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: SIGL

Dissertation submitted to obtain Master degree

By: Salah benchibout, ilyas
hamoudi.

SUBJECT

Mobile Application for Online Doctor's Appointment Booking

Supported before the jury composed of:

Mr.Moussaoui Adel	University of M'sila	President
Mr.Barkat Abdelbasset	University of M'sila	Supervisor
Mr.Ouldmohamedi Najib	University of M'sila	Examiner

Academic year: 2022/2023

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: SIGL

Dissertation submitted to obtain Master degree

By: Salah benchibout, ilyas
hamoudi.

SUBJECT

Mobile Application for Online Doctor's Appointment Booking

Supported before the jury composed of:

Mr.Moussaoui Adel	University of M'sila	President
Mr.Barkat Abdelbasset	University of M'sila	Supervisor
Mr.Ouldmohamedi Najib	University of M'sila	Examiner

Academic year: 2022/2023

Dedications

First of all, thank god for everything "الحمد لله "

To my great father, "رحمه الله".

To my dear mother for her great support,

I wish her good health and

Long happy life.

Benchibout salah

To my only sister, "رحمها الله".

To my sweet parents,

To all my relatives and classmates,

And of course, to all my friends.

Thank you all.

Hamoudi ilyas

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.

thanks to my family who were with me throughout my studies.

My greetings and thanks to the supervising professor: Barkat Abdelbasset, for his continuous support and guidance.

Table of Contents

Figures Table

Tables List

Abbreviations List

GPS: Global **P**ositioning **S**ystem.

CSS: Cascading **S**tyle **S**heets.

HTML: Hyper **T**ext **M**arkup **L**anguage.

IOS: iPhone Operating System.

SDK: Software **D**evelopment **K**it.

UI: User **I**nterface.

USB-C: Universal **S**erial **B**us type-**C**.

SSL: Secure **S**ockets **L**ayer.

DVM: Dalvik **V**irtual **M**achine.

IDE: Integrated **D**evelopment **E**nvironment.

UML: Unified **M**odelling **L**anguage.

API: Application **Programming I**nterface.

RAM: Random **A**ccess **M**emory.

XML: e**X**tensible **M**arkup **L**anguage.

JDK: Java **D**evelopment **K**it.

GENERAL INTRODUCTION

Today, our lives have become easier with the rapid development of technology, which is now integrated into all our daily activities. One of the most popular technological advancements is the smartphone, which offers numerous benefits and plays a significant role in our lives. While phones were originally used for making calls and sending messages, they now provide a wide range of features and services. These include playing games, watching movies or news, and accessing various smart applications. Moreover, smartphones have internet connectivity, enabling the creation of social networking applications that effectively shrink the world into a small village. These applications have found utility in multiple domains, such as the medical field and booking systems. For instance, they help people save time by facilitating appointment bookings from the comfort of their homes or workplaces. They also assist doctors in managing their patients by organizing reservations and maintaining each patient's order.

In light of these advancements, we have embarked on designing and developing a mobile application to simplify the process of scheduling appointments with doctors from anywhere. Our application will also aid doctors in managing their patient queues. Furthermore, it will assist with unfavorable weather conditions and help users navigate through traffic congestion. Importantly, it aims to minimize the risk of cross-infections in waiting rooms, particularly in cases of severe diseases like COVID-19.

This dissertation is divided into three chapters outlining the steps in developing an appointment assistance application. The first chapter provides an overview of mobile applications in general and highlights the most important platforms. In the second chapter, we present the design of our application, including the Unified Modeling Language (UML) and the identification of key actors who will play a role in the implementation of our management tool. We define the functions and use cases of these actors through the use of case and sequence diagrams. The third chapter focuses on the development environment of the mobile application, accompanied by a few screenshots that offer a brief explanation of how to use the application's features. Finally, we conclude with a summary of the main points of our work and evaluate the extent to which we have achieved our goals.

CHAPTER 1

GENERAL INFORMATION ABOUT MOBILE APPLICATIONS

1.1. Introduction

A lot of people today have a smartphone, and they take advantage of their phones by using them in daily activities and tasks like taxi requests, reading news, social media networking, or GPS usage.

In this chapter, we will talk about mobile applications, their types, and the most important platforms; we will take a look at the problem of booking appointments and its solution using the mobile application.

1.2. Mobile applications

A mobile application is a software application developed specifically for use on small, wireless computing devices, such as smartphones and tablets, rather than desktop or laptop computers [1].

Mobile applications are used in different domains like games, messaging, communication, and social networking, etc. we can download any application and install it on the smartphone or we can find it installed already on the phone, there are a lot of applications running on Android, iPhone and Windows and some virtual machines, etc.

1.2.1. Types of mobile applications

Mobile applications are divided into three types:

1. **Native apps**

One of the most popular mobile apps is NATIVE, which is a software program built by developers for use on a particular platform or device [2], that means it is developed for a single operating system and this app will work only on this platform or operating system.

2. **Web apps**

A web app is an app program that is stored on a remote server and delivered over the internet through a browser interface [3],

3. **Hybrid apps**

A Hybrid app is software that combines elements of both Native apps and Web apps [4], those apps are multi-platform technologies developed using CSS, JavaScript, and HTML, Hybrid apps simply are web apps concealed in a native package, Instagram, Netflix, and Amazon are most popular examples of Hybrid apps

1.2.2. Categories of mobile applications

7 main categories of apps have been able to make it to the market [5]:

- **Gaming:** there are a lot of famous games that exploded in the market like pubg, candy crush …, etc.
- **Business:** these apps are called productivity apps and they are the second most demanded app among users, like buying and selling apps.
- **Educational:** a lot of apps are designed for students and even teachers. Such apps are designed to help kids enjoy learning new concepts and technologies.
- **Lifestyle:** from fitness, shopping, and workout, to weight loss and more, these apps are related to offering users ideal solutions related to their working tasks, fun, or other lifestyle problems.
- **Entertainment:** these apps offer an overall experience that is too refreshing for the users, like watching videos online and chatting.
- **Utility:** users love these apps as it helps them to get things done early and easily, such as booking a cab, undertaking healthcare, and hiring a home repair service.
- **Travel:** designed to make travel more comfortable, easier, informative, and fun-filled for the users.

