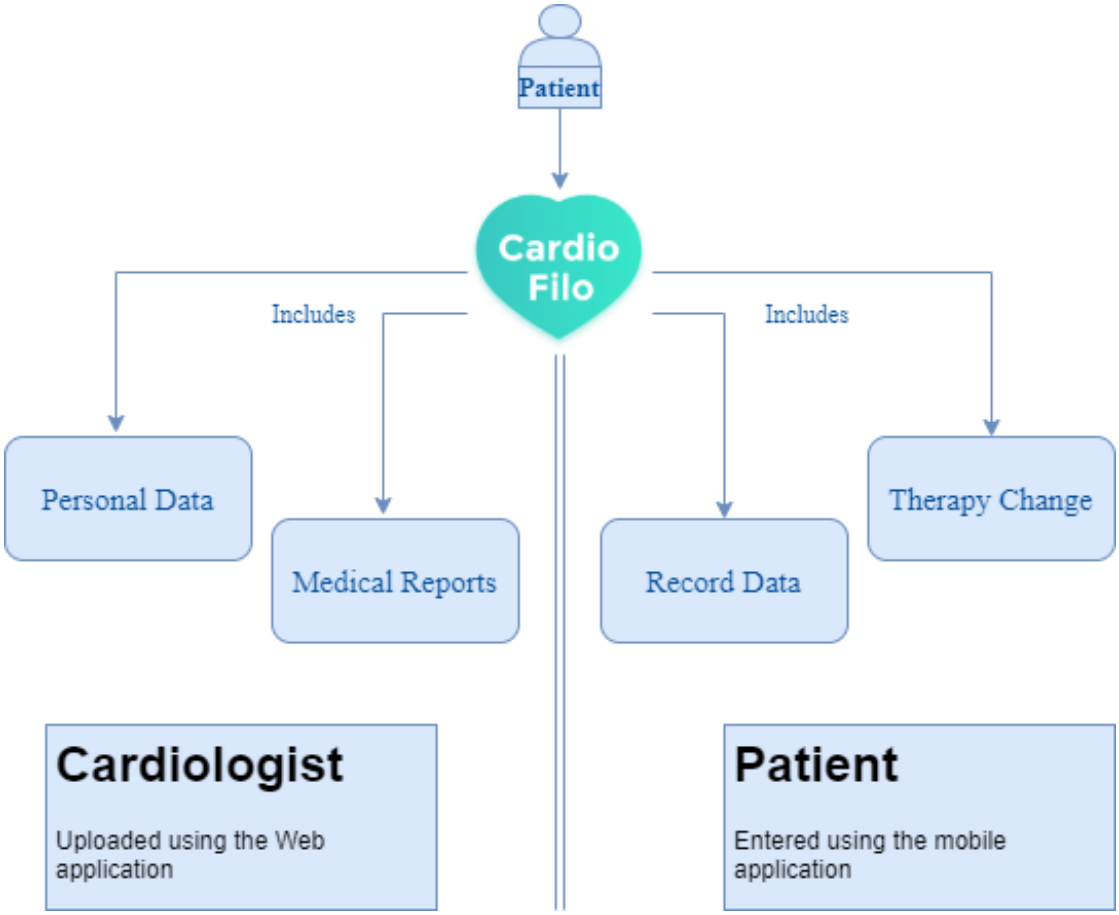

Figure 1.3: Patient's read and write activities in CardioFilo application

Record data is divided into 2 parts, daily and events. In the daily part patients can enter:

- Weight;
- Blood pressure;
- Heart Rate;
- Glycemia;
- Smoked cigarettes;
- Sleep monitoring;

The event part is more focused on the medical situation, visits, and therapy plans. Patients can :

- Change the therapy.
- Ask for a visit by stating the reason.
- Inform about bleeding without help cases.
- Ask for hospitalization.
- Add exam results.

Patients can also check their therapy suggested by the doctor using the application. It will include the drugs needed for each day and time. In addition to that, a calendar is also implemented to provide the ability to check previous and future therapy plans.

Advantages of using Cardiofilo by the patient:

- Continuous monitoring of his health conditions by the reference cardiologist.
- Feeling of comfort and security arising from knowing that there is a doctor who monitors his/her situation.
- Optimized therapeutic measures.
- Simple and effective way for collecting clinical data.
- Increased awareness of his health status.

1.2.3 Caregiver

Caregiver is a [2] "Person who takes primary responsibility for someone who cannot care fully for himself or herself. The primary caregiver may be a family member, a trained professional or another individual. A person may need care due to loss of health, loss of memory, the onset of illness, an incident (or risk) of falling, anxiety or depression, grief, or a disabling condition.".

Since the patients are elderly, we should take into consideration that they may have cognitive or visual impairments which would make it difficult for them to use smartphones or tablets. In this situation, the caregiver can use the mobile application instead of the patient. He/She can login using the patient's credentials and substitute them.

1.2.4 Technician

The technician does not have a direct interaction with the mobile application, but still considered as a main actor in the project. He is responsible of developing the mobile application and fixing all discovered bugs. Even after the application is published his part is not finished, he should always be available for adding new features and updating available ones.

1.3 CardioFilo's Activities

To have the best representation of the mobile application, UML diargrams are presented according to the requirements discussed in previous thesis work [1] about CardioFilo's project (see figures 1.4, 1.5, 1.6).

1.3.1 UML: Use Case Diagrams

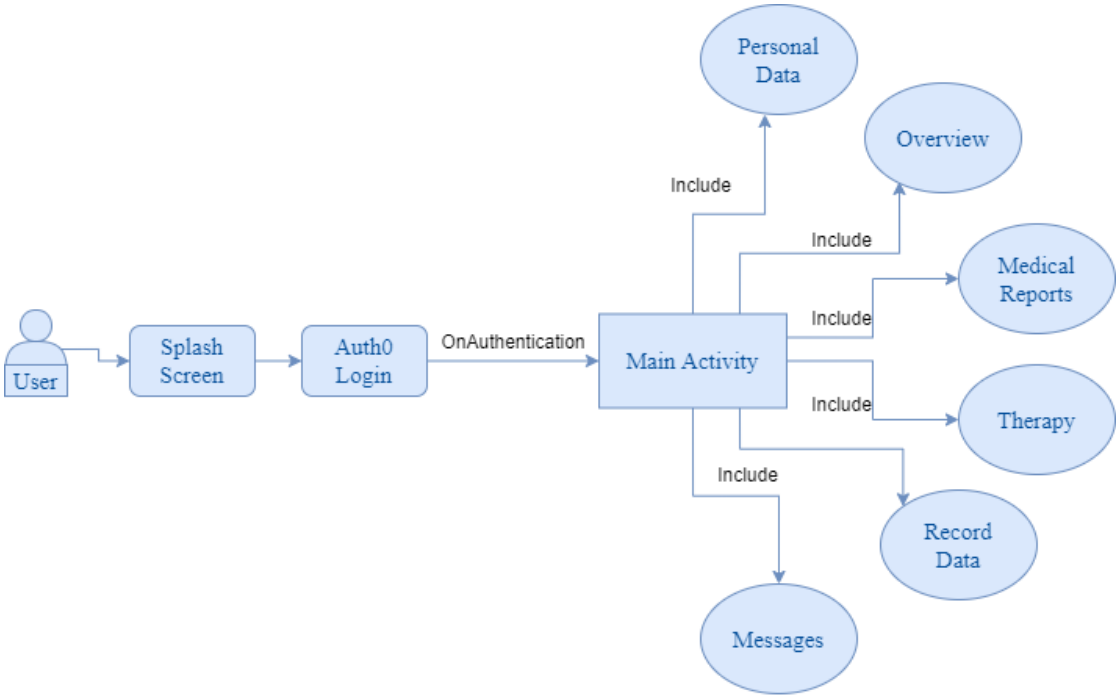

Figure 1.4: Schematic representation of CardioFilo's mobile application use case by the patient

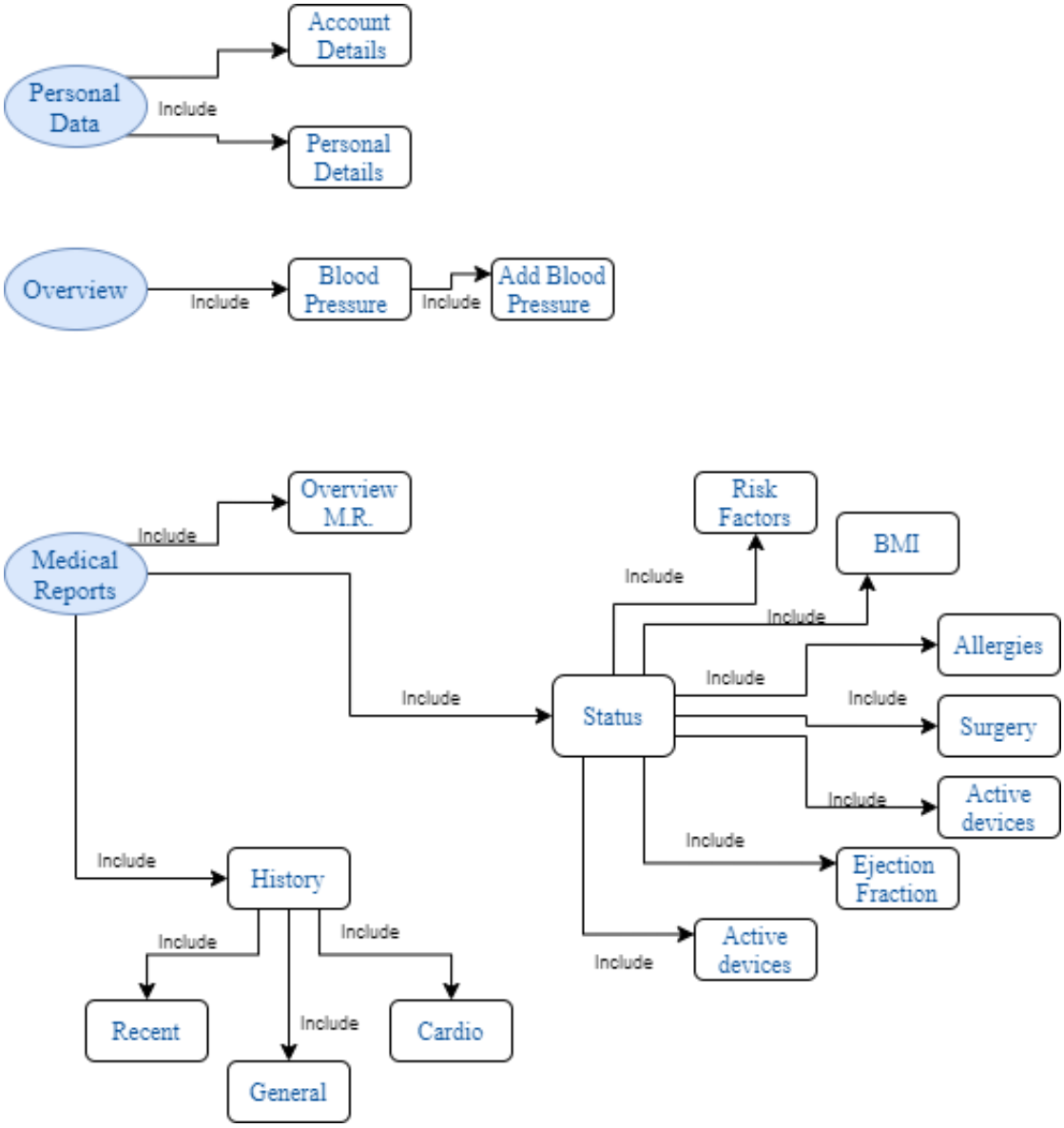

Figure 1.5: Schematic representation of CardioFilo's mobile application use case by the patient

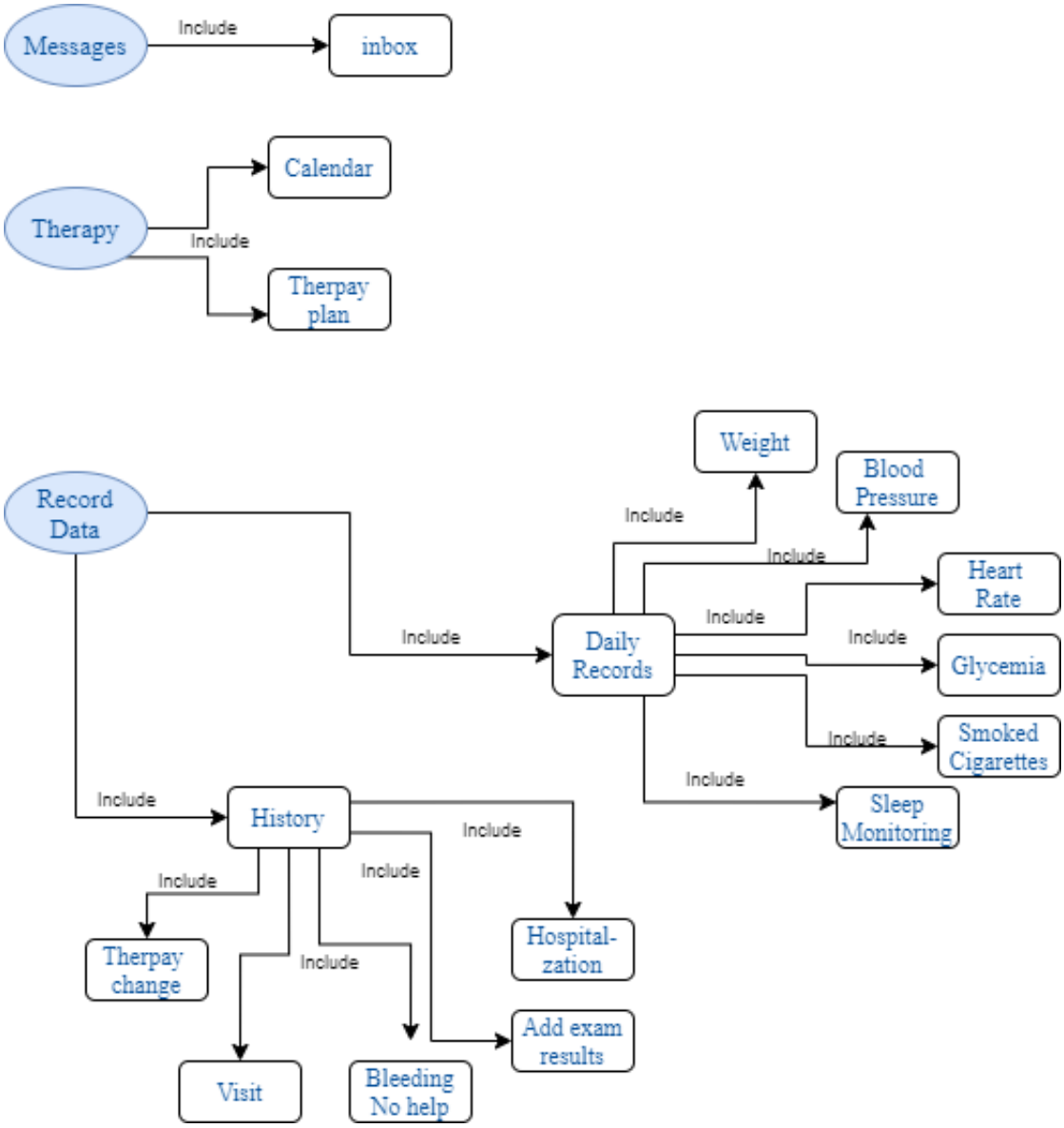

Figure 1.6: Schematic representation of CardioFilo's mobile application use case by the patient

1.3.2 UML: Activity Diagrams

For a more precise design of the mobile application architecture, activity diagrams were made. Each diagram represents a single screen of the CardioFilo's mobile application and contains detailed information about the logical flow of data (figures 1.7, 1.8, and 1.9).

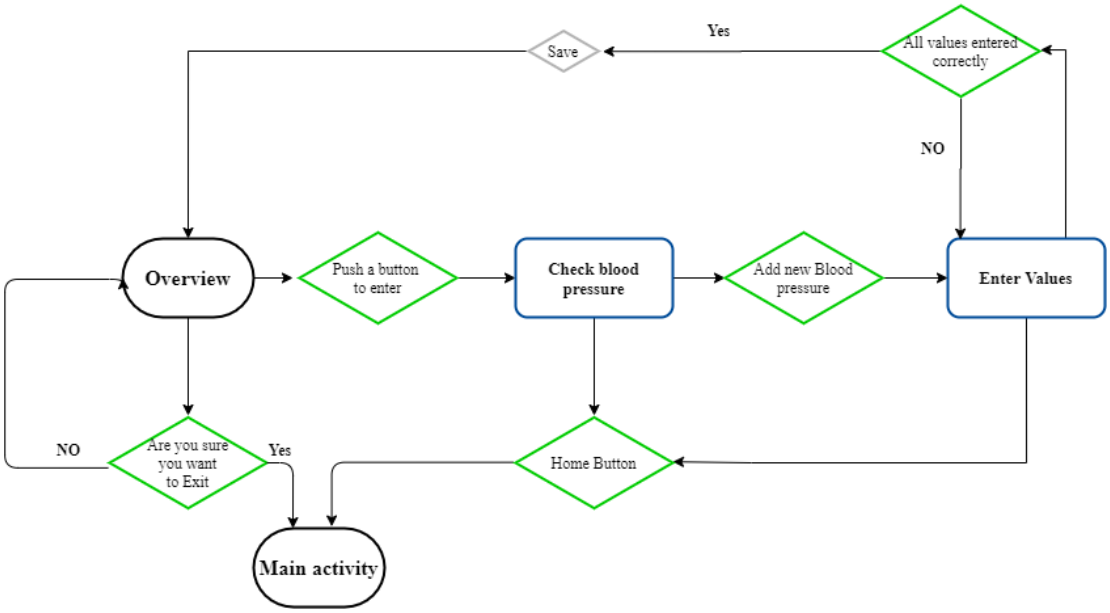

Figure 1.7: Activity Diagram relative to the patient overview

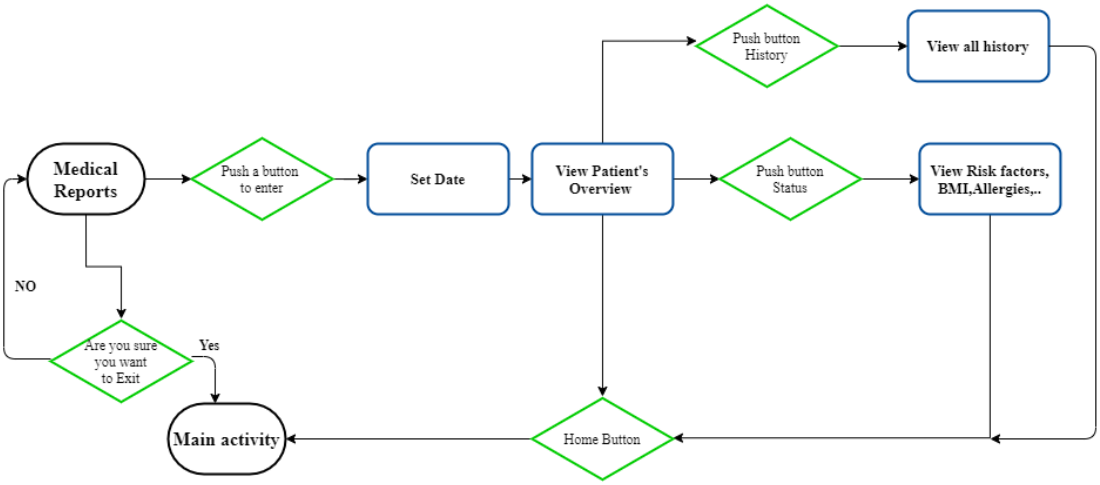

Figure 1.8: Activity Diagram relative to medical reports

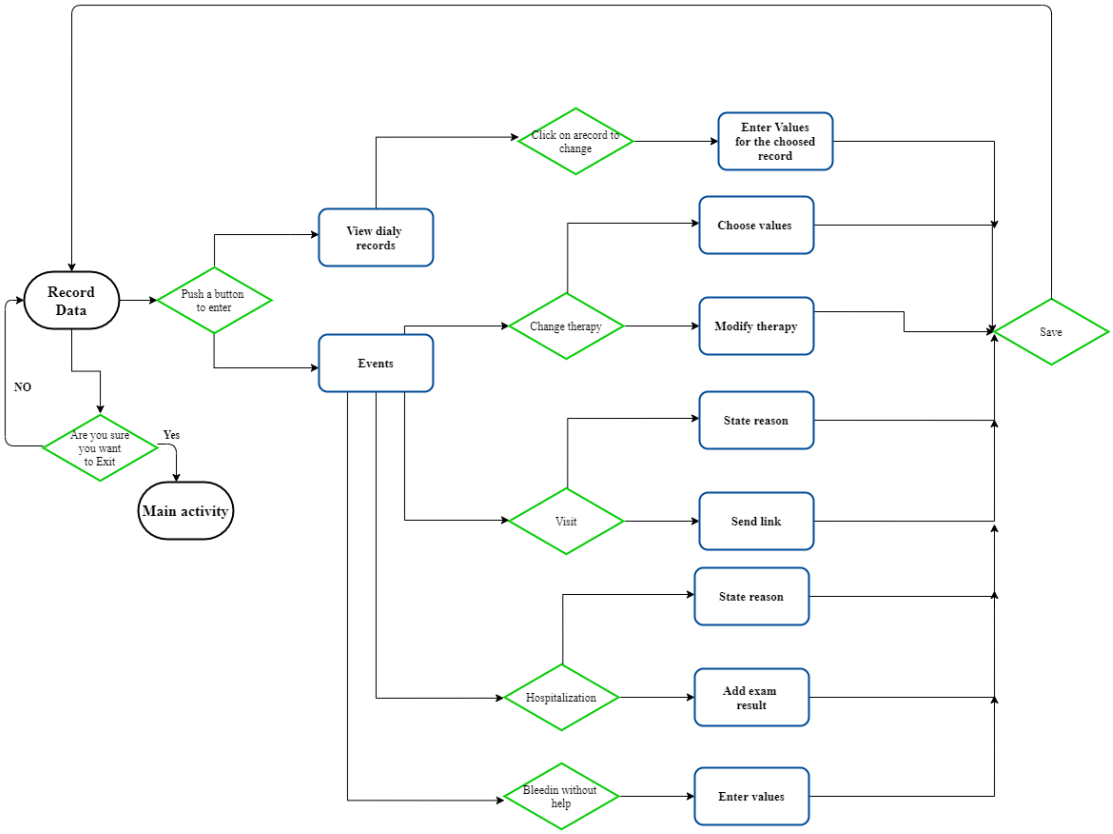

Figure 1.9: Activity Diagram relative to record data

1.4 CardioFilo Graphics

CardioFilo's mobile application graphics where created by Abinsula's graphic team based on several meetings and mock-ups provided in previous thesis work [1]. All the graphics were uploaded on Zeplin, a connected space for product teams to share designs and style guides with accurate specs, assets, and code snippets [3].

These graphic were shared among developers whose email is certified to enter the project. Figures 1.10, 1.11, 1.12, and 1.13 are some of the screens uploaded on Zeplin. Each one of them contain all the styles, buttons, images, dimensions, and colors needed by the developer in order to create the exact view.

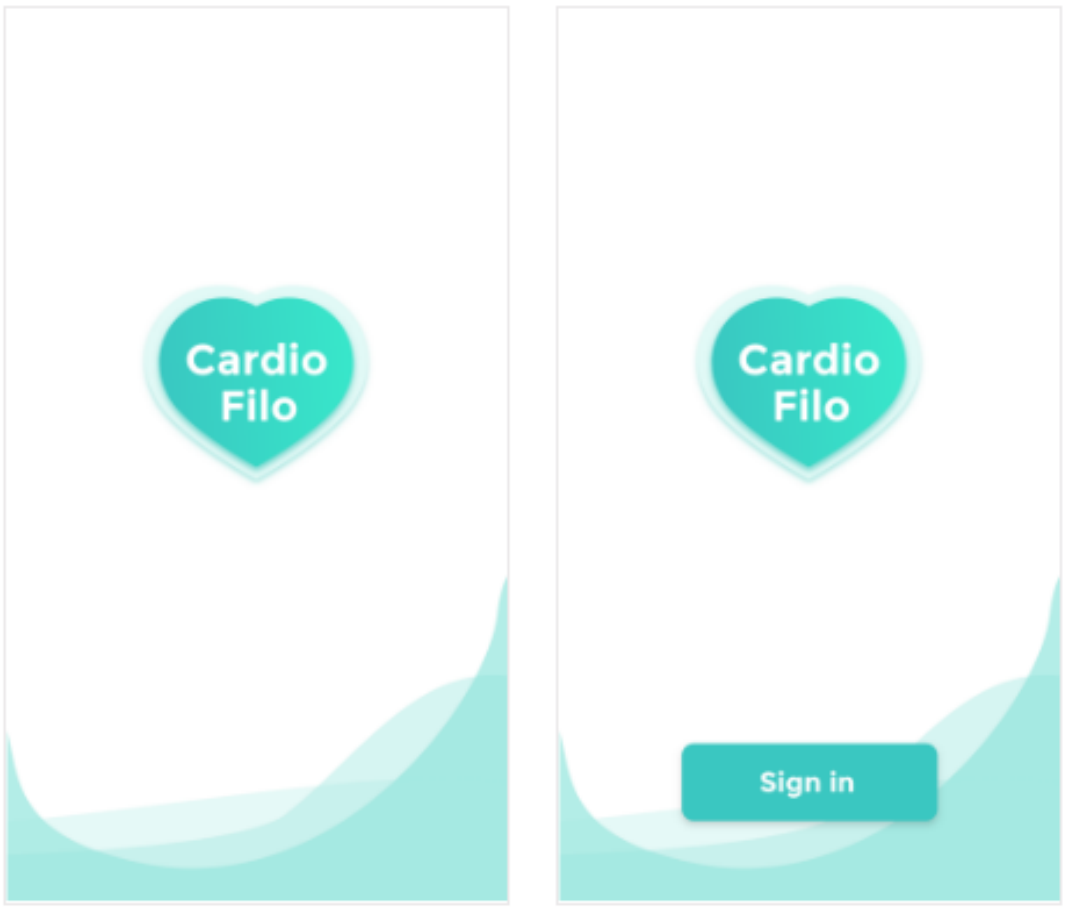

Figure 1.10: Graphics of the mobile application on Zeplin

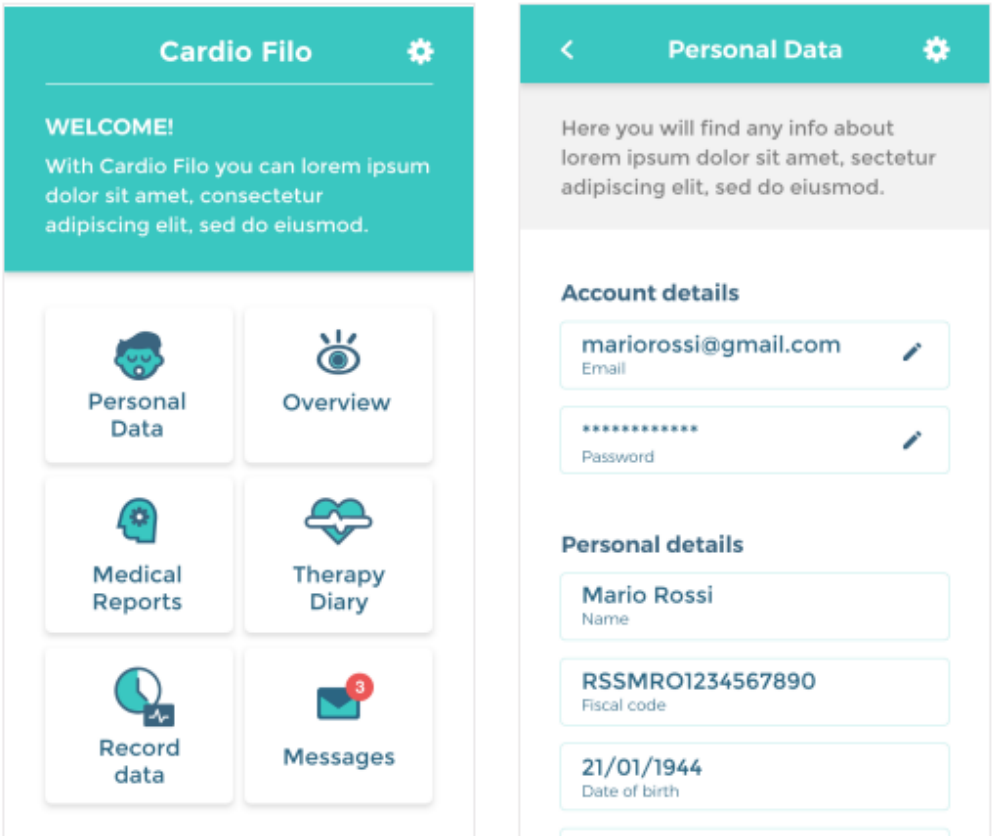

Figure 1.11: Graphics of the mobile application on Zeplin

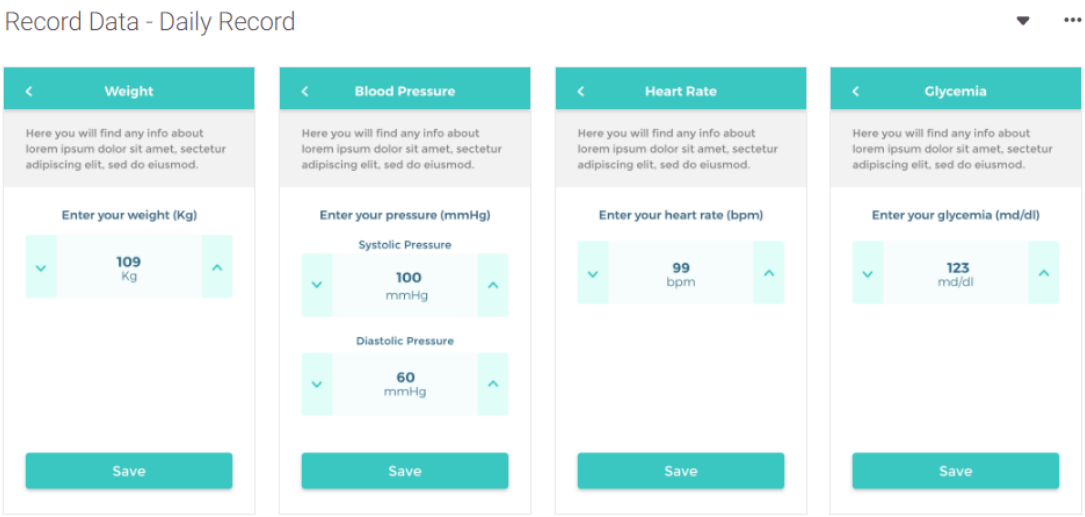

Figure 1.12: Graphics of the mobile application on Zeplin

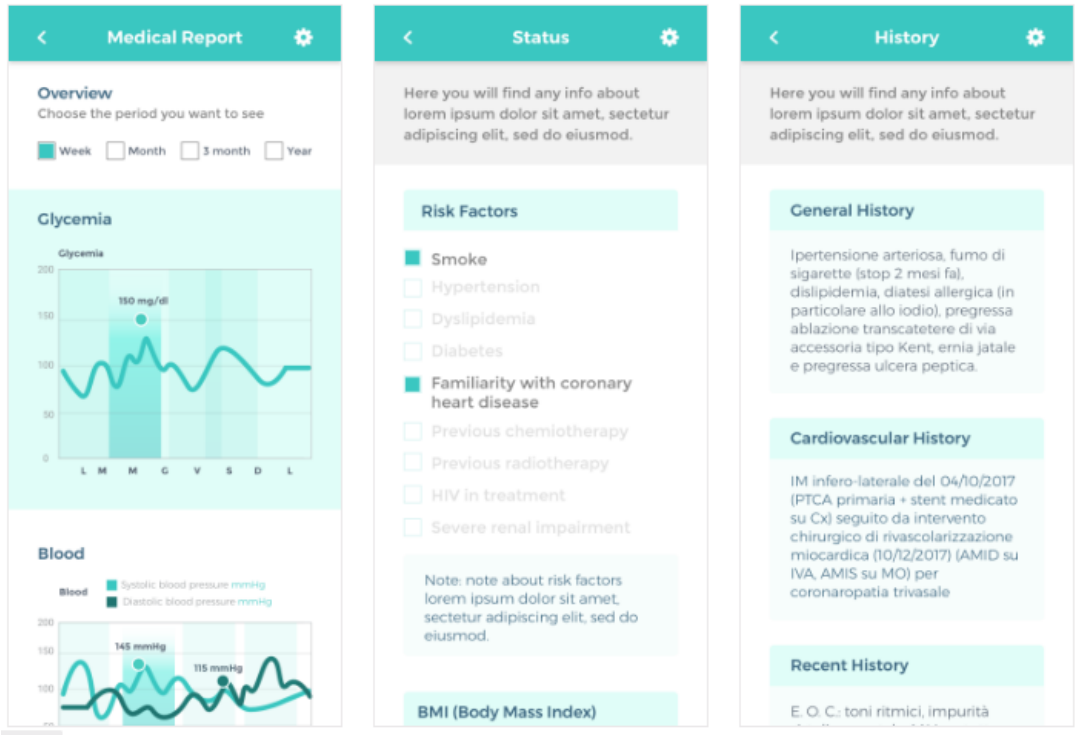

Figure 1.13: Graphics of the mobile application on Zeplin

Chapter 2

Tools used

Although the final application is always the programmer's concern and that is what is finally shown, the base of every project is choosing which tools are used and finding out which is better, in a way that makes the project more efficient and in the same time simpler.

The first step of choosing tools and properties is a deep research and comparisons based on the previous knowledge from the programmer on what is needed. At the beginning of every mobile application development, decisions should be made. Which programming tool should be used? which programming language? and after that comes Authentication strategy, security, database and connection management, and above all project management.

In CardioFilo decisions were made based on researches and the experience of Abinsula Team.

2.1 Android Studio

Android Studio is the official IDE "Integrated Development Environment" for Google's Android operating system, built by JetBrains' IntelliJ IDEA software and designed specifically for Andriod development. Besides JAVA, andriod studio supports Kotlin and C++.

Figure 2.1 shows the main functionality of android studio in which different types of codes are compiled in order to create an APK which will be installed on the mobile phone in order to use the developed application.

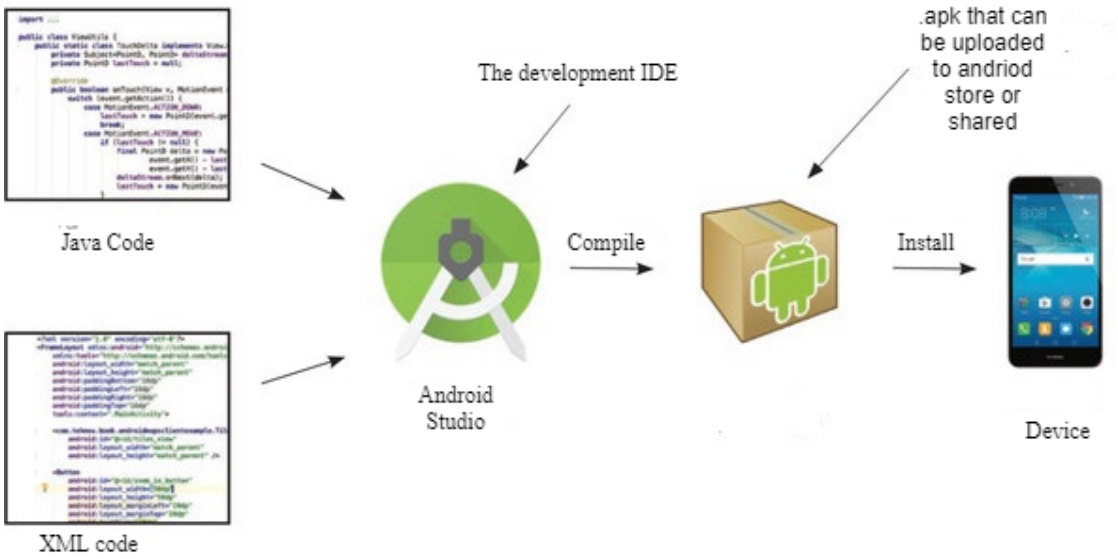

Figure 2.1: Overview of how android studio works

2.1.1 Why android Studio?

Looking for a stable IDE, always choose Android Studio(AS), which is specifically designed to accelerate the process of Android mobile App development. Ever since its announcement in 2013, AS has been in the buzz because it nearly meets all the expectations of developers.

Over 76.6 percent of smartphone giants like Samsung, LG, and HTC are using Android OS, at present. Android is soon expected to take a leap to laptops, smart watches, and cars. Android holds estimated 2 US billion dollars net worth as it's used by a majority of people across the world.

AS have lots of advantages that makes it better than the others, and others mainly means Android-Eclipse[4]:

- **Gradle Build System**

  Android Studio uses highly integrated Gradle build system which offers dependency management and enhances developer experience because it is more extensible.

- **User Interface**

  Eclipse is big, so as a first time use for developers is a sort of a problem which it is not in AS. The menu and tools used in it are prompt and can be easily

used.
Also to mention that AS is specifically built for Android, the user interface is smooth as compared to eclipse which was built for all-purpose IDE.

- **Availability of Drag-and-Drop**

  Android Studio offers Graphical User Interface, so the knowledge of Visual Basic helps developers to use drag-and-drop appropriately.

  Mainly, drag-and-drop is used by coders who are concerned to have an in-depth knowledge of Visual basic, not by those who are not concerned about their applications' visual elements.

- **Java Code Auto Completion**

  Although Java code auto-completion is supported by both Eclipse and AS, AS is better than Eclipse due to the fails to provide the precise solutions most of the time in Eclipse. Considering the importance of code completion for a developer, AS has been designed to offer precise solutions.

- **Stability of System**

  AS offers more stable performance than Eclipse, it arrives with lesser bugs, lower system requirements, and easy approach.

  Eclipse requires a higher Ram and CPU speed because of the larger IDE and the basic Java-based software, so the failure to get the desirable criteria may result a crash and becoming unresponsive. In addition, AS requires 30 seconds to compile a release version which may take up to 2 minutes in Eclipse.

  In addition to all the points mentioned above, Android Studio is absolutely one step ahead for the ability to run and debug apps using the Android Virtual Device (Emulator), support for building Android Wear apps, built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging and Google App Engine, and Template-based wizards to create common Android designs and components.

2.1.2 Java Or Kotlin?

This title comes with lots of tags and questions: what is the difference? which is better? which is more used? lots of facts and different opinions.

The best way to illustrate the difference is by listing the pros and cons of both of them [5].

Pros of Java :

- Easy to learn and understand in addition to flexibility.
- Large open-source ecosystem and libraries.
- Good cross-platform apps.
- Java apps are more compact and lighter as compared to Kotlin.
- Java possesses faster build process than Kotlin.
- Thanks to proceeding assembly with Gradle, collecting large projects becomes easier in Java.

Cons of Java

- Comes with limitations such as it causes problems with Android API design.
- Higher risk of bugs since it requires writing more code.
- Slower in comparison to many other languages since it requires a lot of memory.

Pros of Kotlin

- Fewer errors and bugs due to the less amount of code.
- Kotlin aids in building a clean API.
- Usage of Java libraries and frameworks in Kotlin.
- Solves the nullability problems faced in Java by placing null directly in its type system.
- Reduces unit test case because of static typing.

Cons of Kotlin

- Slower compilation speed than Java
- There is definitely a learning curve with Kotlin: the very short syntax, while a great benefit, requires some learning.

- Some features of Android Studio such as auto-completion and compliance run slower in Kotlin than Java.
- Experienced Kotlin developers are still a rarity.

Despite all the benefits of Kotlin, Java was the programming language used in Cardiofilo, considering the fact that Java is one of the most powerful programming languages and more flexible.

2.2 Auth0

Auth0 is a flexible, drop-in solution to add authentication, authorization services, and Token-based Single Sign On for your Apps and APIs with social, databases, and enterprise identities. It is a set of unified APIs and tools that instantly enables Single Sign On and user management to all your applications[6].

Teams and organizations have to pay to use Auth0 but save time and reduce the risk that comes with building their own solution to authenticate and authorize users. In addition to connecting any application written in any language to Auth0, developers can define the identity providers they want to be used in order to log in. Authentication can be provided in different forms in Auth0, either by providing username and password or by using social accounts already certified by Auth0.

2.2.1 Why use Auth0?

Auth0 is trusted by Abinsula as the Authentication provider, and that trust was earned based on several things that Auth0 provides[7]:

- Ability to login using social accounts.
- Implementation of Single Sign ON (SSO) for more than one app.
- Secure access API's.
- Authentication of users using Security Assertion Markup Language (SAML) in web applications.
- Generation of one-time codes delivered by email or SMS.
- Proactive block of suspicious IP addresses if they make consecutive failed login attempts, in order to avoid Distributed Denial of Service (DDoS) attacks.

Why setting effort and time for building your own solution to provide Authentication when there is a fully functional and secured library that can be easily used?

2.2.2 CardioFilo Authentication using Auth0

When designing the authentication experience for an application using Auth0, two choices can be considered concerning the login flow, universal or embedded login[8]. Both have one main purpose which is authenticating the user in order to login but they are different in the way they achieve it.

In Cardiofilo both of them were used, separated in 2 different buildTypes, Release and Debug. 2 different flavors were developed in each buildType, both with the same functionality but with different types of login flows.

Figure 2.2 shows the 4 apks created for both Universal and Embedded login in 2 separated buildTypes which the company have the right to choose which one the user can use.