1.3. Mobile operating systems

Like computers, a mobile phone is powered by an operating system to run mobile apps, in the next paragraph; we are going to mention the most known operating systems for mobile:

- **Android:** Android was built from the ground-up to enable developers to create compelling mobile applications that take full advantage of all a handset has to offer. It was built to be truly open. Android is built on the open Linux Kernel. Furthermore, it utilizes a custom virtual machine that was designed to optimize memory and hardware

resources in a mobile environment. Android is open source; it can be liberally extended to incorporate new cutting-edge technologies as they emerge [6].

- **Apple IOS:** iOS is a proprietary operating system that runs on mobile devices such as the iPhone and iPad and other Apple devices. The first version of iOS was released in June 2007 [7].
- **Blackberry OS:** Blackberry OS is a proprietary mobile operating system designed specifically for RIM (research in motion) Blackberry devices, it runs on Blackberry variant phones like the Blackberry Bold, Curve, Pearl, and Storm series [8].
- **Windows OS:** Windows mobile was a Microsoft operating system that targeted smartphones and pocket PCs. It included basic apps developed with the Microsoft Windows API and options for customization and software development with no restrictions by Microsoft [9].

Since we chose to develop an Android application, we will give in the next pages a short overview of the Android operating system.

1.4. Android operating system

1.4.1. Android definition

Android is an open-source operating system that offers a unified approach to application development for mobile devices, which means that the developers need to write the code and compile their application only one time for Android, and their application should be able to run on different devices powered by Android. The first beta version of the Android Software Development Kit (SDK) was released by Google in 2007 whereas the first commercial version, Android 1.0, was released in September 2008 [10].

1.4.2. Android versions

Google makes incremental changes to the OS with each release. This often includes security patches and performance improvements [6], these versions are introduced in the table below:

Android version	Name	Release date	characteristics
Android 1.0	/	23/09/2008	Include a suite of Google apps like Gmail and YouTube.
Android 1,5	Cupcake	27/04/2009	Introduced an onscreen virtual keyboard and the framework for third-party app widgets.
Android 1.6	Donut	15/09/2009	Introduced the ability for the OS to run on different screen sizes and resolutions.
Android 2.0	Eclair	26/10/2009	Added turn-by-turn voice navigation, time traffic information, and pinch-to-zoom capability.
Android 2.2	Froyo	20/05/2010	Added dock at the bottom of the home screen which allows users to tap an icon and speak a command. Also introduced support for Flash to the web browser.
Android 2.3	Gingerbread	06/12/2010	Introduced black and green into the user interface (UI).
Android 3.0 to 3.2	Honeycomb	22/02/2011	This release was exclusive to tablets and introduced a blue space-themed holographic design.
Android 4.0	Ice Cream	18/10/2011	Introduced a unified UI to both tablets and smartphones, emphasizing swiping as a navigational method.
Android 4.1 to 4.3	Jelly Bean	09/07/2012 13/11/2012 24/07/2013	Introduced Google Now, a day planner service. Added interactive notifications and improved voice search system.
Android 4.4	KitKat	31/10/2013	Introduced lighter colors into the UI, along with a transparent status bar and white icons.
Android 5.0	Lollipop	12/11/2014	Introduced hands-free voice control with the spoken “ok, Google” command.
Android 6.0	Marshmallow	05/10/2015	This release marked Google’s adoption of an annual release schedule. Introduced more granular app permissions and support for USB-C and fingerprint readers.
Android 7.0 and 7.1	Nougat	22/08/2016 04/10/2016	Introduced a native split-screen mode and the ability to bundle notifications by app.

Android 8.0 and 8.1	Oreo	21/08/2017 05/12/2017	Introduced a native picture-in-picture (PIP) mode and the ability to snooze notifications.
Android 9.0	Pie	06/08/2018	This version replaced the Back, Home, and Overview buttons with a multifunctional Home button and a smaller Back button. Introduced productivity features, including suggested replies for messages and brightness management capabilities.
Android 10	Android Q	03/09/2019	Abandoned the Back button in favor of a swipe-based approach to navigation. Introduced a dark theme and focus mode, which enables users to limit distractions from certain apps.
Android 11	Red Velvet	08/09/2020	Added built-in-screen recording. Created a single location to view and respond to conversations across multiple messaging apps. This version also updated the chat bubbles so users can pin conversations to the top of apps and screens.
Android 12	Snow Cone	04/10/2021	Added customization options for the user interface. The conversation widget let users store preferred contacts on their home screens. Added more privacy options, including sharing when apps access information such as camera, photos, and microphone.
Android 12L		07/03/2022	The L stands for a larger screen. This update aimed to improve the UI and optimize for the larger screen of a tablet, foldable, or Chromebook. This version added a dual-panel notification center for tablets and foldable.
Android 13		15/08/2022	Included more customizable options such as colors and music. Security updates included control over information apps can access, notification permission required for all apps, and clearing of personal information on the clipboard. This update enables multitasking by sharing messages, chats, links,d photos across multiple Android devices – including phones, tablets, and Chromebooks.

Table 1. Android versions

1.4.3. Android System Architecture

Android OS is a stack of software components, which is roughly divided into five sections and four main layers as shown below in Figure 1 [11].