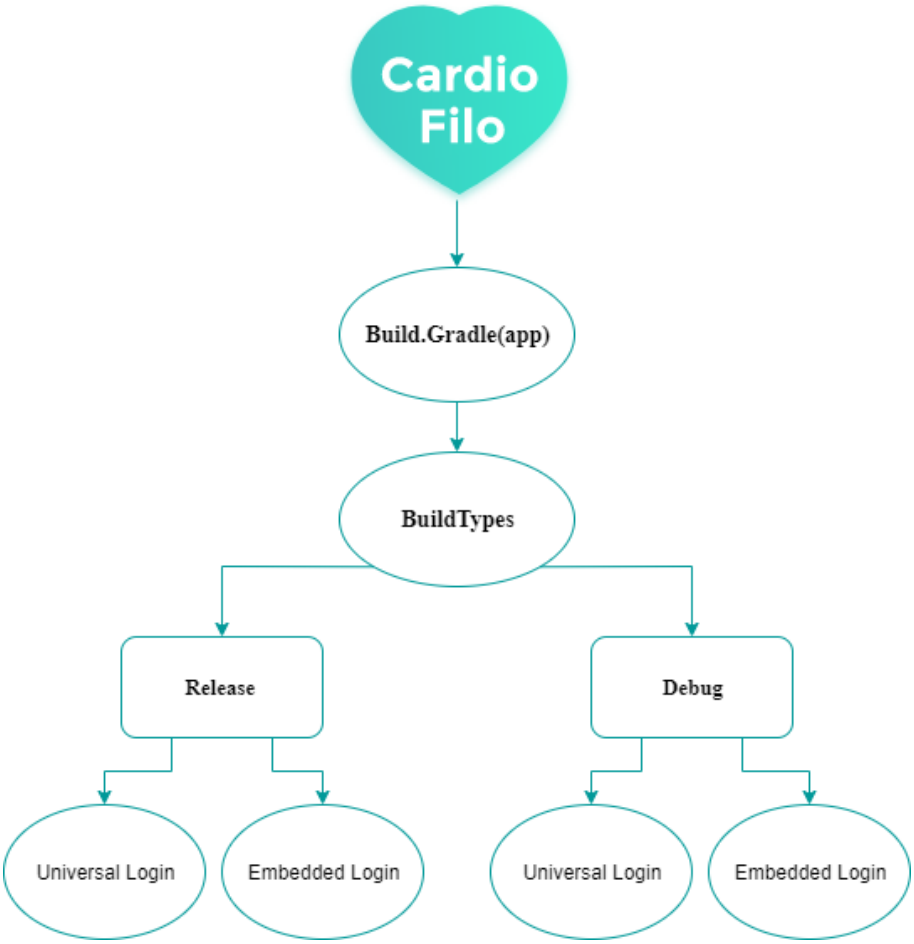

Figure 2.2: CardioFilo Build Variants division to support 2 different types of Login

Universal Login

Universal Login is Auth0's implementation of the login flow, which is the key feature of an Authorization Server. Each time a user needs to prove their identity, the application redirects to Universal Login and Auth0 will do what is needed to guarantee the user's identity. In other words as shown in figure 2.3, with Universal Login, when the users try to log in they are redirected to a central domain, through which authentication is performed, and then they are redirected back to the app[9].

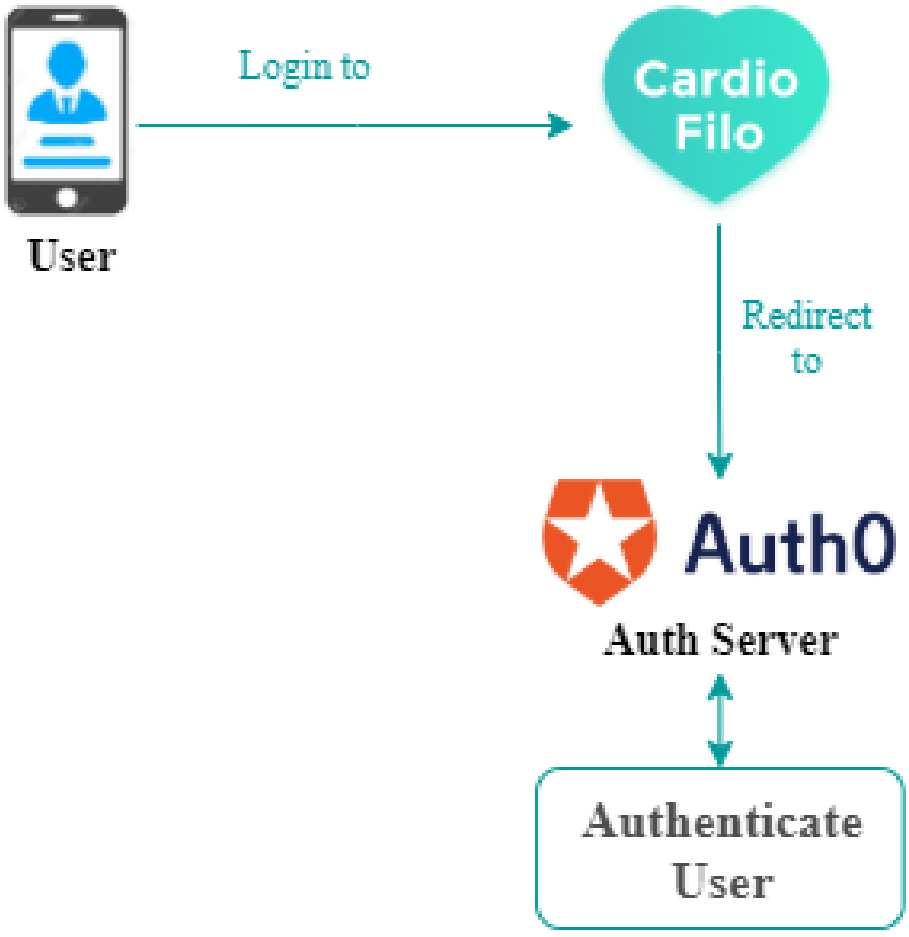

Figure 2.3: Universal Login Flow in CardioFilo

By choosing Universal Login, it is not necessary to do any integration work to handle the various flavors of authentication, simple username and password login can be implemented. In addition there is a toggle switch that can include features such as social login (Facebook, twitter,..)[8].
SSO is better supported in Universal Login, due to the fact that with embedded

login, it is necessary to collect the user credentials in an application served from one origin and then send them to another one, which can present certain security vulnerabilities.
Whenever the app triggers an authentication request, the user will be redirected to the login page in order to authenticate. This will create a cookie. In subsequent authentication requests, Auth0 will check for this cookie, and if it is present the user will not be redirected to the login page. They will see the page only when they need to actually log in. This is the easiest way to implement SSO.
Consistency and Maintenance in Universal Login is mush easier if you have multiple apps because you do not have to implement different login pages. The Authorization Server owns all the login pages which makes the management easier and the pages more consistent and secure.

Embedded Login (Lock)

In Embedded login flow, Lock for mobile was used. Lock does not redirect the user somewhere central. The login widget is served from the same page without redirecting the user to another domain. The credentials are then sent to the authentication provider for authentication all inside the app as shown in figure 2.4.

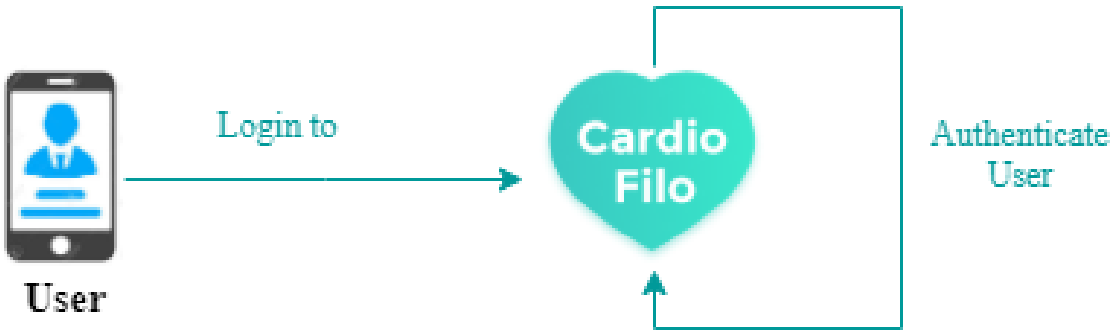

Figure 2.4: Embedded Login Lock Flow in CardioFilo

In Embedded Login, User experience is better because it does not require redirecting to another sub domain, users will stay in the same application to perform authentication. In addition, Embedded Login contains a library which the programmer can manage in order to change the UI.

2.3 OkHttp for server connection

In general, HTTP is the way modern applications network exchange data over the internet. HTTP efficiently loads faster and saves bandwidth.

OkHTTP is an open source project designed to be an efficient HTTP client, which by default[10]:

- Connection pooling reduces request latency in case HTTP/2 is not available.
- Transparent GZIP shrinks download sizes.
- Response caching avoids the network completely for repeat requests.

Initially Android had only two HTTP clients: HTTPURLConnection and Apache HTTP Client, for sending and receiving data from the web. OkHttp android provides an implementation for both client interfaces by working directly on top of a java Socket without using any extra dependencies.
Using OkHttp is easy. Its request/response API is designed with fluent builders and immutability. It supports both synchronous blocking calls and asynchronous calls with callbacks.
In CardioFilo, Okhttp was used in order to request and post data from the web server.

2.3.1 Request Data

In order to Request Data in Android using Okhttp, 2 main things are needed, Base URL from which the data is stored and the access token that grants authorization.

```
final Request.Builder reqBuilder = new Request.Builder()
        .get()
        .url(url)
        .addHeader( name: "Authorization",  value: "Bearer " + accessToken);

    OkHttpClient.Builder builder = new OkHttpClient.Builder();
    builder = IClient.configureToIgnoreCertificate(builder);
    OkHttpClient client = builder.build();

Request request = reqBuilder.build();
client.newCall(request).enqueue(new Callback() {

    @Override
    public void onFailure(Call call, IOException e) { e.printStackTrace(); }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
```

Figure 2.5: Okhttp Request code used in CardioFilo

Figure 2.5 shows the code of how to request data using okHttp. Based on the URL and Access token, a request.builder is generated which will be called using a new generated OKHttpCLient builder as shown in figure 2.6. In case of success, data will be retrieved in the onResponce callback. Otherwise, onFailure will return the reason of failure.

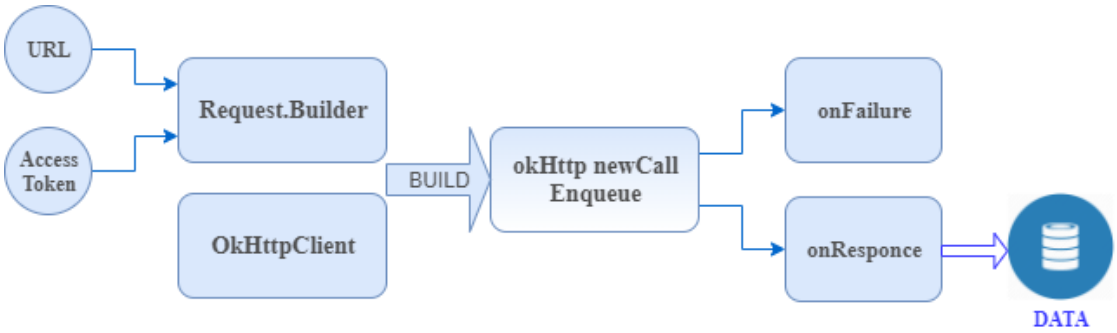

Figure 2.6: Okhttp Request Diagram

2.3.2 Post Data

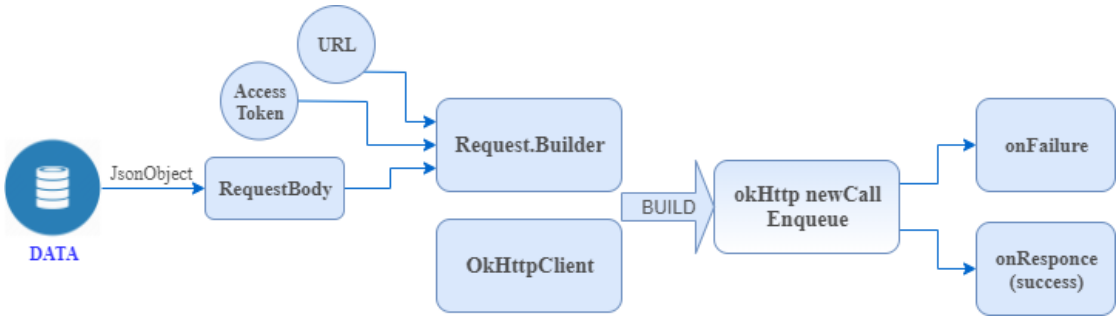

Figure 2.7: Okhttp Post schema

Posting Data to a web server using okHttp have the same procedure as when requesting data. The only difference is that here, data should be included (see figures 2.7 and 2.8).

```
RequestBody PostD = RequestBody.create(MEDIA_TYPE, PostData.toString());

String url = BuildConfig.BASE_URL + "documents/bloodPressure/";

final Request request = new Request.Builder()
        .url(url)
        .post(PostD)
        .addHeader( name: "Authorization",  value: "Bearer " + accessToken)
        .header( name: "Accept",  value: "application/json")
        .header( name: "Content-Type",  value: "application/json")
        .build();


OkHttpClient.Builder builder = new OkHttpClient.Builder();
builder = IClient.configureToIgnoreCertificate(builder);
OkHttpClient client = builder.build();


client.newCall(request).enqueue(new Callback() {

    @Override
    public void onFailure(Call call, IOException e) {
        e.printStackTrace();
        showBaderror();
    }

    @Override
    public void onResponse(Call call, Response response) throws IOException {
```

Figure 2.8: Okhttp Post Code used in CardioFilo

2.4 Git

Git is generally known as a distributed version-control system for tracking source code changes during any software development. The main purpose of Git is coordinating work among programmers, manage a project, or a set of files, as they change over time. Its goal is data integrity, speed, and support for distributed, non-linear workflows in additon to many other Charactristics[11]:

- Strong support for non-linear development in which multiple merges and branches could be commited and created in the same porject.
- Compatibility with existent systems and protocols as HTTP, FTP, CVS, and IDE plugins.

- Efficient handling of large projects
- Cryptographic authentication of history
- Distributed development which also includes local branches
- Pluggable merge strategies

2.4.1 GitLab

GitLab is a web-based DevOps platform, that provides Git-repository. It is known with its unmatched visibility and the high level of efficiency in a single application across the DevOps lifecycle. This makes GitLab somehow unique and makes concurrent DevOps possible[12].

CardioFilo with GitLab

Gitlab was used in CardioFilo in order to manage the work on the project between multiple parties. Different partitions were created, Cardiofilo Web project, front-end, back-end, and of course mobile project.
Beside using gitlab as Git to add, fetch, and commit codes, it was also used to organize tasks. Different tasks can be created and assigned to developers along side with a branch in which the task is done. Figure 2.9 shows some of the branches created while developing Cardiofilo.

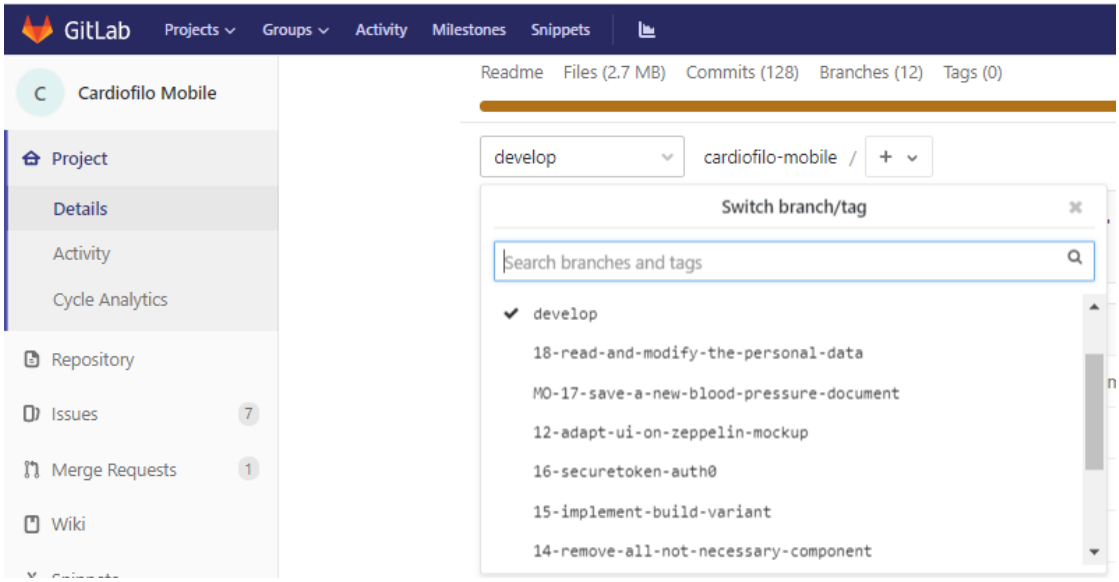

Figure 2.9: Overview of CardioFilo branches in Gitlab

Branches can be created for each set of related changes in which it keeps each one of them separated from the other, to allow changes be made in parallel without any collisions as shown in figure 2.10. After pushing all the changes a merge can be done to the original branch that contains the original finished code.

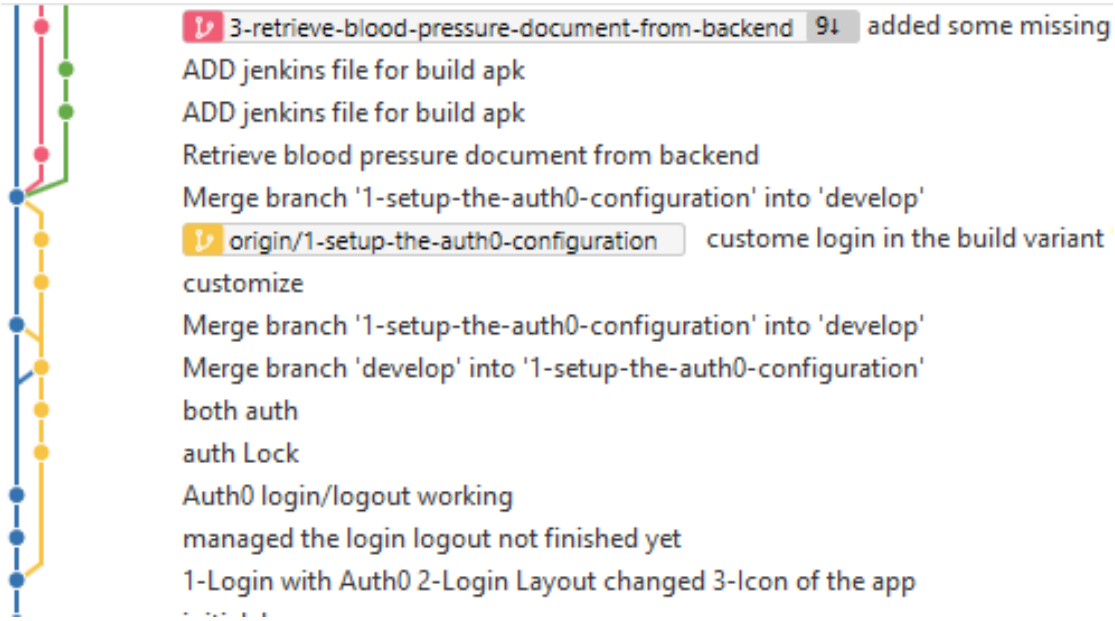

Figure 2.10: Screenshot of branches created and merged in Cardiofilo

GitLab is compatible with Android Studio. Branches can be fetched directly, same as adding, committing and pushing changes to it.

2.5 Database and API's

CardioFilo's mobile application contains clinical data for patients who suffer from different kinds of cardiac diseases, so a health information system with efficient representation is required. For this reason, openEHR was used as a database for storing the patient's clinical data.

OpenEHR is an open health standard that concerns electronic medical records and the standardization of the data contained in them. And as defined in [13]: "openEHR is feasible and easy-to-apply IT platform that possibly provide several benefits over single-level information systems in the care of elderly, comorbid population".

Sqlite3 and Postgres databases were also used for non-clinical data (i.e. personal data).

- SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine [14]. It was used for local development environment.
- Postgres is a free and open-source relational database management system emphasizing extensibility and technical standards compliance [15]. It was used for test and stage environment.

CardioFilo's databases were written in python using django, a high-level Python Web framework that encourages rapid development and clean, pragmatic design [16]. All of them including the APIs were developed by Abinsula's back-end development team.

The goal of this Back-End service is to provide a REST API to save and retrieve clinical documents, record data, and personal data of patients using CardioFilo's mobile application. All clinical documents were stored following the open EHR standard specification, but provided to doctors through human readable format. So that, this service will store each document reference into its database, which will be used to retrieve clinical data through the Clinical Gateway service downstream. To allow the application access these APIs to retrieve or post data, an Auth0 opaque token should be presented. The identity service is based on Auth0 identity platform and the authentication process uses the token granted by Auht0 after authenticated login to the app. This token should be included in the header of the request of these APIs (discussed briefly in section 3.1 of this thesis work).

Figure 2.11 shows the process of requesting data, used in CardioFilo, from the database using REST APIs and authorized by Auth0.

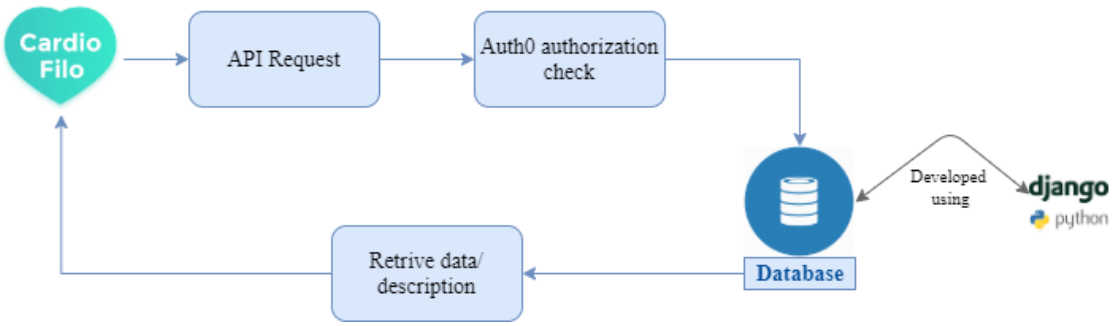

Figure 2.11: General overview of the API and database connection

Table 2.1 shows the GET and POST API requests, both of them include the URL of the API, but they differ in the parameters and response. Obviously, POST must include parameters which meant to be stored in the database using the API called, and as a response it will get a status code that represents the condition of the request. Each status code stands for a result already developed by the programmer

(i.e 200 success, 400 error, 500 connection error,...). The description if these codes are presented in the description, this helps the developer understand the result of the request especially if the developer of the APIs is different from the developer who uses them. GET requests include only the URL of the requested API and of course the access token. On success, the response retrieves the requested data.

	GET	POST
URL	URL of the document needed	URL to the document to be added or updated
Parameters	None	body request model to be saved
Response	body request model of the data requested	status code
Description	description of the document	description of the document and status code

Table 2.1: GET and POST requests for storing and retrieving data from APIs

Chapter 3

Security

Security is the most important aspect that must be taken into consideration while developing any kind of project. By making the application more secure, user trust and device integrity is granted[17]. Lack of security gives hackers direct control on the device which can result to:

- Access to personal information.
- Denial of service to a single user.
- Loss of service.
- Damage to the systems of thousands of users.

Cardiofilo is a medical application which contains patient's medical data. Security measures were applied on each phase of developing with the supervision of Abinsula team.

3.1 Cryptography

Cryptography is a technique used for securing confidential data against unauthorized recipient. It prevents determining the content of private messages or any kind of information sent through the internet and/or stored in any kind of storage systems. In other words, cryptography is constructing and analyzing protocols that prevent third parties from reading private messages [18].

Nowadays, security is a preliminary need and it is really challenging especially in network broad systems and technology domains. Data could be hacked in different forms (i.e. text data, audios, images, content of information, and videos). These forms can be intercepted during the communication process between 2 users or at storage systems, so it is more secure to encapsulate them. For this purpose cryptography is the most popular and basic need for data security. [19]

Cryptography algorithms are widely available and used in information security. They are categorized into Symmetric and Asymmetric keys encryption. These algorithms are mainly based on 2 methods, encryption for encrypting data at sender side and decryption for decrypting at the receiver side as shown in figure 3.1. Encryption is the process of encoding a plain text in a way that only authorized users can access and read it. Mainly, encryption uses a mathematical algorithm which transforms plain-text data into cipher-text (encrypted) using an encryption key. In order to read the cipher-text, decryption should be applied. Decryption is the process of converting encrypted text back into plain-text that can be read.

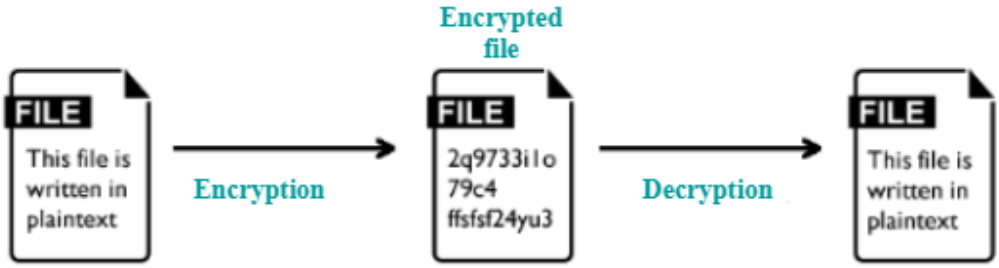

Figure 3.1: Encryption and decryption in Cryptography

The main purpose of cryptography is to achieve as mush as possible security goals especially nowadays with all the improvements in the communication networks.[20]

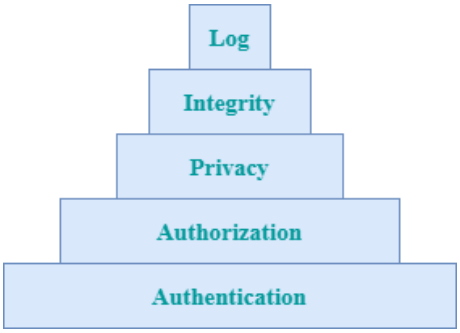

Figure 3.2: Pyramid of security in security systems.

Authentication:

A security system offers the authentication property if it allows to verify a proof which has the purpose to attest the truthfulness of a data or entity attribute.

Authentication should be the most important property in any security system, if it is not strong, all the security measures at the upper layers will not make sense (fig. 3.2). Keeping track of operations which have been performed after authentication (saved in the Log) makes sense, from the security point of view, only if there is the certainty of who performed those operations.

Cryptography can provide two types of authentication services [21]:

1. Integrity authentication: used to verify that no modifications has occurred to the encrypted data.
2. Source authentication: used to verify the identity of the creator of the information.

Authorization:

Authorization is used to provide permission to perform a security function or activity. It is generally granted after the successful execution of an authentication service. This security service is often maintained by a cryptographic service.

Data integrity:

Data integrity provides assurance that the data have not been altered or destroyed in an unauthorized manner like:

- Modification: changing transmitted or stored data.
- Cancellation: data can be blocked before arriving to the destination.
- Replay: data in transit could be transmitted more than one time.

Cryptography mechanisms, digital signatures and message authentication, can be used to detect data modifications either accidental or due to hardware failure which can guarantee data integrity.

Privacy and data confidentiality:

Privacy is the right of individuals to control what information may be collected and stored and by whom and to whom that information may be disclosed. It allows to guarantee that no one can access confidential information without being authorized.

Cryptography is used to encrypt the data to make it unintelligible to everyone other than those who are authorized to view it. To provide privacy and data confidentiality, cryptography is implemented in a way that an unauthorized party will not be able to determine the keys that have been used in the encryption process or have the ability to derive the information without using the correct keys.

3.1.1 Symmetric Encryption

Symmetric encryption cryptography is a branch of cryptography that uses the same key for both operations, encryption and decryption. In symmetric cryptography, the symmetric key is a single secret-key, shared only between the sender and the receiver as shown in figure 3.3.

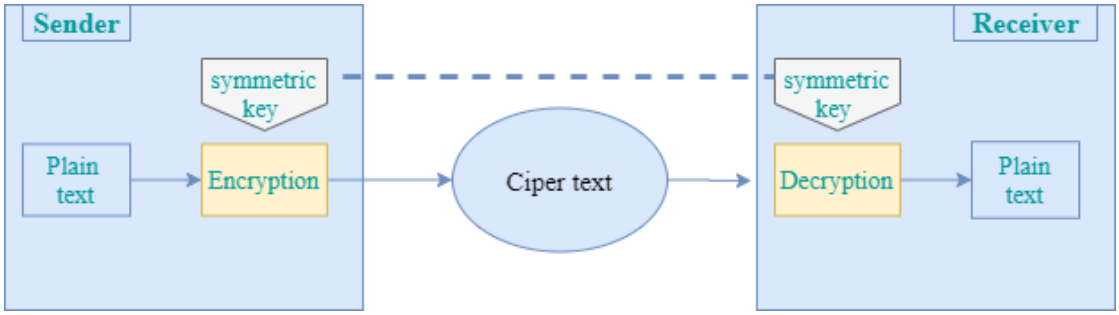

Figure 3.3: Symmetric encryption based on the same key shared between sender and receiver.

	Key length	Block size
DES	56 bits	64 bits
3DES	112 or 168 bits	64 bits
RC2	8 to 1024 bits	64 bits
AES	128, 192, or 256 bits	128 bits
RC5	0 to 2048 bits	1 to 256 bits
RC4	variable	stream

Table 3.1: Key and block size of symmetric algorithms

Symmetric algorithms (table 3.1) are split into two classes:

- Block algorithms: data is split to be encrypted into equal blocks, then they are processed block at a time (e.g. DES, RC2, RC5, AES).
- Stream algorithms: data is encrypted one bit or byte at a time, in data stream (e.g. RC4).

Data Encryption Standard (DES)

DES Algorithm was designed in the 70s and it was the first encryption standard to be recommended by National Institute of Standards and Technology. It uses 64 bits key size with 64 bits block size. DES faced many attacks since it was first

created due to the shortness of the key used. The actual key is 64 bit but the effective key is just 56 bit which made it an insecure block cipher because of the strength of the brute force attack which is equal to 2^{56}.

3DES is an enhanced version of DES to solve the vulnerabilities presented in the actual version. It is 64 bit block size with 192 bits key size. It is similar to the original DES, but the encryption method is applied 3 times to increase the level and the average safe time. It is known that 3DES is slower than the other block cipher methods due to the 3 time encryption [22].

Ron's Code 2 (RC2)

RC2 is a symmetric block algorithm developed by cryptographer Ron Rivest. It is 3 times faster than DES and it is implemented at the software level. RC2 is a 64-bits block cipher with a variable key size that range from 8 to 128 bits.

Ron's Code 5 (RC5)

The RC5 algorithm is also developed by Ron Rivest. It was created specifically for Wireless Application Protocol. In RC5 lots of parameters, key length and the basic block size, are variable, but the algorithm works better when block size B is equal to twice the word in the CPU it is running on.

Advanced Encryption Standard (AES)

AES was published first time in the national institute of standard technology and developed by two Belgian cryptographers Daemon and Trijiman. It is a block cipher with variable key length of 128, 192, or 256 bits but usually 256 bit. it encrypts data blocks of 128 bits in 10, 12 and 14 round depending on the key size [22].

Ron's Code 4 (RC4)

RC4 is a stream symmetric algorithm made by Ron Rivest. It is 10 times faster than DES and it uses a variable key length. Stream algorithms work on a stream of data without requiring to split them into blocks. Each bit in the plain-text data stream is matched with a bit in the key flow, and these bits are combined together in some way (i.e. XOR) to get the cipher-text. The key stream must be the same in the encryption and decryption.

3.1.2 Asymmetric Encryption

Asymmetric cryptography is a modern branch of cryptography (known also as public key cryptography) in which the algorithm uses a pair of keys, public and private.

Both keys are used in the cryptographic operations, encryption and decryption or signature creation and signature verification as shown in figure 3.4.

The asymmetric key is made of 2 keys:

- Private key: known only by its owner, and it is kept secret to the rest of the world.
- Public key: spread as widely as possible to the rest of the world.

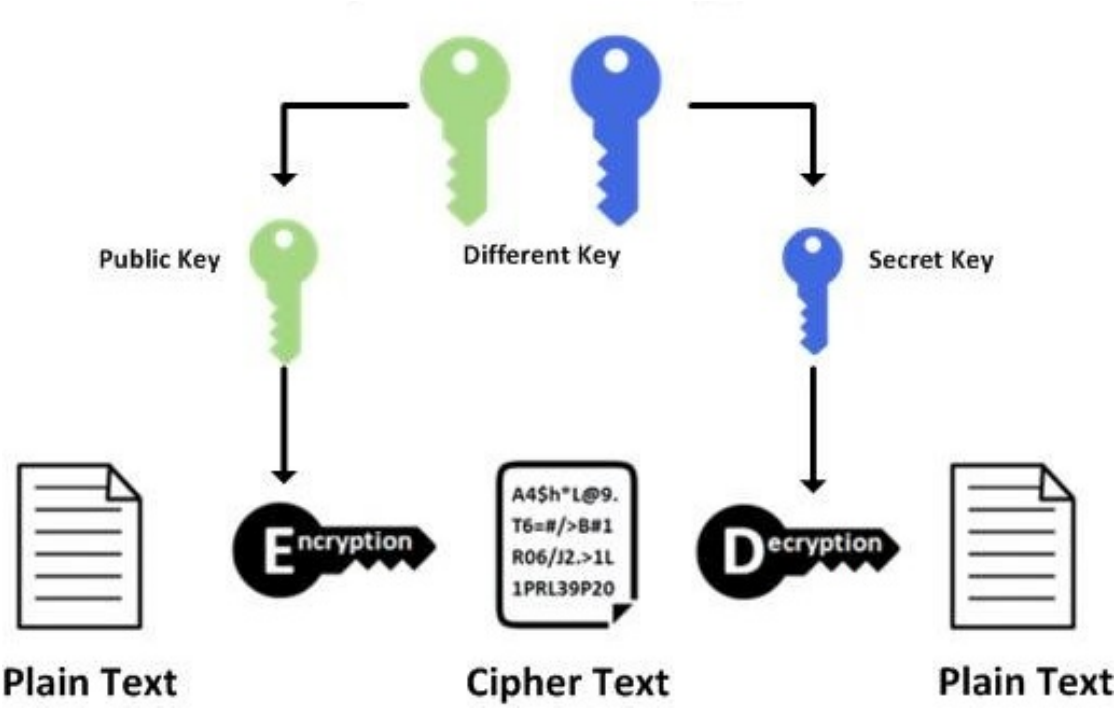

Figure 3.4: Asymmetric encryption based on the different keys [23]

Main asymmetric algorithms:

Diffie-Hellman (DH)

Diffie-Hellman is the first asymmetric encryption algorithm, created by Whitfield Diffie and Martin Hellman. It allows two users to exchange the secret key without any shared secrets. The Diffie–Hellman protocol is a widely used method as a key exchange algorithm. It is based on cyclic groups. A prime of length 2048 bits should be chosen for long-term security [24].

Rivest Shamir Adleman (RSA)

RSA is the most used asymmetric algorithm. It can be used for key exchange, digital signatures, and the encryption of small blocks of data. Nowadays, RSA is mainly used to encrypt the message's hash value or the session key used for secret key encryption. The 512 and 768 bit keys are weak so it is better to use RSA with a key size greater than 1024 for a reasonable level of security.

Elgamal Encryption Algorithm

ElGamal encryption system is an asymmetric key encryption algorithm described by Taher Elgamal for public key cryptography which is based on the Diffie-Hellman key exchange. It provides confidentiality without shared secrets.

Digital Signature Algorithm (DSA)

DSA was suggested and standardized by the National Institute of Standards and Technology. Its focused on the technique of generating and validating digital signatures. DSA can be used to verify that the message was not been changed or altered during transit. The digital signature is the electronic version of the written signature that proves to the recipient that the message was signed by the originator.

3.1.3 Symmetric vs Asymmetric

	Symmetric	Asymmetric
Encryption and decryption	Fast	slow
Key distribution	Difficult	Easy
Processing load	low	high
Complexity	O(Log N)	O(N3)
Inherent Vulnerabilities	Brute Forced, Linear and differential cryptanalysis attack	Brute Forced and Oracle attack
Vulnerabilities cause	weak key usage	Weak implementation
Suitable for amounts of data	large	low
Security Services	Confidentially	Confidentially, integrity, non repudiation
Keys	1 key shared between sender and receiver	2 keys public and private

Table 3.2: Difference between symmetric and asymmetric encryption

Both symmetric and asymmetric agree that the taller the key the higher the security as shown in figure 3.5

Figure 3.5: Key strength of both symmetric and asymmetric algorithms

Table 3.2 and figure 3.6list the main difference between symmetric and asymmetric algorithms.

Parameter	Symmetric Encryption				Asymmetric Encryption	
	DES	3DES	AES	Blowfish	RSA	Diffie-Hellman
Key Used	Same key used for encryption and decryption	Same key used for encryption and decryption	Same key used for encryption and decryption	Same key used for encryption and decryption	Different key used for encryption and decryption	Key Exchange
Throughput	Lower than AES	Lower than AES	Lower than Blowfish	Very High	Low	Lower than RSA
Encryption Ratio	High	Moderate	High	High	High	High
Tunability	No	No	No	Yes	Yes	Yes
Power Consumption	Higher than AES	Higher than AES	Higher than Blowfish	Very Low	High	Lower than RSA
Key Length	56 Bits	112 to 168 Bits	128, 192 or 256 Bits	32 Bit to 448 Bit	>1024 Bit	Key Exchange Management
Speed	Fast	Fast	Fast	Fast	Fast	Slow
Security Against Attack	Brute Force Attack	Brute Force Chosen-Plaintext, Known Plaintext	Chosen- Plain, Known Plaintext	Dictionary Attack	Timings Attacks	Eavesdropping

Figure 3.6: Difference between symmetric and asymmetric algorithms

In CardioFilo, cryptography will be used in order to encrypt the data flow from the back-end to the front-end to guarantee an end-to-end encryption. Even though asymmetric encryption was presented as more secured, symmetric encryption is the better choice for CardioFilo due to the large amount of transmitted data. AES will be used with key length 256bit and can be different for every instance.

3.2 Auth0 Access Tokens

Access tokens are objects that encapsulate the security identity of a process. Generally, they are used to perform security decisions in order to grant authentication

and authorization for the user account. In other words, they contain the identity and the privileges of the user in order to be able to authenticate by comparing the information stored in the security database.

In Auth0, Access Tokens are also used in token-based authentication to allow an application access the API's stored previously [25]. When the user is successfully authenticated and authorized, by providing correct username and password, the application receives an access token which will be used in order to get the data stored in those APIs.

In order to securely and authentically retrieve data from the Auth0 APIs, the access token should be passed in the credentials of the call for those target APIs. It will inform the API that the bearer of the token, the application of this specific user, has been authorized to access the target API and perform the actions he mentioned in the scope (read or write) as shown in figure 3.7.

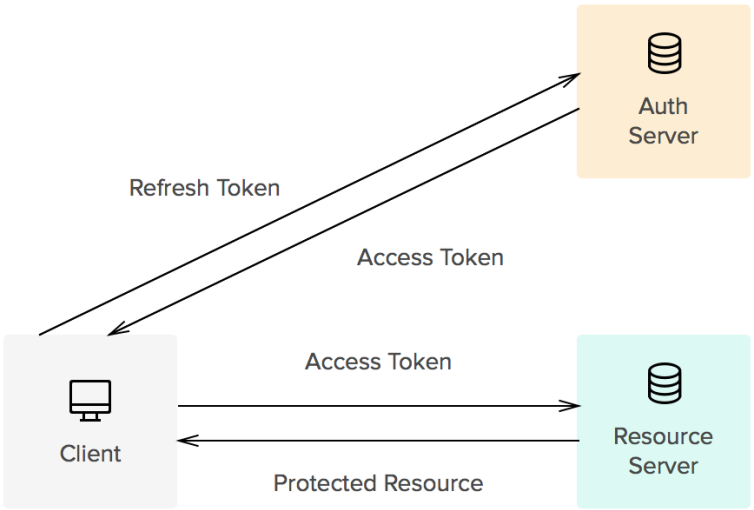

Figure 3.7: Client's Token process in order to retrieve data

Access Tokens have an expiration time which usually is 24 hours. So in order to solve this short time of authentication, Refresh Tokens were presented. Refresh Token is defined as a special token used to obtain a renewed access token. Programmers are able to request a new access token until the refresh token is blacklisted [26].

In Auth0 there are 2 types of Tokens, both have the same purpose but different

functionality:

1. **Opaque Access Tokens :**

   Contains identifiers to information in the server's storage. A call to the server that issued the token should be made in order to validate it.

2. **JSON Web Token Access Tokens :**

   Contains information about an entity in the form of claims. It is not necessary for the recipient to call a server to validate the token because the are self-contained.

3.2.1 Access Tokens In CardioFilo

Opaque Access Tokens (OAT) where used in Cardiofilo in order to maintain authentication and authorization which provides more security for the user and the data he can access using CardioFilo.

The credentials are obtained by building a request the includes the URL of Auth0 and some other scopes which perform an Authentication Callback when it is built. In case of authentication, OnAuthentication will retrieve these credentials that contain the access token and will be save in the credentials Manager for security reasons. Otherwise, onError will return an exception with the reason of failure as shown in the code below.

```
1          @Override
2          protected void onCreate() {
3          lock = Lock.newBuilder(auth0, callback)
4                  .withAudience(URL)
5                  .initialScreen(InitialScreen.LOG_IN)
6                  .setMustAcceptTerms(false)
7                  .closable(true)
8                  .allowShowPassword(true)
9                  .hideMainScreenTitle(false)
10                 .allowLogIn(true)
11                 .allowSignUp(false)
12                 .build(this);
13         }
14
15     private LockCallback callback = new AuthenticationCallback() {
16         @Override
```

```
17          public void onAuthentication(Credentials credentials) {
18              try {
19                  credentialsManager.saveCredentials(credentials);
20              }catch (Exception e){
21              }
22          @Override
23          public void onError(LockException error){
24              showResult(error.getMessage());
25          }
26      };
```

After the access token is obtained, it is used to authorize getting data from the APIs by including it in the header of the request.

```
1       final Request.Builder reqBuilder = new Request.Builder()
2               .get()
3               .url(url)
4               .addHeader("Authorization", "Bearer " + accessToken);
```

3.3 Unit Tests

Program testing is the most practiced means of verifying that a program possesses the features as specified. Unit testing is a dynamic approach of verification in which the code is executed with test data to check the presence (or absence) of required features[27], and to make sure that all the code paths are working correctly. In addition to that, by using unit tests, developers can :

- Identify the defects before integration and fixing them improves the quality of the code.
- Increase security while changing and updating in future due to the already created unit tests.
- Simplify the debugging process.
- Reduce the cost.
- Find bugs early.