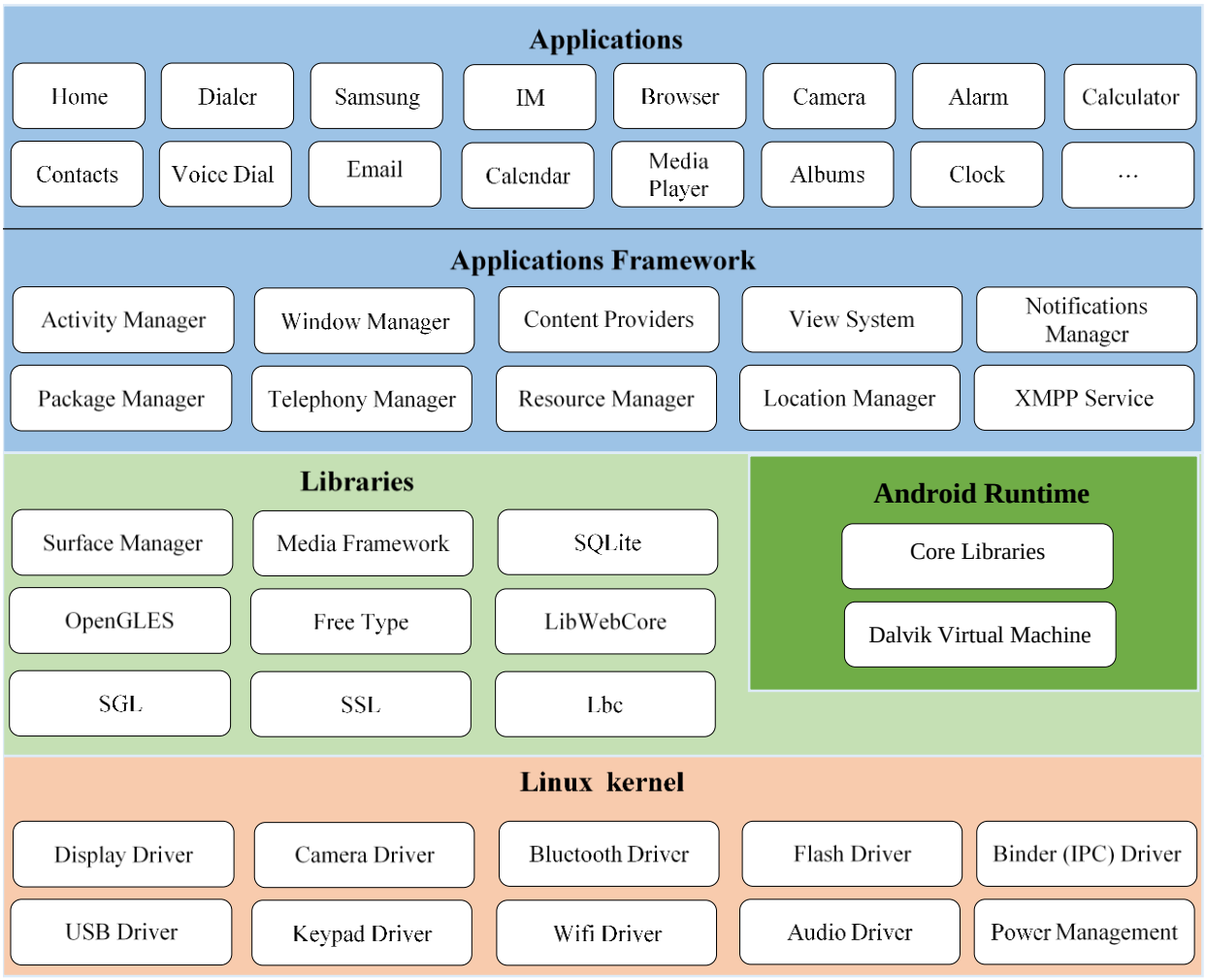


Figure 1.Android architecture and its components.

And these are the main components of this system:

1. **Linux kernel:** At the bottom of the layers is Linux 3.6 with approximately 115 patches. This provides a level of abstraction between the device hardware and it contains all the essential hardware drivers like the camera, keypad, and display. Also, the kernel handles all the things that Linux is good at such as networking and a vast array of device drivers, which take the pain out of interfacing to peripheral hardware [11].
2. **Libraries:** On top of the Linux kernel there is a set of libraries including the open-source web browser engine WebKit, the well-known library libc, SQLite database which is a useful repository for storage and sharing of application data, libraries to play and record audio and video, SSL libraries are responsible for internet security, etc. [11].

3. **Android libraries:** This category encompasses those Java-based libraries that are specific to Android development. Examples of libraries in this category include the app framework libraries in addition to those that facilitate user interface building, graphics drawing, and database access. A summary of some key core Android libraries available to the Android developer is as follows [11]:

- **Android. app:** provides access to the app model and is the cornerstone of all Android apps.
- **Android. content:** Facilitates content access, publishing, and messaging between the app and its components.
- **Android.database:** Used to access data published by content providers and includes SQLite database management classes.
- **Android.OpenGL:** A Java interface to the OpenGL ES 3D graphics rendering API.
- **Android.os:** Provides apps with access to standard OS services including messages, system services, and inter-process communication.
- **Android.text:** Used to render and manipulate text on a device display.
- **Android.view:** The fundamental building blocks of app user interfaces.
- **Android.widget:** A rich collection of pre-built user interfaces components such as buttons, labels, and list views, etc.
- **Android.webkit:** A set of classes intended to allow web-browsing capabilities to be built into apps.

Having covered the Java-based core libraries in the Android runtime, it is now time to turn our attention to the C/C++-based libraries contained in this layer of the Android software stack.