The earlier the defects are found, the less impact they have and the less it costs to fix them, therefore it is more practical to apply unit test activities earlier in the software development life cycle. Usually unit tests are created by the developer who wrote the code or someone who has full knowledge of its content. In CardioFilo, unit testing was performed by Abinsula team after the end of each task.

3.4 Secure Coding

Secure coding is developing a software that provides security against the accidental vulnerabilities. Defects, bugs, and logic flaws are usually the primary cause of exploited software vulnerabilities[28]. Security professionals discovered that most of vulnerabilities are originated from a small number of common software programming errors after analysing thousands of reported vulnerabilities.

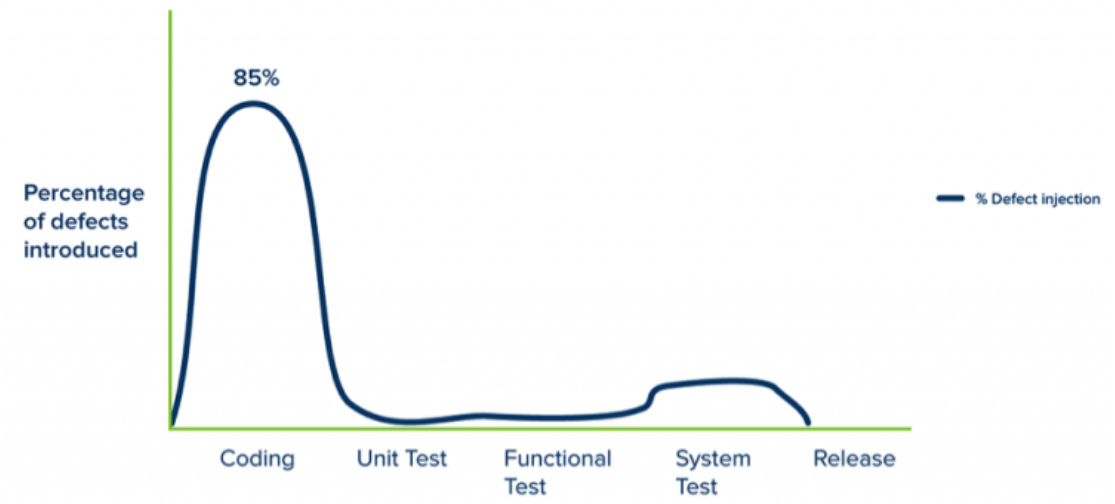

Figure 3.8: Increasing cost of bugs and defects at each phase of software development[29].

As figure 3.8 shows, 85 percent of defects are introduced into the software at the coding phase which indicates the importance of identifying the insecure coding that leads to errors. This can be achieved by educating programmers and developers on secure alternatives and taking a proactive step to reduce or eliminate vulnerabilities in any software before the deployment phase.

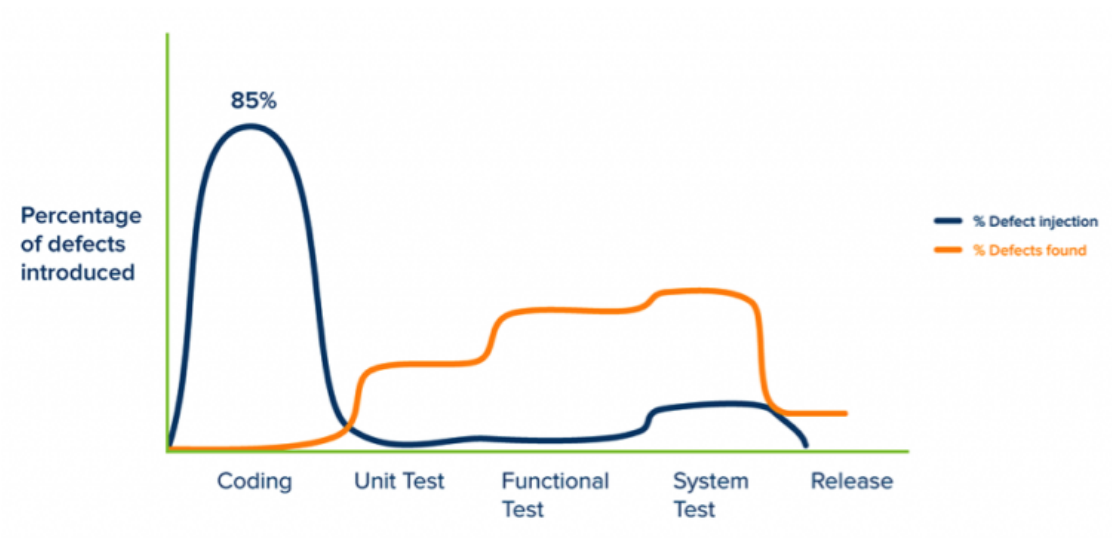

Figure 3.9: Defects found at each phase of software development[29].

As the testing phases advance, the cost of fixing the defects increases same as detecting them, as shown in figures 3.9 and 3.10. Bugs and defects are more likely to be detected in the late cycle of testing phases, and it will cost more to recover from them. Defects will be fundamentally ingrained in the code and this explains the dramatic increase in the cost shown in figure 3.10.

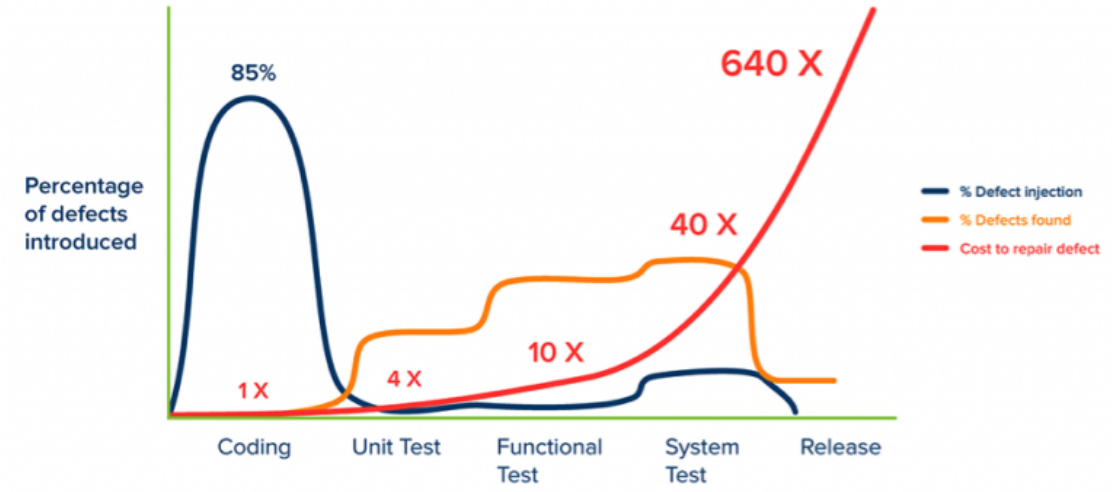

Figure 3.10: Cost to repair defects at each phase of software development[29].

To mitigate potential security risks while writing the code, programmers can take into consideration these aspects[17]:

- Enforce secure communication.
- Use implicit intents and non-exported content providers.
- Apply signature-based permissions.
- Ask for credentials before showing sensitive information.
- Add a network security configuration.
- Share data securely across apps.
- Store private data within internal storage.
- Use external storage cautiously.
- Check validity of data.
- Store only non-sensitive data in cache files.
- Use SharedPreferences in private mode.

Secure coding was properly taken into consideration while developing CardioFilo. Private SharedPreferences were used to save access tokens in order to securely share them across the activities of the app. All other private data where stored in internal storage XML files. Other than that, all data inserted by the user were validated before sending them to the server, mainly to store the correct values and prevent SQL injection. If there is any incorrect value or unexpected character, the app will show a toast with some information of the expected ones as shown in figure 3.11.

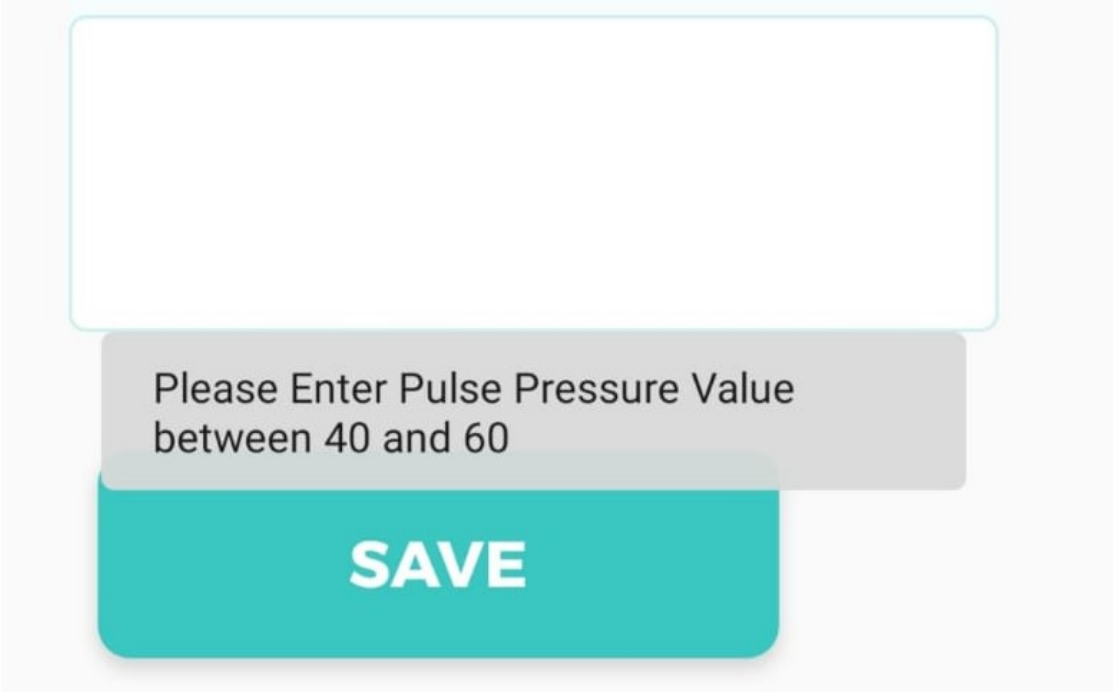

Figure 3.11: Toast example that shows the expected value to be inserted

Chapter 4

Final Mobile Application UI

In this thesis work, the coding part of CardioFilo mobile application was developed. All the design parts were discussed with Abinsula's graphic team and the back end-part with the back-end team.

After the designs were done and uploaded to Zeplin, the developing of the front-end was started. The front-end was based on combination between layouts (Relative and Linear), each layout represents a part or an object of the main layout. The code below is an example of how the layout was divided in approximately all the presented activities. One main Relative Layout that was encapsulated in a scroll view in order to fill and make the activity scroll-able. Inside this Relative Layout, Linear Layouts were added based on the objects that will be shown in this specific activity.

```
<RelativeLayout...>

    <ScrollView
        android:layout_width="fill_parent"
            android:layout_height="fill_parent">
            <RelativeLayout
            android:id="@+id/form_layout"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content">



        <LinearLayout
            android:id="@+id/layout_enterwreight"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_below="@+id/layout_header"
```

```
18              android:layout_marginLeft="36dp"
19              android:layout_marginTop="28dp"
20              android:layout_marginRight="36dp"
21              android:orientation="vertical">
22
23
24              <TextView
25                  android:id="@+id/Dialyrecords_tv"
26                  android:layout_width="wrap_content"
27                  android:layout_height="wrap_content"
28                  android:layout_marginLeft="50dp"
29                  android:fontFamily="@font/montserrat_bold"
30                  android:lineHeight="26sp"
31                  android:text="@string/enternewWeight"
32                  android:textColor="@color/Cardio_titles"
33                  android:textSize="20sp" />
34          </LinearLayout>
35
36          </ScrolView>
37      </RelativeLayout>
38  </RelativeLayout>
```

As for the back-end part, the work was mainly connecting CardioFilo's mobile app with the database created by Abinsula, in-addition to that, connecting all the activities with each other and some other functionlaties that are performed by the app. As discussed in the earlier sections, lots of tools were used in order to successfully send and retrieve data. The codes listed below are the codes of the main back-end operations(e.g. login, logout, request data, store data).

The first code represents the login operation. A new Auth0 is created and built with some chosen activities.

```
1   auth0 = new Auth0(this);
2         auth0.setOIDCConformant(true);
3         credentialsManager = new SecureCredentialsManager(this, new
      AuthenticationAPIClient(auth0), new SharedPreferencesStorage(this)
      );
4
5         lock = Lock.newBuilder(auth0, callback)
6                 .withAudience(getResources().getString(R.string.
      Authentication_url))
7
8                 .initialScreen(InitialScreen.LOG_IN)
9                 .setMustAcceptTerms(false)
10                .closable(true)
11                .setShowTerms(false)
```

```
12                  .allowForgotPassword(false)
13                  .allowShowPassword(true)
14                  .hideMainScreenTitle(false)
15                  .allowLogIn(true)
16                  .allowSignUp(false)
17                  .build(this);
18          startActivity(lock.newIntent(this));
```

When pressing the sign-out button in the main activity (fig. 4.2) simple call-back to Auth0 is launched, as showed in the code below, which in case of success, the onSuccess function will start by clearing all the login credentials saved while the login process.

```
1     private void doLogout() {
2         WebAuthProvider.logout(auth0)
3                 .withScheme("demo")
4                 .start(this, logoutCallback);
5     }
6     private VoidCallback logoutCallback = new VoidCallback() {
7         @Override
8         public void onFailure(Auth0Exception error) {
9             showResult("Log out cancelled");
10        }
11
12        @Override
13        public void onSuccess(Void payload) {
14            credentialsManager.clearCredentials();
15            showResult("Logged out!");
16        }
17    };
```

The code below shows the API call for retrieving the blood pressure presented in the overview (Fig.4.3). When the response is successful, the data will be retrieved in a form of a Json object which contains all the requested data. After that the data will be fetched and presented in its specific place in the activity to be shown to the user.

```
1
2     final Request.Builder reqBuilder = new Request.Builder()
3                 .get()
4                 .url(url)
5                 .addHeader("Authorization", "Bearer " + accessToken);
6             OkHttpClient.Builder builder = new OkHttpClient.Builder()
      ;
```

```
7             builder = IClient.configureToIgnoreCertificate(builder);
8             OkHttpClient client = builder.build();
9         Request request = reqBuilder.build();
10        client.newCall(request).enqueue(new Callback()
11            @Override
12            public void onResponse(Call call, Response response)
    throws IOException {
13                System.out.println("success");
14                try {
15                    String jsonData = response.body().string();
16                    if (response.isSuccessful()) {
17                        JSONObject OverVIreJSONobj=newJSONObject(
    jsonData);
18                        JSONObject jsondataValues = OverVIreJSONobj.
    getJSONObject("data");
19                // get all the data and show them
20                        }
```

A different API call is used in order to store data in the database. Due to security reasons a part of the code will not be presented. The code below shows the main part of the API request in order to store the data. A Json object is created containing all the data to be sent in a specific format. Then the request is build based on the URL and the access token in order to be authorized. A response will be retrieved that shows if the data was successfully stored.

```
1         JSONObject PostData = new JSONObject();
2         RequestBody PostD = RequestBody.create(MEDIA_TYPE, PostData.
    toString());
3         final Request request = new Request.Builder()
4                 .url(url)
5                 .post(PostD)
6                 .addHeader("Authorization", "B " + accessToken)
7                 .header("Accept", "application/json")
8                 .header("Content-Type", "application/json")
9                 .build();
10        OkHttpClient.Builder builder = new OkHttpClient.Builder();
11        builder = IClient.configureToIgnoreCertificate(builder);
12        OkHttpClient client = builder.build();
13             client.newCall(request).enqueue(new Callback() {
14            @Override
15            public void onFailure(Call call, IOException e) {
16            }
17            @Override
18            public void onResponse(Call call, Response response)
    throws IOException {
19            }
```

Figures from 4.1 to 4.11 represent the final activities of the CardioFilo mobile application developed in this thesis.