4. Android Runtime: This is the third section of the architecture and is available on the second layer from the bottom. This section provides a key component called Dalvik Virtual Machine which is a kind of Java virtual machine specially designed and optimized for Android. The DVM makes use of Linux core features like memory management and multithreading, which are intrinsic in the Java language. It also enables every Android app to run its process, with its instance of the DVM.

The Android runtime also provides a set of core libraries that enable Android app developers to write Android apps using standard Java programming language.

4. **Application Framework:** The app Framework layer provides many higher-level services to apps in the form of Java classes. Application developers are allowed to make use of these services in their apps.

The Android framework includes the following key services:

- Activity Manager: Controls all aspects of the application lifecycle and activity stack.
- Content Providers: Allows apps to publish and share data with other apps.
- Resource Manager: Provides access to non-code embedded resources such as strings, color settings, and user interface layouts.
- Notifications Manager: Allows apps to display alerts and notifications to the user.
- View System: An extensible set of views used to create application user interfaces.

6. Applications: You will find all the Android applications on the top layer. You will write your application to be installed on this layer only. Examples of such applications are contacts, browsers, games, etc.

1.5. Subject presentation

1.5.1. The problem of booking appointments

A lot of patients have a problem with making a doctor's appointment, like living far from the clinic, hard weather changes, and transport problems. They have to waste hours and hours waiting for their turns, and the biggest problem here is how to avoid gathering in the waiting room which rise the percentage of being affected by an infectious disease like Covid-19.

1.5.2. The solution with a mobile application

We are in a revolution of technology and raging competition between mobile companies, mobile phones became cheaper especially those using the Android operating system and we can find at least one phone in each house. Because of that, we think to develop a mobile application that works on Android OS to facilitate having appointments for those patients, this app allows them to have an appointment from anywhere and gives them a chance to go into the clinic anytime they want, they can consult the booking list and know their order, as a result, they gain a lot of time for another activity. They can also consult the doctor's announcements for example if he changes his address or he is on a holiday.

1.6. Conclusion

All domains are going to be digitized and booking appointments must have its share of digitization. Therefore, we chose to develop a mobile application for its benefits and use smartphones to help people and make their lives easier. In this chapter, we have seen smart apps in general and their types, we also mentioned the most famous operating systems, and finally, we presented the problem of having a doctor's appointment and the solution using a mobile application on Android. In the next chapter, we will take a look at the analysis and design of our doctor's appointment booking application.

CHAPTER 2
ANALYSIS AND DESIGN

2.1. Introduction

The first step of designing an application is to analyze the situation to take into consideration the constraints, risks, and everything related to the development of any application. Before starting the development, we will identify all features of our application, specify our objectives and express all needs of the actors. This chapter contains the architecture of our application, the identification of actors and their needs, and course of the modeling language (UML). All needs will be modeled by a use case diagram for each actor, followed by their sequence diagrams and finally the class diagram.

2.2. General Architecture of our application

Our application is divided into two applications, the first is for the doctor or his receptionist and the second one is for patients. The next figure identifies the synchronization of data momentarily (real-time update of data) if any user changed something.

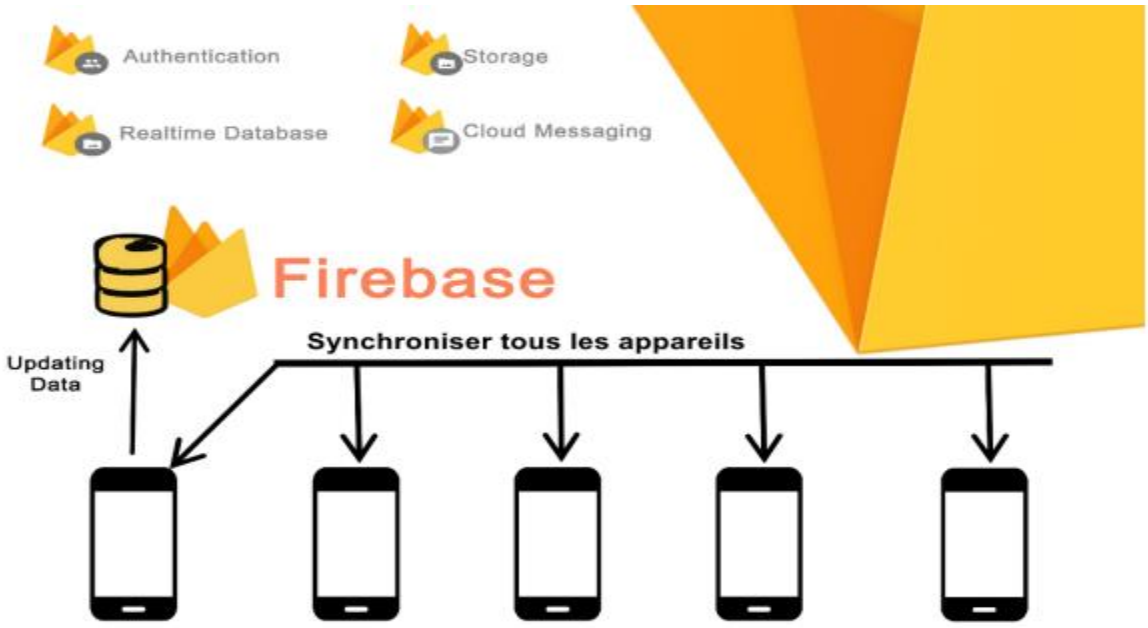


Figure 2. The general architecture of our application

2.3. UML

The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system. The UML gives you a standard way to write a system's blueprints, covering conceptual things, such as business processes and system functions, as well as concrete things, such as classes written in a specific programming language, database schemas, and reusable software components [19]. In our case we use three types of UML diagrams: use case diagram, sequence diagram, and class diagram.

2.3.1. Use Case Diagram

The use case diagrams describe the functionality of a system in terms of actors, goals as use cases, and dependencies among them. In another word, they represent the list of actions or event steps typically defining the interactions between the actor and the system (the application) to achieve a goal [12].

In our application, we have two types of actors, *the patient* (user) who is a member of our app and he can use it within his account's limits. The other user is the *admin* who is the doctor or his receptionist, where they have all permissions and controls.

2.3.1.1. Use case diagram of the user: as shown in Figure 3, the user has many functionalities that we can summarize as:

- Create a user account.
- Log in and log out.
- Make an appointment and cancel it.
- Consult the order of patients in the list.
- Consult and interact with the doctor's announcements.
- Contact and text the doctor.
- Manage his account and receive notifications.

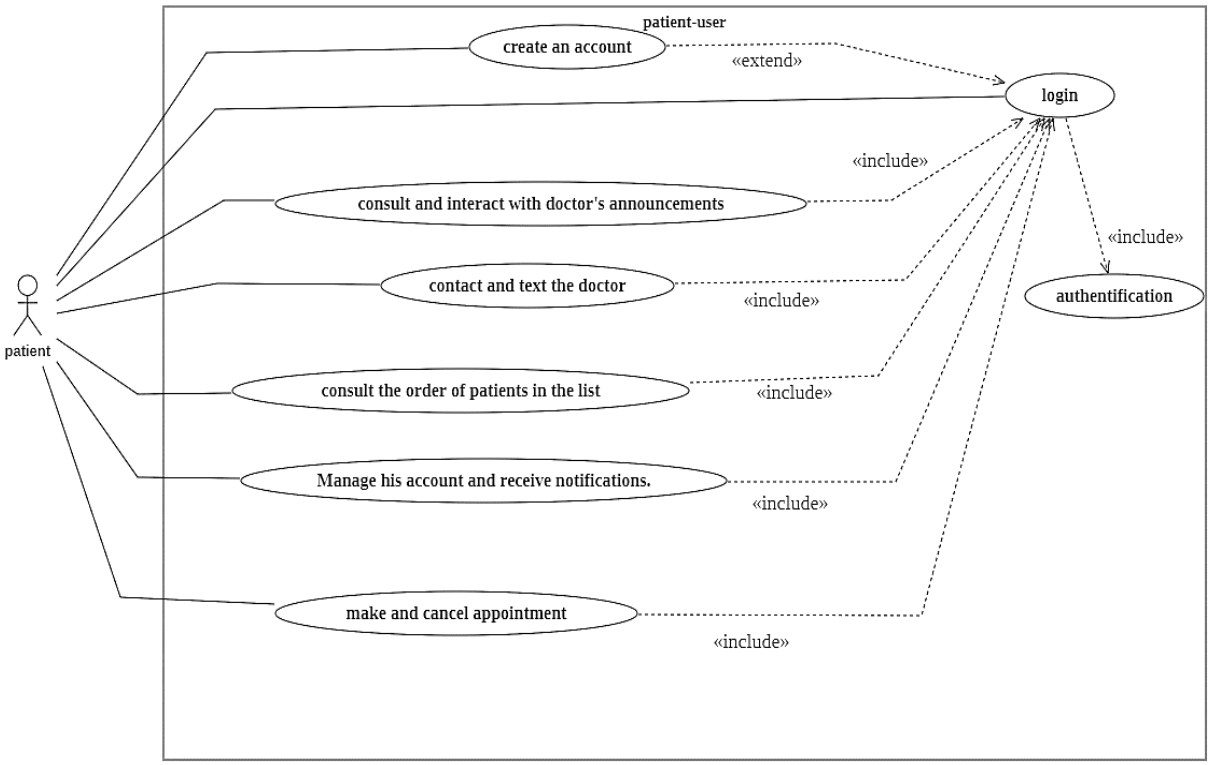


Figure 3. User use case diagram.

2.3.1.2. Use case diagram of the admin: as shown in Figure 4, the admin has many functionalities that we can summarize as:

- Create an admin account.
- Log in and Log out.
- Create and consult booking list.
- Make an appointment manually.
- Cancel and delete appointments.
- Post and delete announcements.
- Manage his account and receive notifications.
- Receive and respond to patient messages