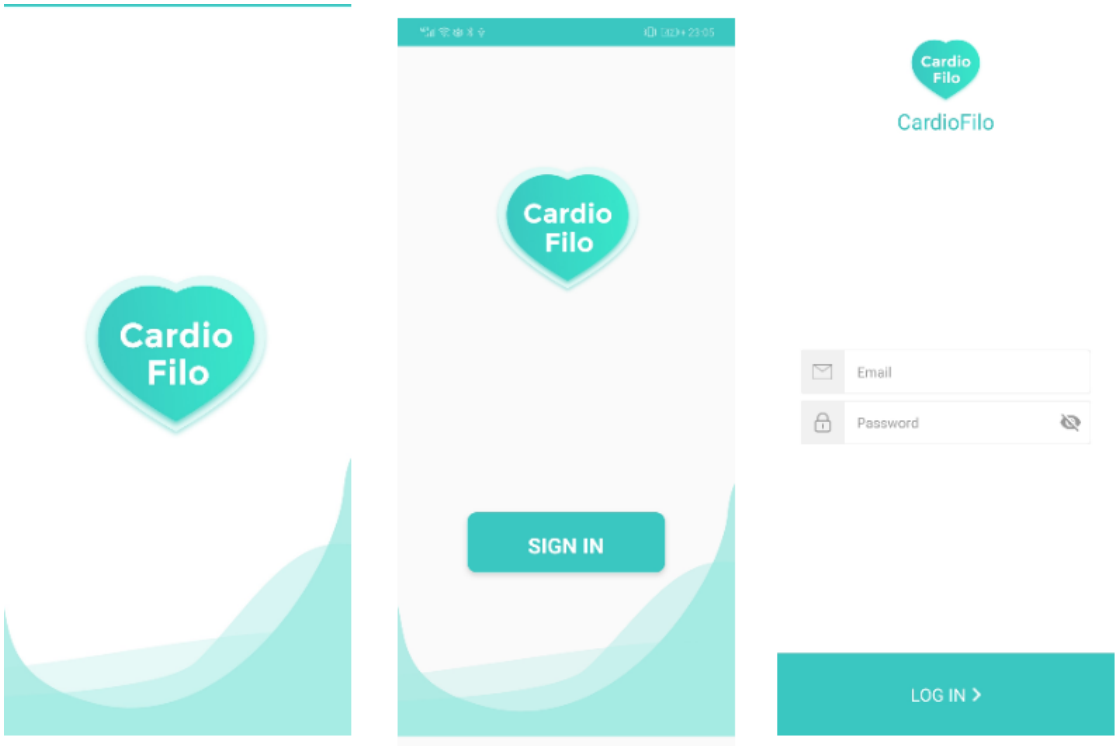

Figure 4.1: Splash screen and login

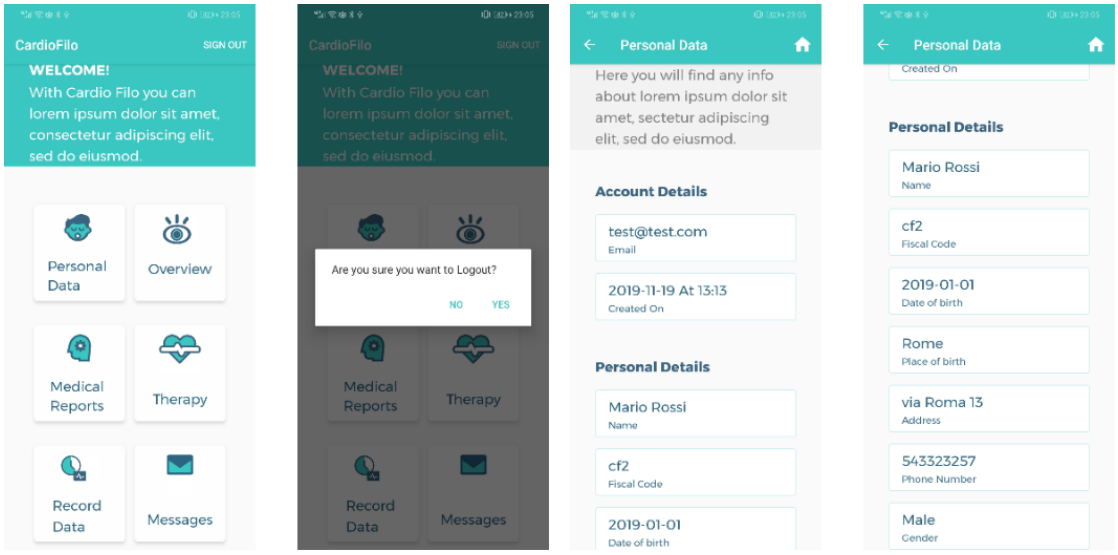

Figure 4.2: Main activity

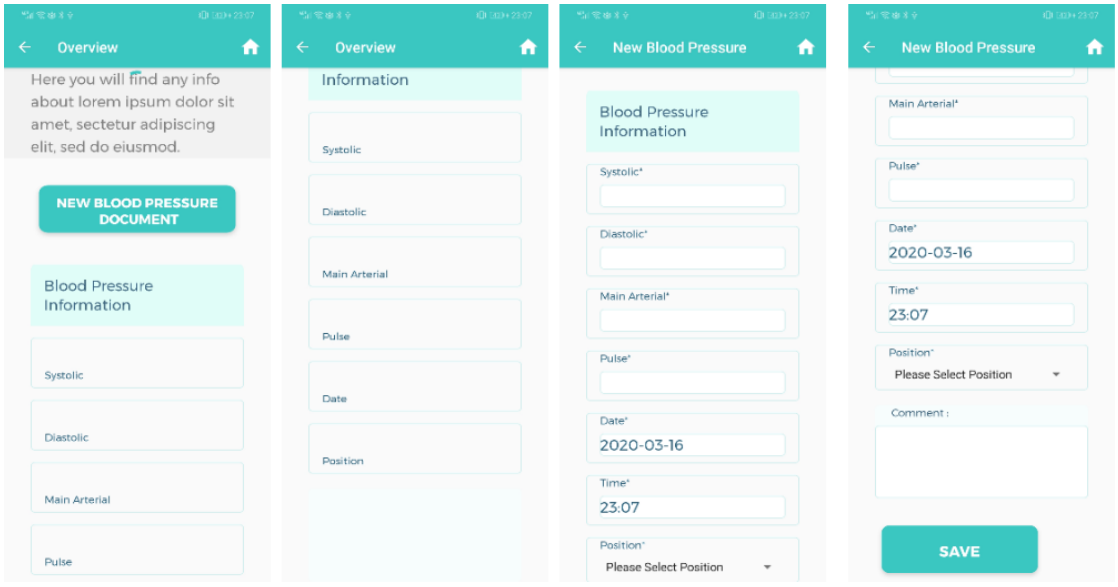

Figure 4.3: Overview

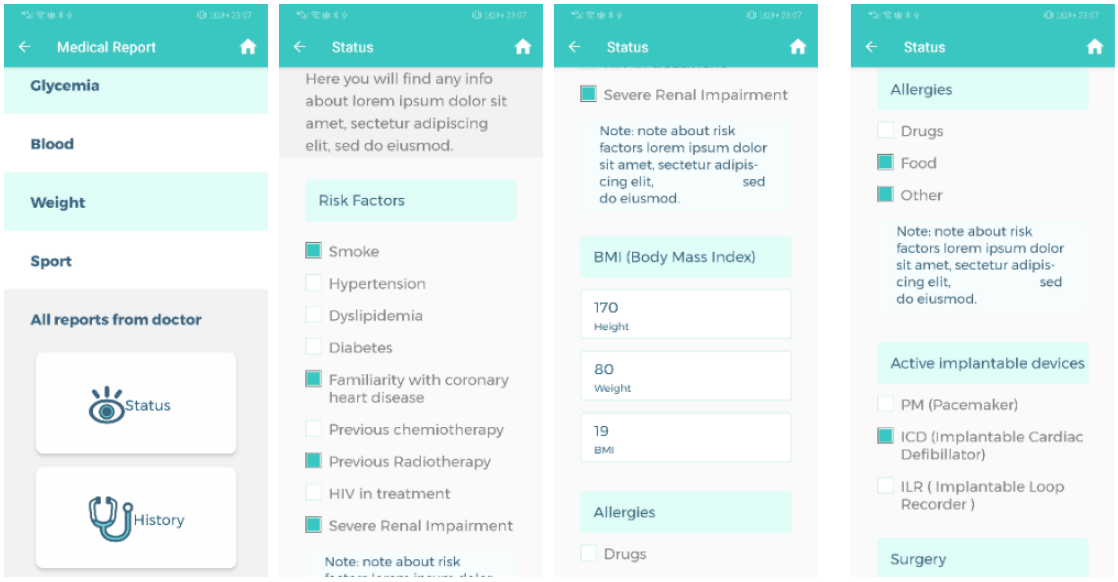

Figure 4.4: Status and history

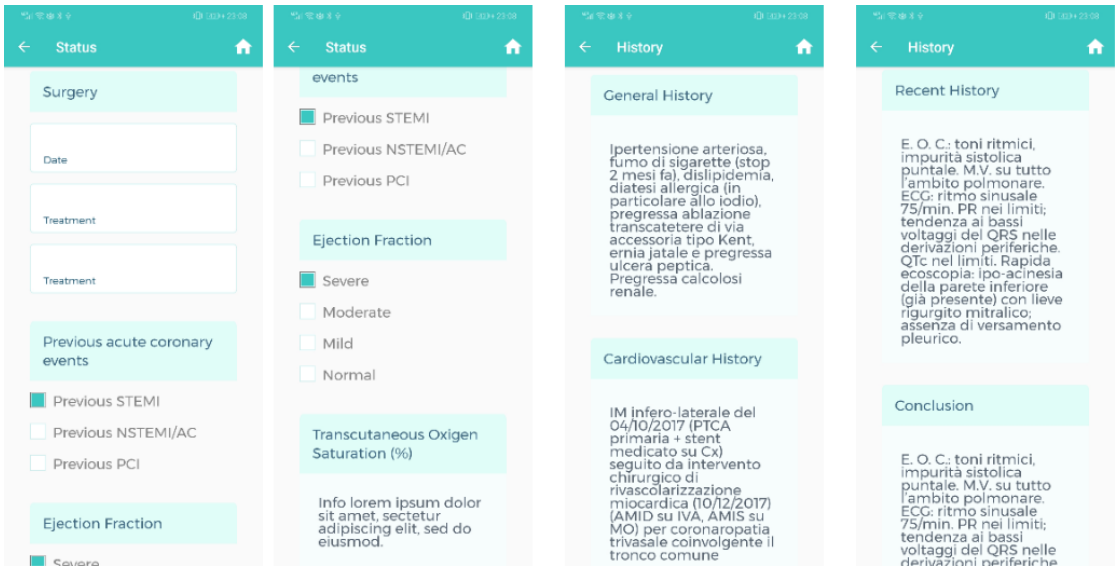

Figure 4.5: Status and history

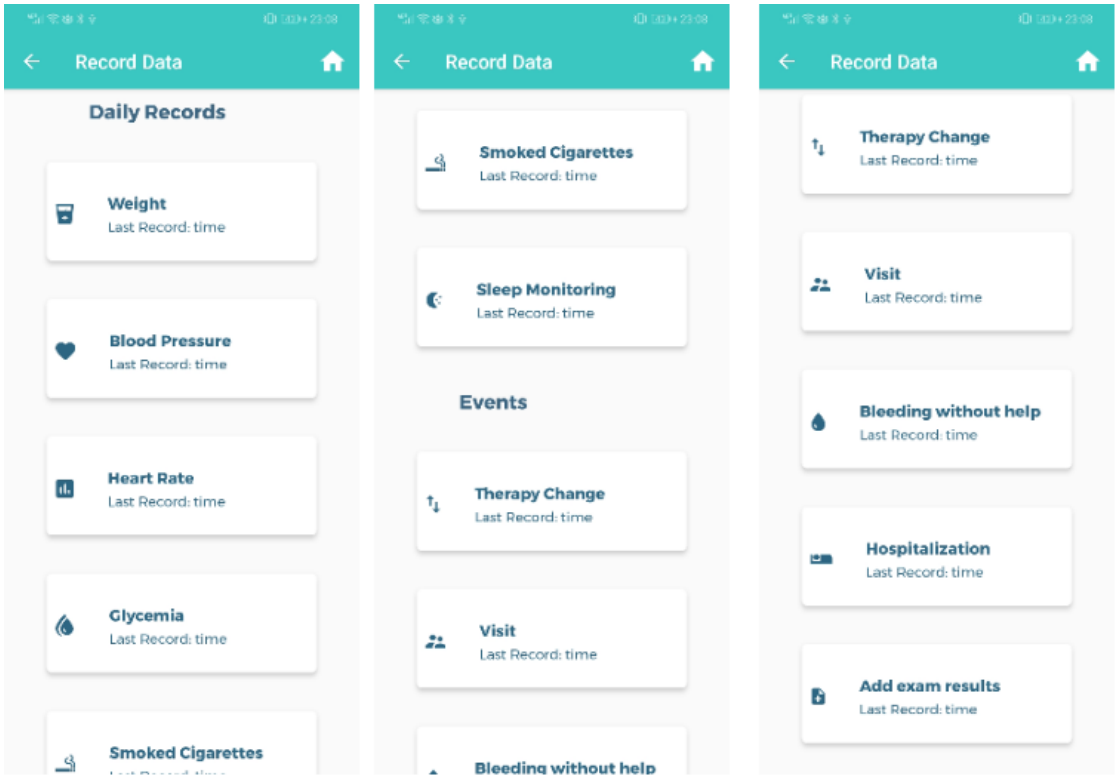

Figure 4.6: Record Data

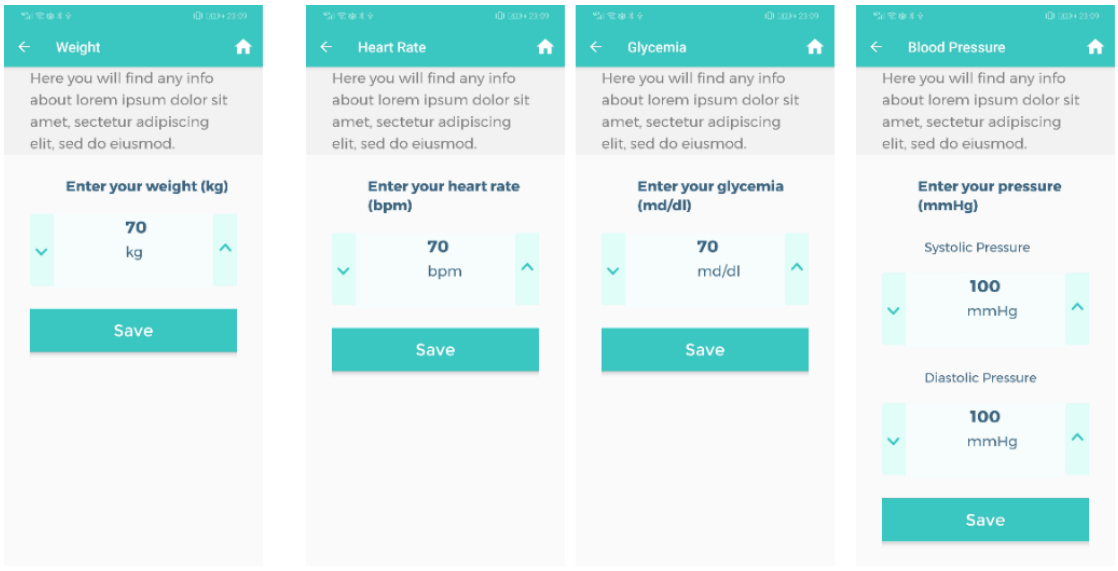

Figure 4.7: Record Data

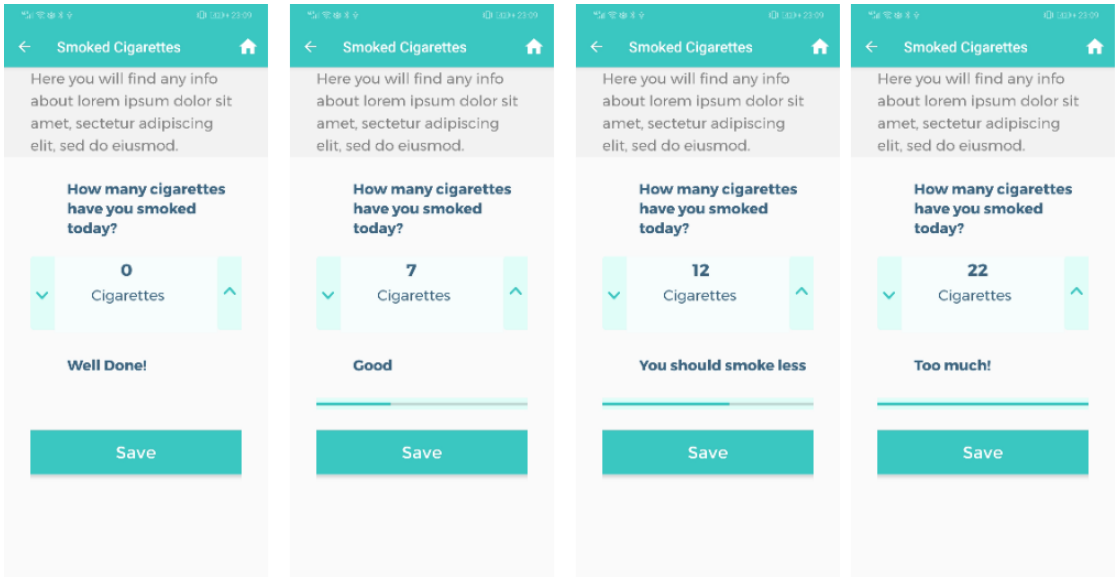

Figure 4.8: Record Data

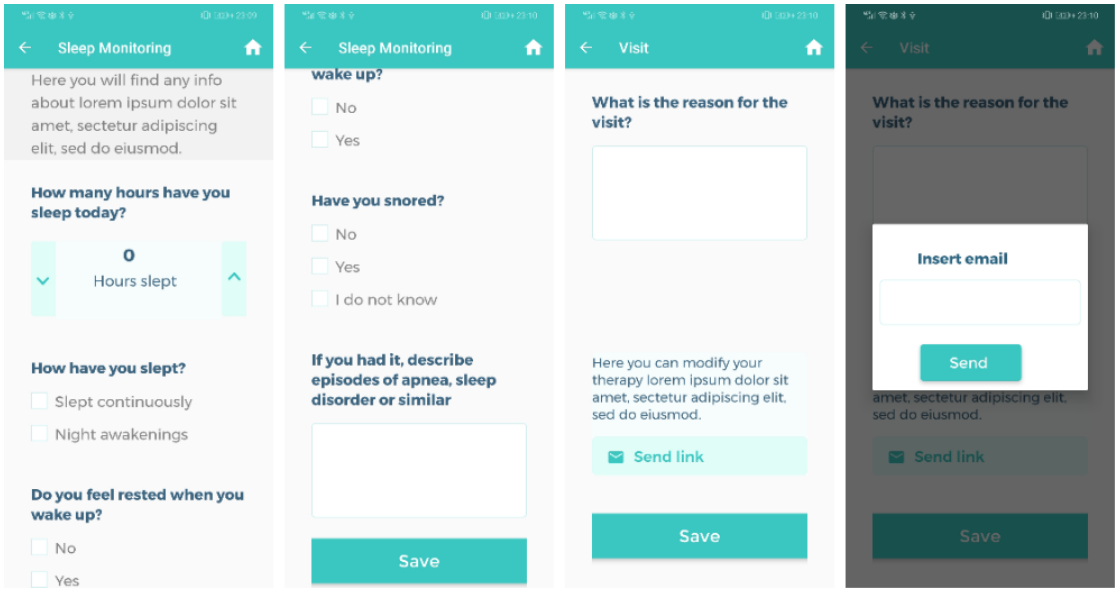

Figure 4.9: Record Data

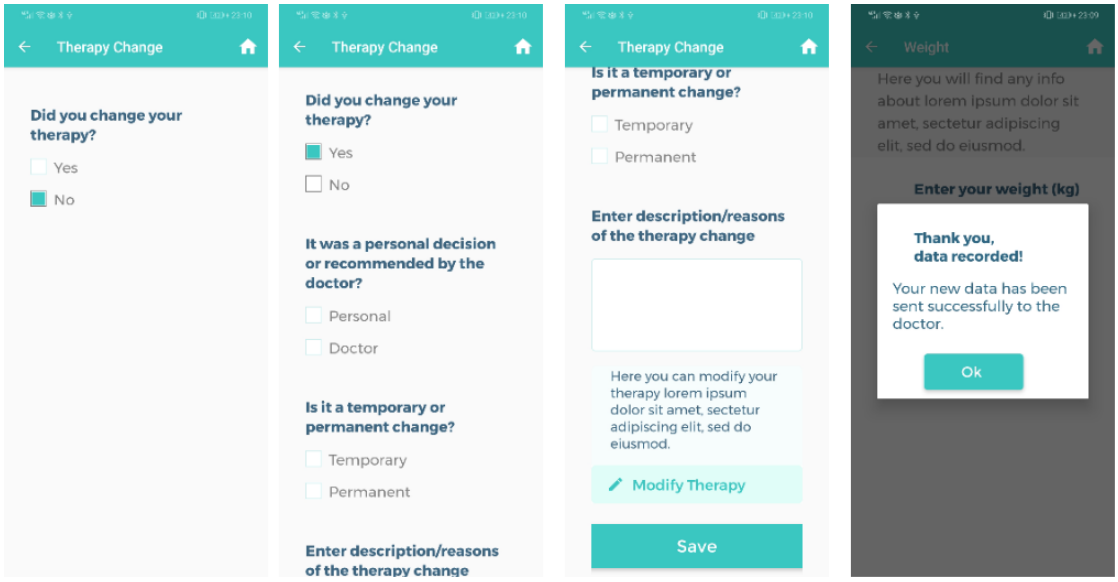

Figure 4.10: Record Data

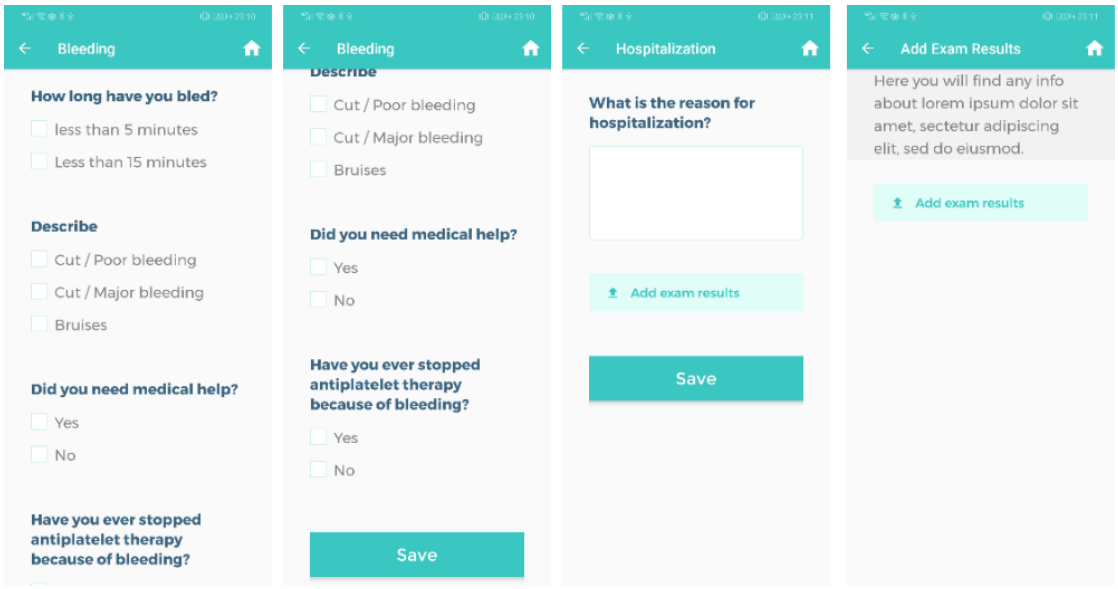

Figure 4.11: Record Data

Chapter 5

Future Work

5.1 Application improvements

5.1.1 Languages

Nowadays, mobile applications spread all over the world once they are uploaded to the store, where any user can download theirs: thus means different cultures, minds, and most important languages. Even though CardioFilo is meant to be used by specific users, this does not deny the fact that it should contain more than one language.

For now CardioFilo comes only in English, but adding other languages was always considered while developing it. A specific XML file was created which contains all the application's titles, comments, notes, and strings used in it, and that will make it easier to switch to newly added languages. A new XML file is created for each new language. These files contain the translated version of the main XML file, each one with its according language. When the user selects his preferred one, the application will load the appropriate resources as shown in figure 5.1.