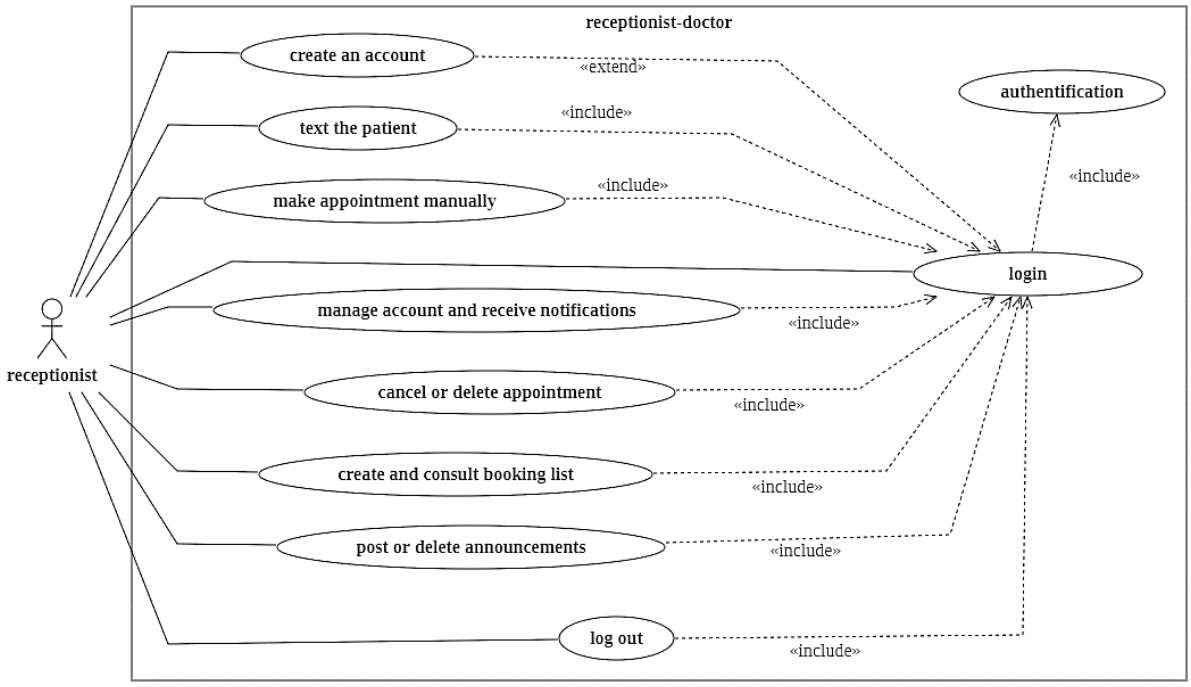


Figure 4. Admin use case diagram.

In the next paragraphs, we will provide a textual explanation to specify the requirements of some use cases that we reckon are very important in our work:

i. Authentication's use case

The Authentication provides access to reserved features for each type of user, includes log in with email and password. It is required from both admin and user, authentication needs an accessible application and connection to the internet as pre-conditions, if the authentication process failed, an error message will be sent to them.

ii. Create account use case

Any user needs to create an account to have advantage of the application's features. The creation of a new account needs access to the application and a connection to the internet; both the admin and user do it by filling all fields with valid information.

iii. Make appointment's use case

The main purpose of our application is to make appointments; the two actors need an existing booking list to register on it before the ending time. They can register in a booking list

just once, making an appointment ends with failure in case of a blank name or a bad connection to the internet.

iv. Delete/cancel appointment's use case

On the other hand, the actors have the freedom to cancel their appointments sometimes; we make them able to cancel them at any time for any reason. They must make an appointment to cancel it, this process needs the same conditions as making an appointment: connection to the internet, success authentication, accessible application, and of course the existence of the member on the booking list.

v. Post announcement's use case

This is only for the admin, who he can post announcements to inform his patient about any news, notes, or advice. To post an announcement the admin needs to insert an image with a textual description, the previous conditions are required also. The admin can delete an existing announcement if he wants to.

2.3.3. Sequence Diagram

A sequence diagram represents communication between objects in terms of a sequence of messages [12]. The next part contains the main sequence diagrams of our system according to the use cases, the authentication and inscription sequence diagrams are not mentioned because they are probably the same in all applications.

I. Make appointment Sequence Diagram

By Admin: the admin needs to login into the admin's application successfully, and then click on the 'add patient' button. A query starts here to check if an existing list is started: in case of started list, the admin must enter a patient name (non-blank) then the application sends the patient name to the database where it checks the existence of that member in the booking list. if yes, the application asks the admin to change the patient name, else, the database updates the booklist by adding the patient name to the list. As a result, the appointment is registered successfully. On the other hand, the application informs the admin that there is no started list, the scenario detailed in Figure [5] below.

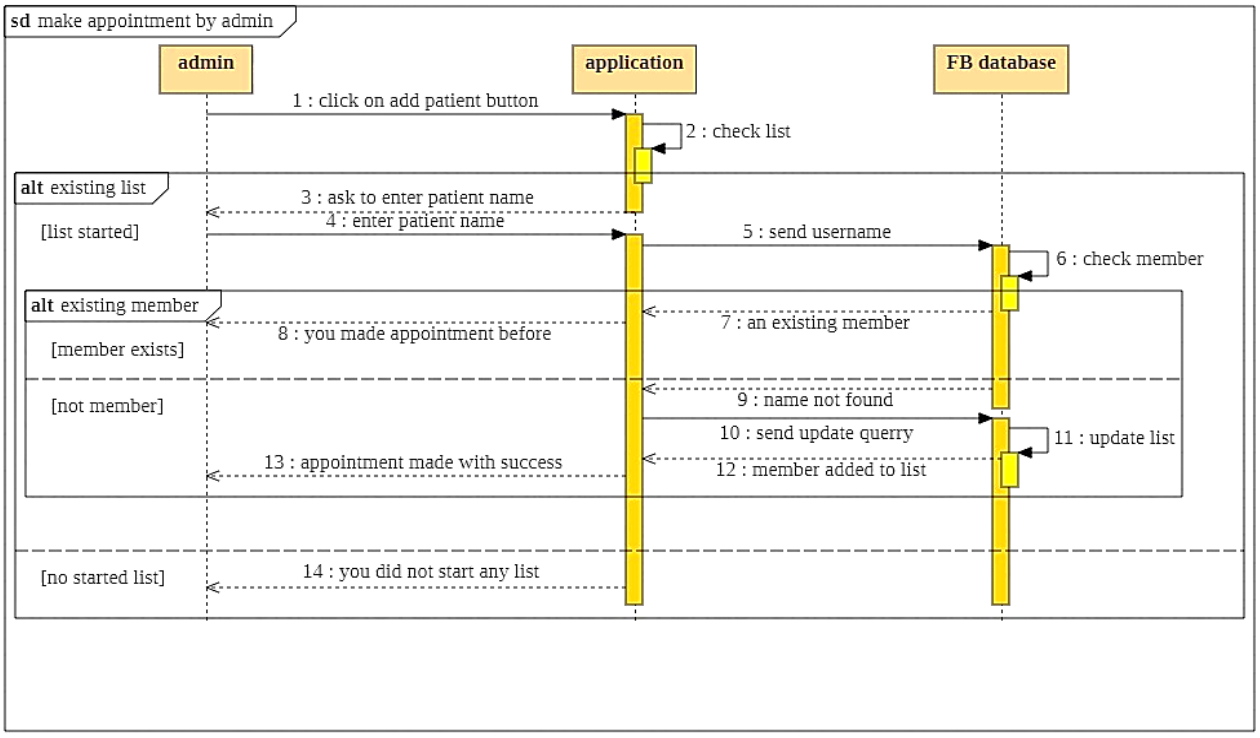


Figure 5.Sequence diagram of making appointments by admin.

By user/patient: the user needs to login into the user's application successfully, then click on the 'make appointment' button. The process begins with a query to check if an existing list has been started. If the list has been started, the application sends the username to the database to verify if the member already exists in the booking list. If the member is found in the booking list, the application informs the user that they have made an appointment before. However, if the member is not found, the database updates the booklist by adding the patient's name. This results in the successful registration of the appointment. As a result, the appointment is registered successfully. On the other hand, the application informs the user that there is no started list, the scenario detailed in Figure [6].

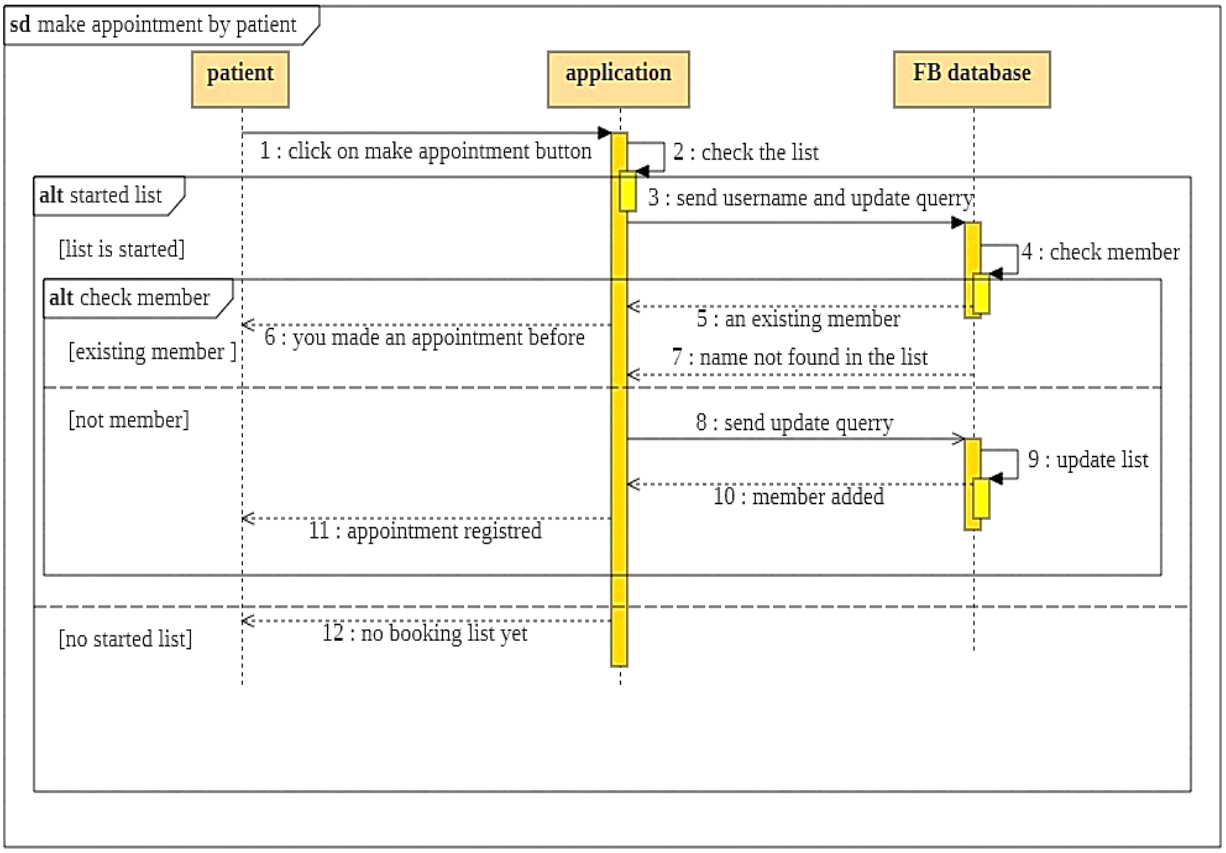


Figure 6.Sequence diagram of making appointments by the user.

II. Cancel appointment Sequence diagram

By admin: this diagram describes the operation of appointment cancelation from the admin's application. Where the admin needs a successful authentication before choosing a name from the booking list to cancel his reservation, the application here asks for confirmation. The admin has two choices either to cancel his order and terminate the operation or to confirm it, then the application sends a deletion query to the database where it updates the booking list by removing the patient from it. This diagram is shown in Figure [7].

By user: this diagram describes the operation of appointment cancelation from the user's application. Where the user needs a successful authentication before entering the booking list and choosing the cancelation option, the application here asks the database if that user has made an appointment before, then it asks for confirmation to cancel the appointment or cancel the removing operation if the user was a member in the list if the user was not a member in the booking list the application sends an error message, this diagram is detailed in figure [8].