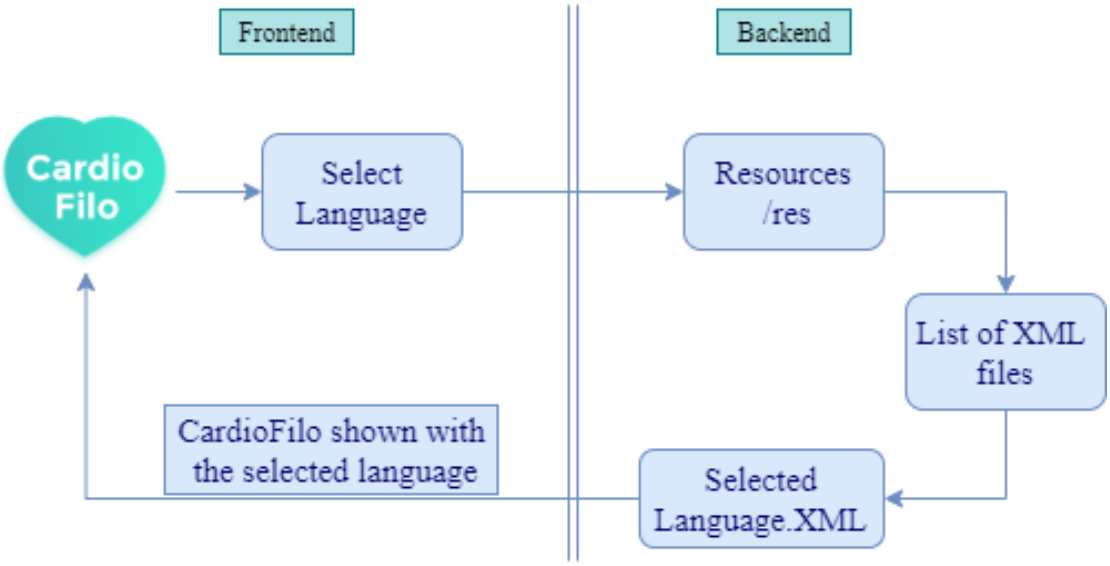

Figure 5.1: Mechanism for changing the application's language.

5.1.2 Notifications

One of the properties of CardioFilo is to show the data uploaded by the doctor concerning the patient's situation. For now the user has to open the application in order to check for new notifications. As a future update, a notification will be sent to the user's phone whenever the doctor submits one of these results:

- Medical reports.
- Risk factors.
- New discovered allergies.
- Surgery dates.
- Therapy dates.
- Messages from the doctor.

The application will also raise some notifications to remind the user to take his pills, check upcoming therapy dates, and also to update some information like blood pressure if it has been a time since the last one.

5.2 IOS Application

Google Android and Apple IOS have 98 percent of the global market share for operating systems. Statistics in figure 5.2 shows that 74.4 percent of the mobile

operating systems is dominated by Android [30], due to the fact that it is shared among different mobile companies (Samsung, Huawei, LG, OnePlus,..), and the remaining 24.76 percent are by IOS, which is huge compared to the number of users among the world.

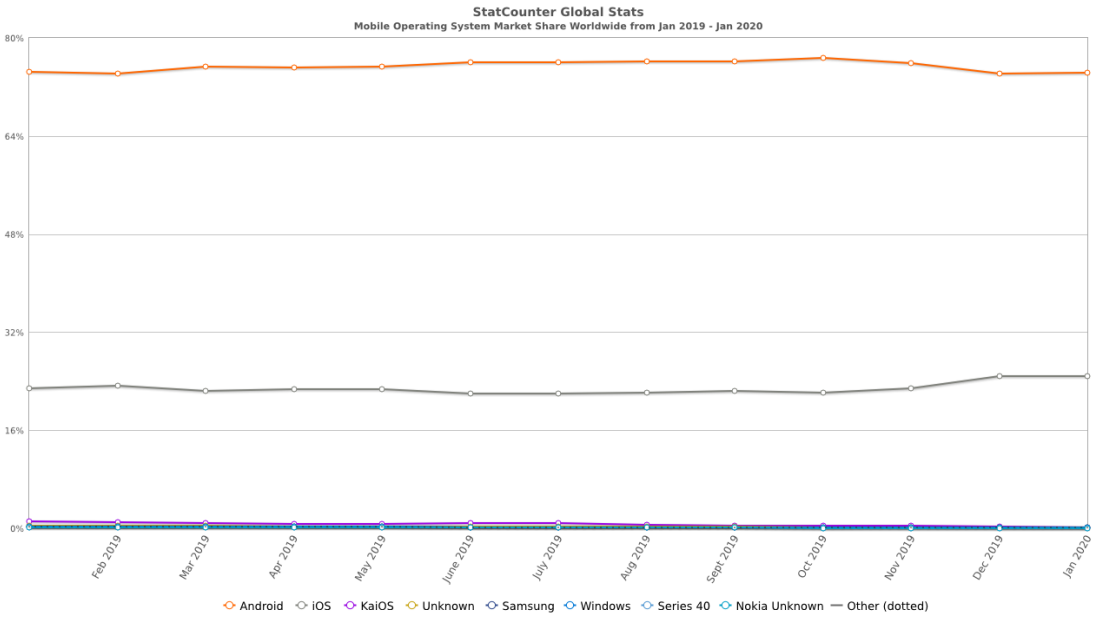

Figure 5.2: Mobile Operating System Market Share Worldwide. [30]

In these days, developing both Android and IOS mobile applications is a must. Even though Android users are obviously many more, still there is a huge percentage that uses IOS. Some developers tend to use Cross-platform development for developing both Android and IOS applications due to the less time-consuming and cost. These platforms allow the usage of one code base which is then compiled to be used as native application for both platforms. Developers should be familiar with CSS, HTML, and Java script in order to use these cross-platforms. Beside that, they should also use frameworks which are the most important tools in cross platform app development like React-native, Xamarin, Ionic, PhoneGap, and Flutter.

Apple IOS, same as android, has its own programming language, libraries, and platforms. Swift is the primary programming language for IOS mobile application development which is created by Apple. It supports the core concepts associated with Objective-C, the first programming language used by Apple, but in a safer way.

Developing an IOS application for CardioFilo is intended to be the next step. It will contain the exact functionalities as the Android's version.

5.3 Doctor's application

As presented before, doctors use the web application in order to monitor the situation of their patients. Whenever the doctor wants to check some information about a specific patient, he has to open the web application. Developing a mobile application for doctors will make that easier and more updated after the notification's feature is added.

Doctor's application will be in read only mode. It is similar to the patients app with some extra features like searching for a patient or contacting him when something urgent occurs. Doctors can search for a specific patient and check his personal data, medical records, risk factors, history, status, therapy plans, and of course notifications.

Some mock-ups are presented as an overview of the functionalities that the application can perform [1]. Figure 5.3 shows the login page where the doctor can login in using the same credentials as the web application. After that the main activity will start showing a button that will view the patient's filtering activity where he can search for a specific patient or more, using the search bar or by checking the features he is interested in (figures 5.4 and 5.5).

Figure 5.6 shows the list of patients that is retrieved after filtering. The doctor can choose one of them in order to see the medical records, which show the situation of that patient like personal data, risk factors, status, therapy plan, sleep monitoring and overview (figures 5.7, 5.8, 5.9, and 5.10).

Doctors can also check and receive notifications when the patient change something in their records. These notifications will be shown in the Recorded events activity 5.11. Doctors can contact the patient by email, text message, or by calling them if there is something urgent. Taking into consideration the huge number of patients which means huge number of notifications, doctors can delete handled notifications and as a future adjustment maybe select what kind of notification they want to receive.

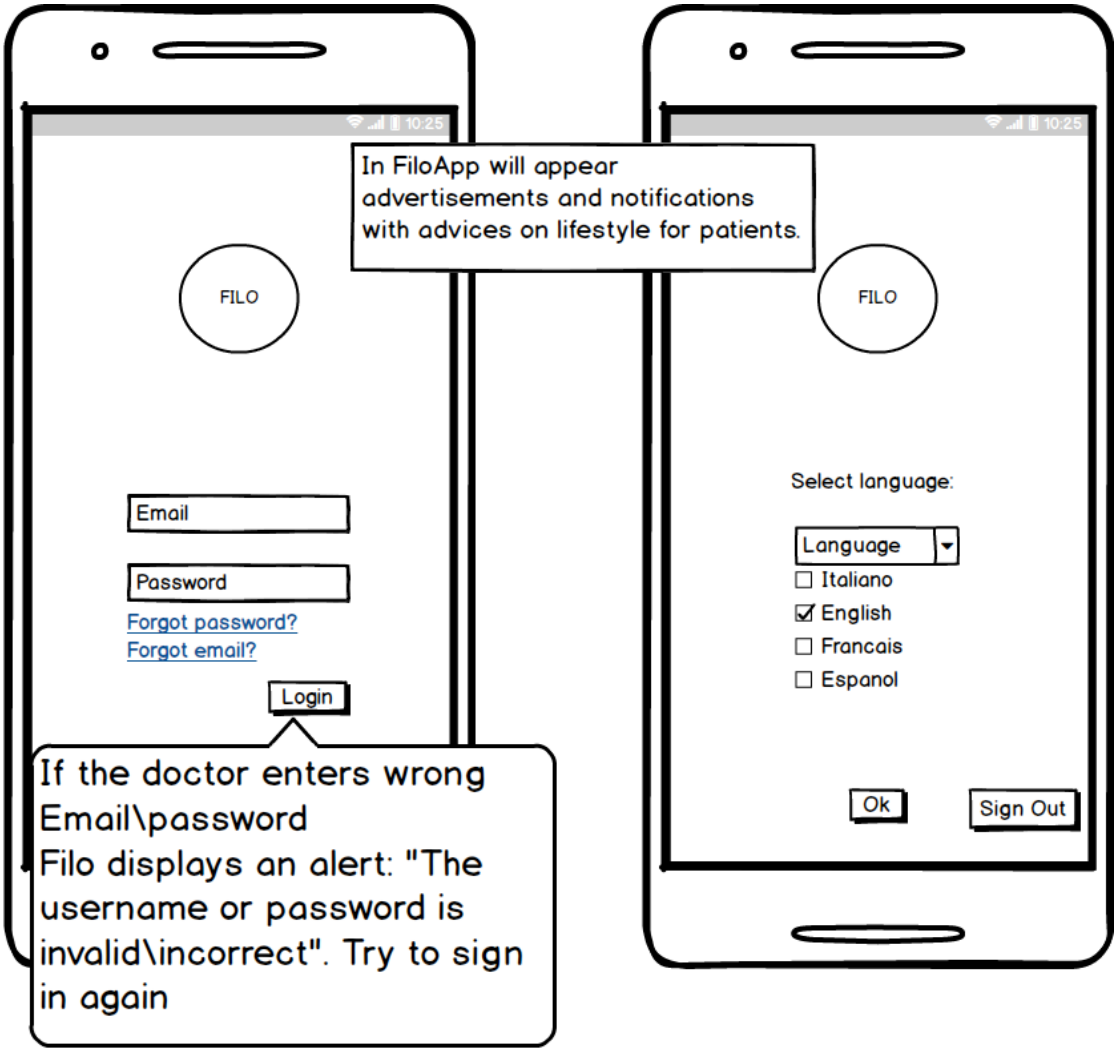

Figure 5.3: Login In activity.

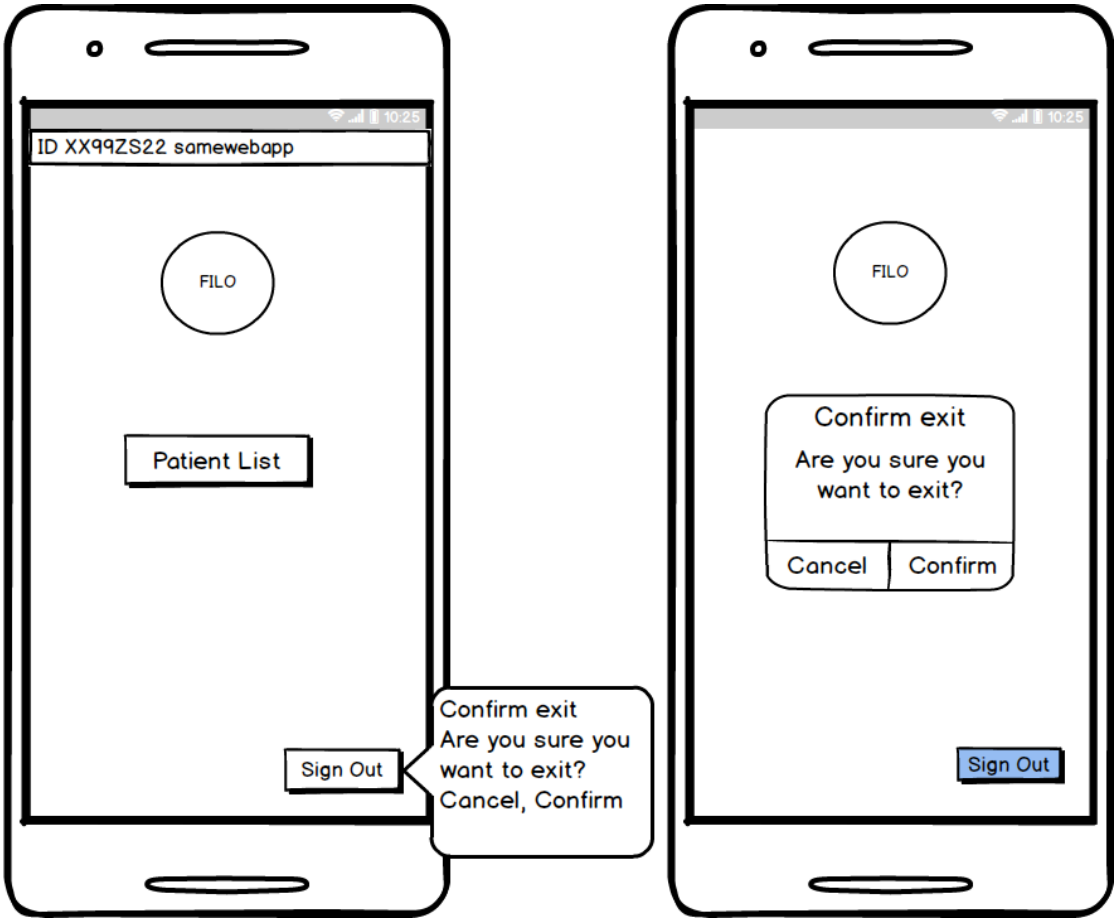

Figure 5.4: Home Page.

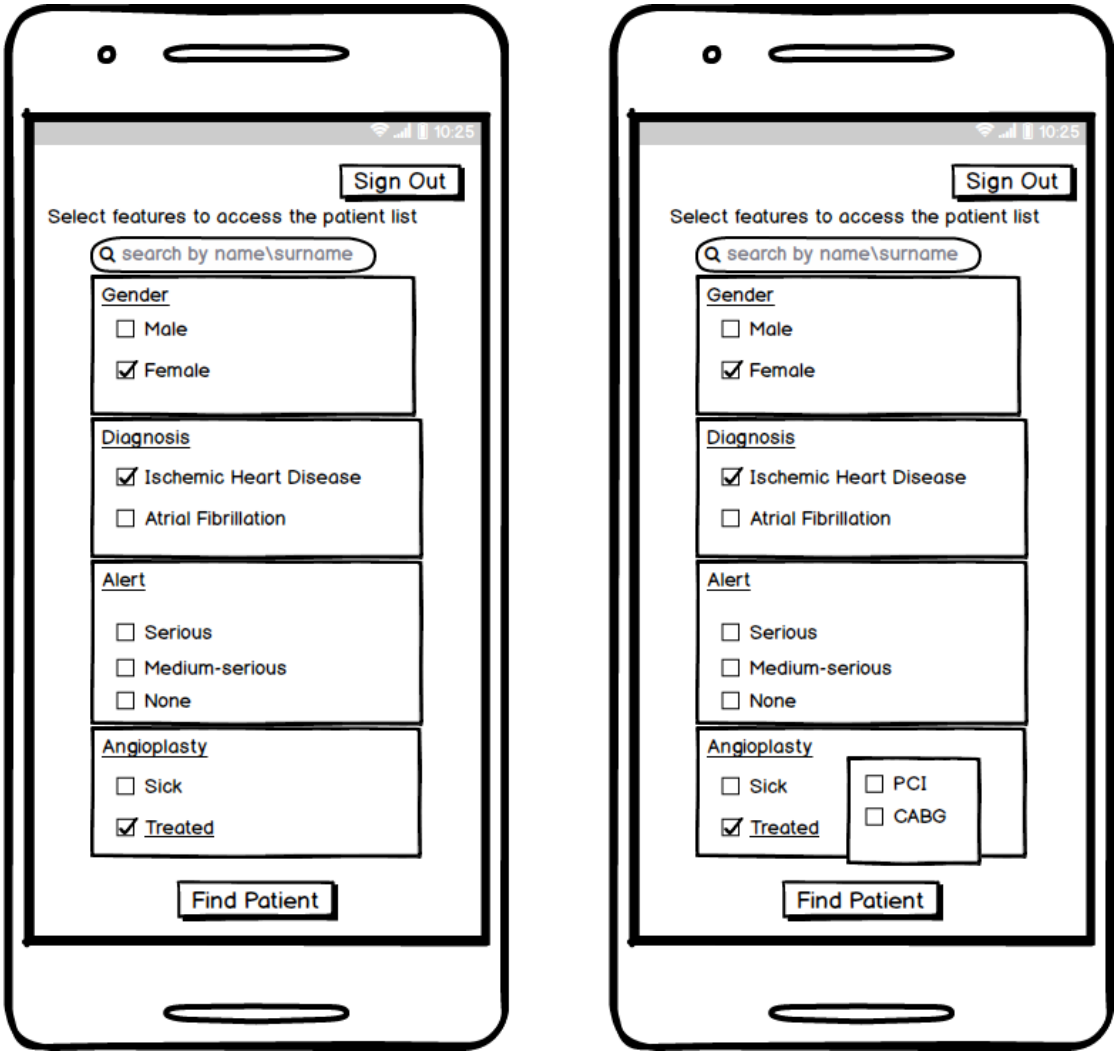

Figure 5.5: Patient's list filtering and searching for specific one.

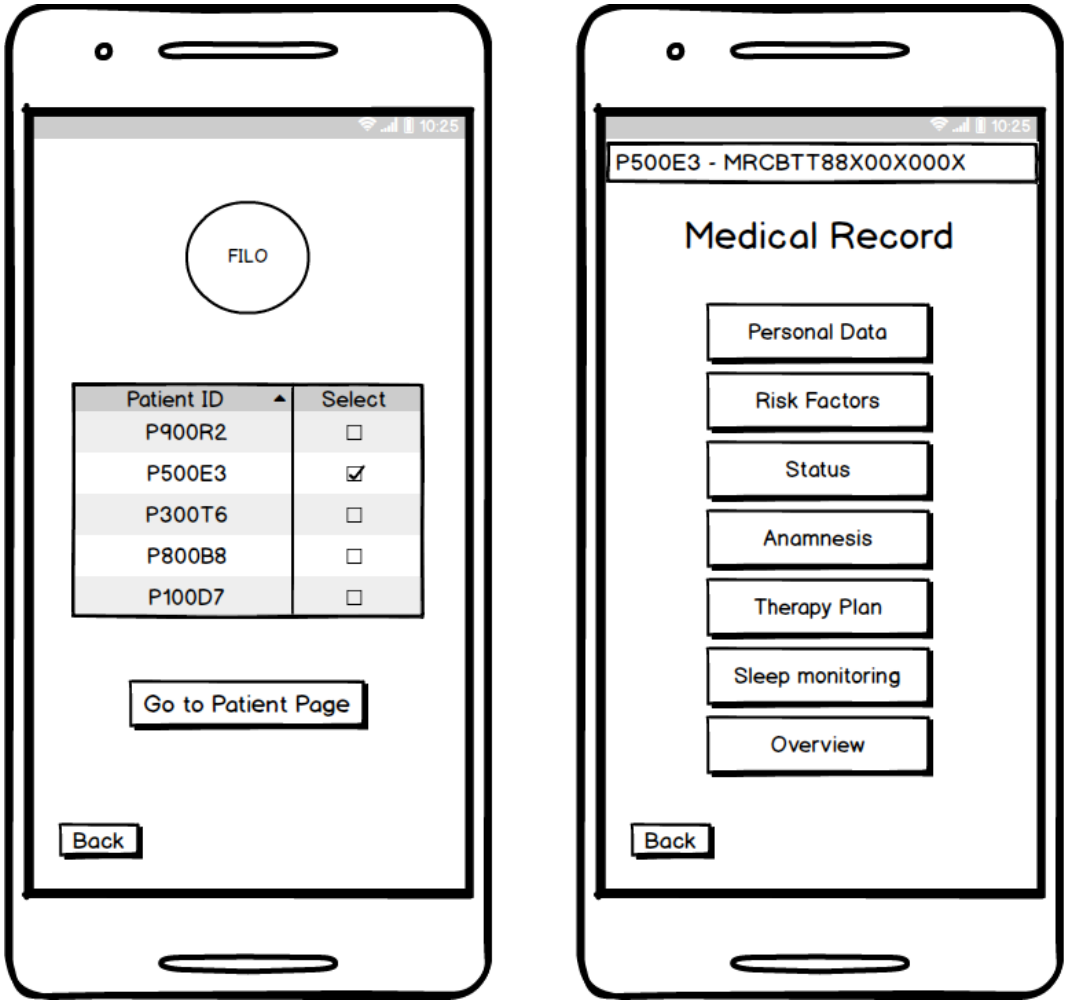

Figure 5.6: Patient's Data.

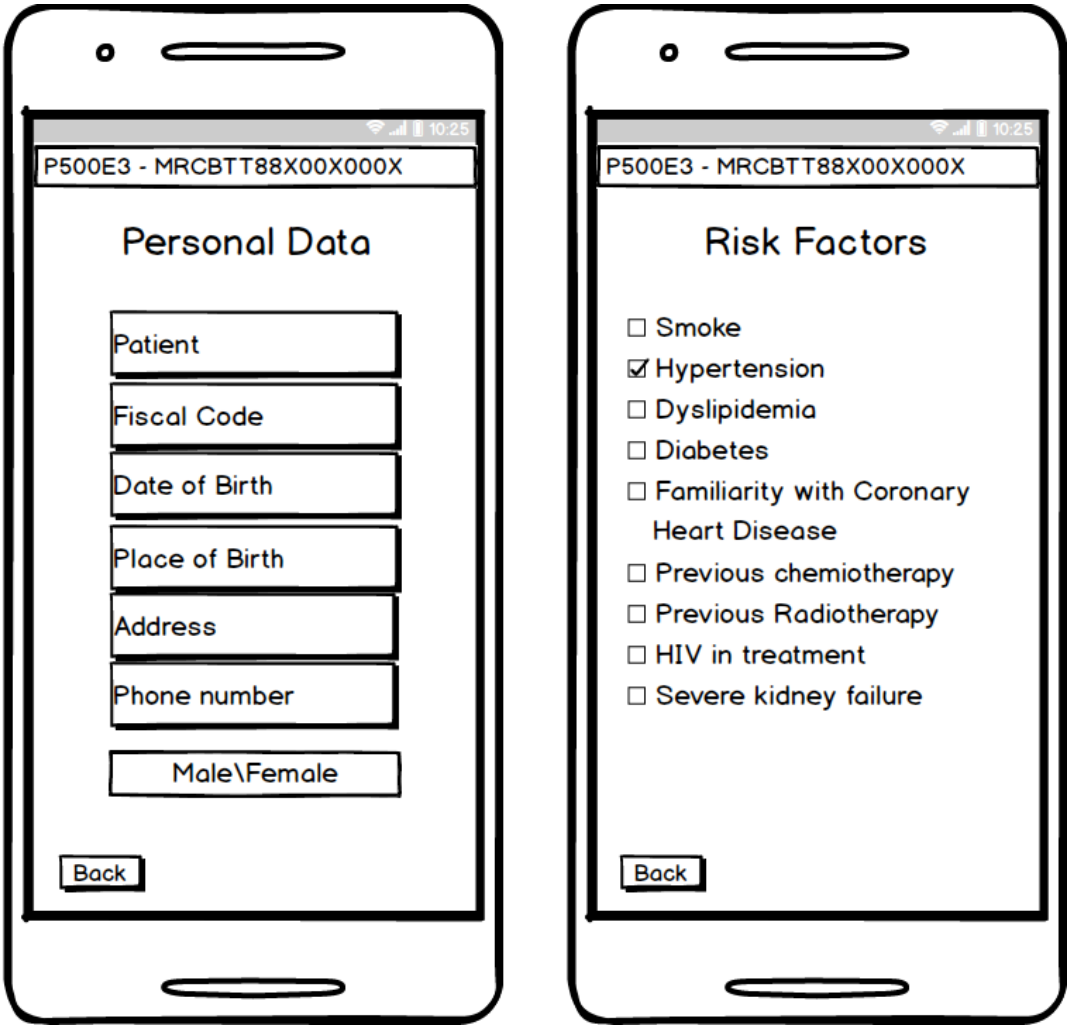

Figure 5.7: Patient's Data.

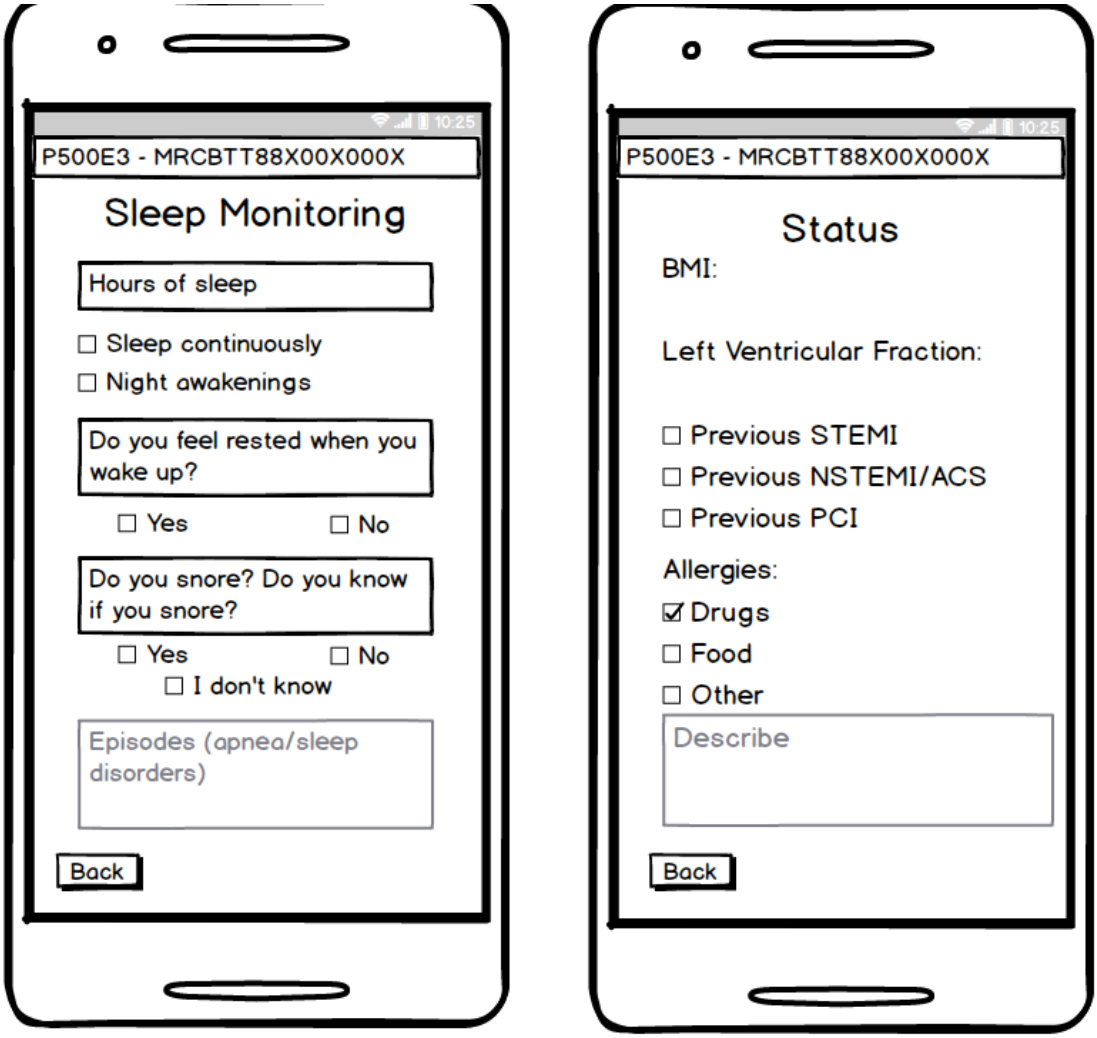

Figure 5.8: Patient's Data.

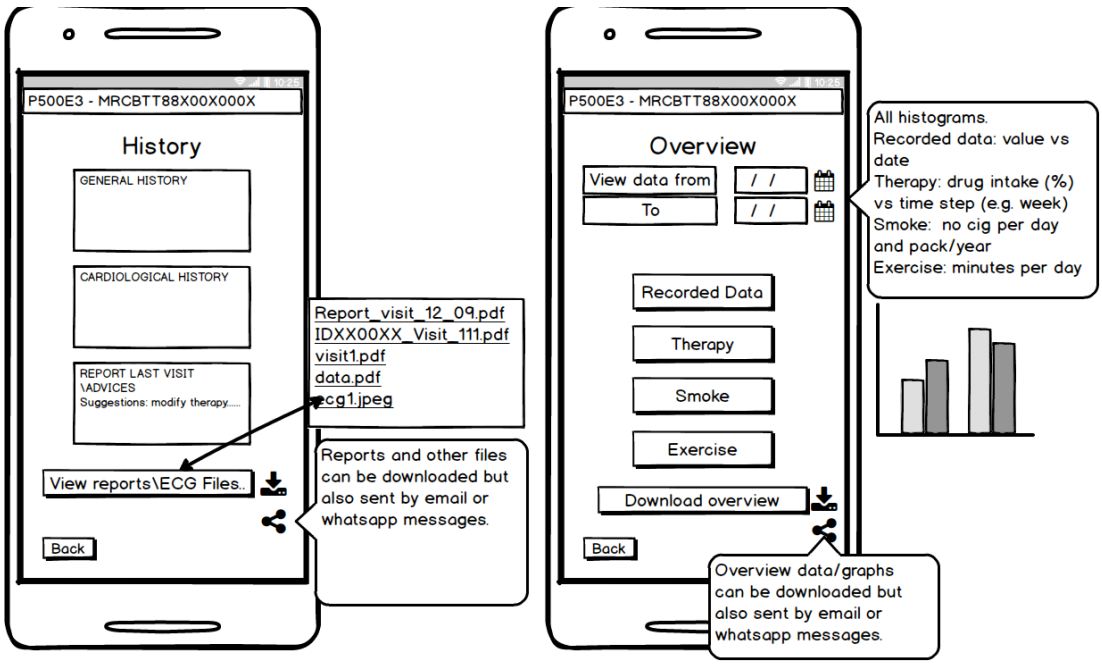

Figure 5.9: Patient's Data.

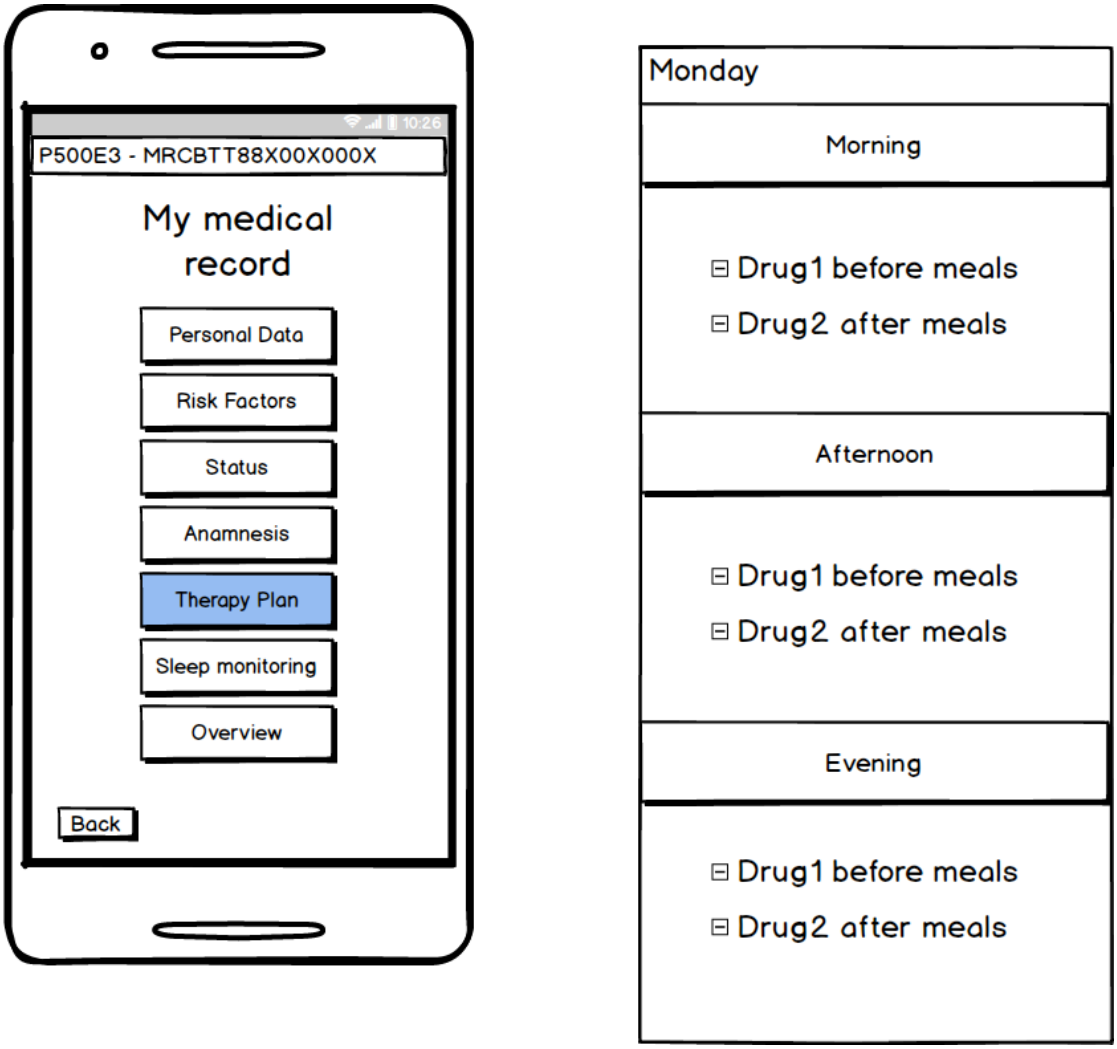

Figure 5.10: Patient's Data.

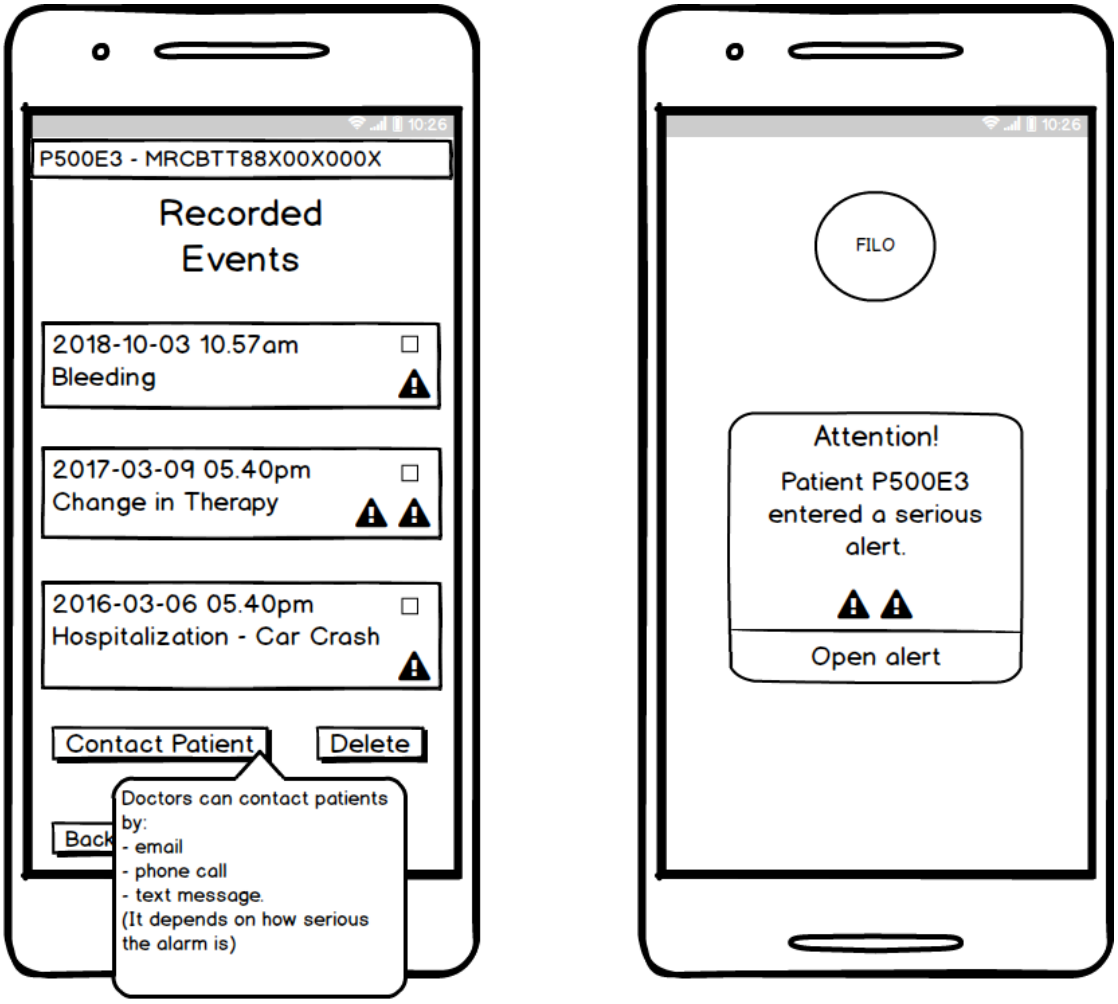

Figure 5.11: Notification's Activity.

Bibliography

[1] Sara Prete. «Ballistocardiographic heart rate and respiratoryrate detection». In: (Apr. 2019) (cit. on pp. iv, 1, 7, 11, 58).

[2] *Definition of Caregiver*. URL: https://en.wikipedia.org/wiki/Caregiver (cit. on p. 6).

[3] *Zeplin*. URL: https://zeplin.io/ (cit. on p. 11).

[4] *Why Android Developers Should Prefer Android Studio Over Eclipse?* 2016. URL: https://www.brainvire.com/android-developers-prefer-android-studio-eclipse (cit. on p. 16).

[5] Michael Williams. *Java-Kotlin comparison for android programming*. 2019. URL: https://www.promptbytes.com/blog/java-vs-kotlin-the-no-nonsense-comparison-of-android-programming-languages (cit. on p. 18).

[6] *Auth0*. URL: https://auth0.com/docs/getting-started/overview (cit. on p. 19).

[7] *Why Auth0?* URL: https://auth0.com/why-auth0 (cit. on p. 19).

[8] *Universal vs Embedded Login*. URL: https://auth0.com/docs/guides/login/universal-vs-embedded (cit. on pp. 20, 21).

[9] *Universal Login*. URL: https://auth0.com/docs/universal-login (cit. on p. 21).

[10] *OkHttp*. URL: https://square.github.io/okhttp/ (cit. on p. 23).

[11] *Why Git for your organization*. URL: https://www.atlassian.com/git/tutorials/why-git (cit. on p. 25).

[12] *gitlab*. URL: gitlab.com (cit. on p. 26).

[13] Liran Karni. «openEHR approach for the evauation and management of elderly patients with multi-morbidity». In: *Research Gate* (May 2016), pp. 1–17 (cit. on p. 27).

[14] *SQlite3*. URL: https://www.sqlite.org/index.html (cit. on p. 28).

[15] *Postgre SQL*. URL: https://www.postgresql.org/ (cit. on p. 28).

[16] *Django*. URL: https://www.djangoproject.com/ (cit. on p. 28).

[17] *Android Security and App security best practices*. URL: https://developer.android.com/topic/security/best-practices (cit. on pp. 30, 42).

[18] Jonathan Katz and Yehuda Lindell. «Introduction to Modern Cryptography». In: (2007) (cit. on p. 30).

[19] Muhammad Aamir Panhwar Sijjad Ali khuhro Ghazala Panhwar Kamran Ali memon. «SACA A Study of Symmetric and Asymmetric Cryptographic Algorithms». In: (Jan. 2019) (cit. on p. 30).

[20] Nicola Gallo Loris Gabriele Luca Ghio Lorenzo Liotino Muhammad Ali. «Computer system security». In: (Feb. 2015) (cit. on p. 31).

[21] Dawn M. Turner. «Applying Cryptographic Security Services». In: (Aug. 2017) (cit. on p. 32).

[22] Yogesh Kumar Rajiv Munjal Harsh Sharma. «Comparison of Symmetric and Asymmetric Cryptography with Existing Vulnerabilities and Countermeasures». In: (Oct. 2011) (cit. on p. 34).

[23] Ons JALLOULI. «Chaos-based security under real-time and energy constraints for the Internet of Things». PhD thesis. Oct. 2017 (cit. on p. 35).

[24] Nicola Gallo Loris Gabriele Luca Ghio Lorenzo Liotino Muhammad Ali. «A Study on Asymmetric Key Cryptography Algorithms». In: (Apr. 2015), p. 8 (cit. on p. 35).

[25] *Auth0 Access Tokens*. URL: https://auth0.com/docs/tokens/concepts/access-tokens (cit. on p. 38).

[26] *Auth0 Refresh token*. URL: https://auth0.com/docs/tokens/concepts/refresh-tokens (cit. on p. 38).

[27] Larry Morell. «Unit Testing and Analysis». In: (Apr. 1989), p. 41 (cit. on p. 40).

[28] Gary McGraw Viega Jhon. «Building Secure Software: How to Avoid Security Problems the Right Way». In: (2001), p. 5 (cit. on p. 41).

[29] Capers Jones. *Applied Software Measurement : Global Analysis of Productivity and Quality: Global Analysis of Productivity and Quality*. 2008 (cit. on pp. 41, 42).

[30] *Mobile Operating System Market Share*. URL: https://gs.statcounter.com/os-market-share/mobile/macedonia (cit. on p. 57).