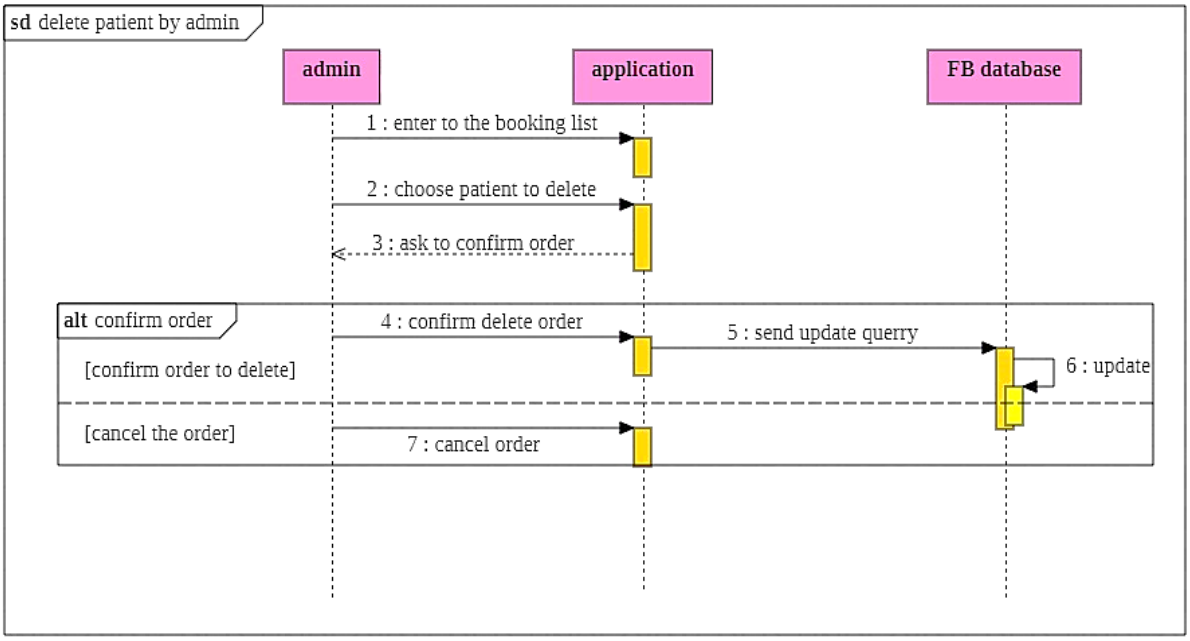


Figure 7. Sequence diagram of appointment's cancelation/deletion by admin.

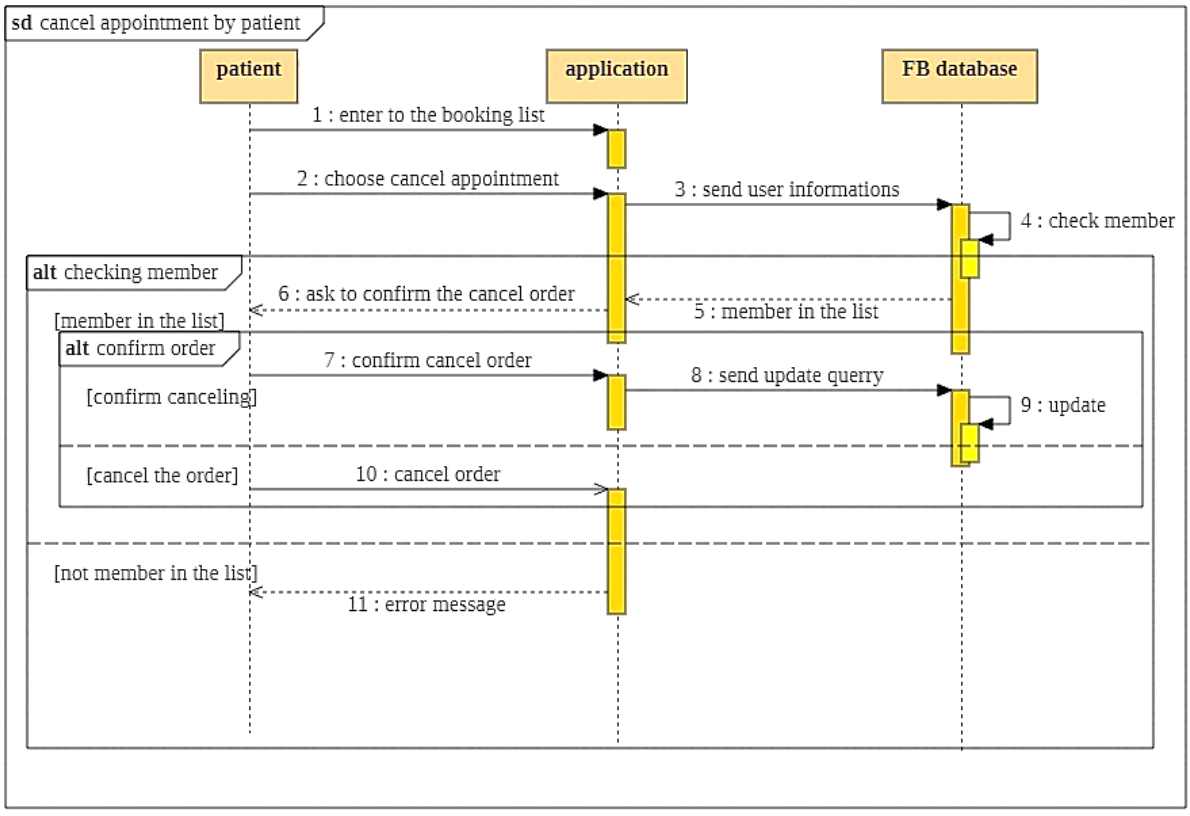


Figure 8. Sequence diagram of appointment's cancelation by the user.

III. Post announcement Sequence diagram

This action is only performed by the admin. First of all, he needs to log in successfully, then he displays the announcement page by clicking on its button. Second, after the application displays the announcement's page, the admin must enter an image with a textual description in the shown dialog box after clicking on add announcement button, here the application makes a check for an empty field or no image entry. The announcement will be posted only when the user enters a text with an image or the operation will fail. This diagram is shown in Figure [9].

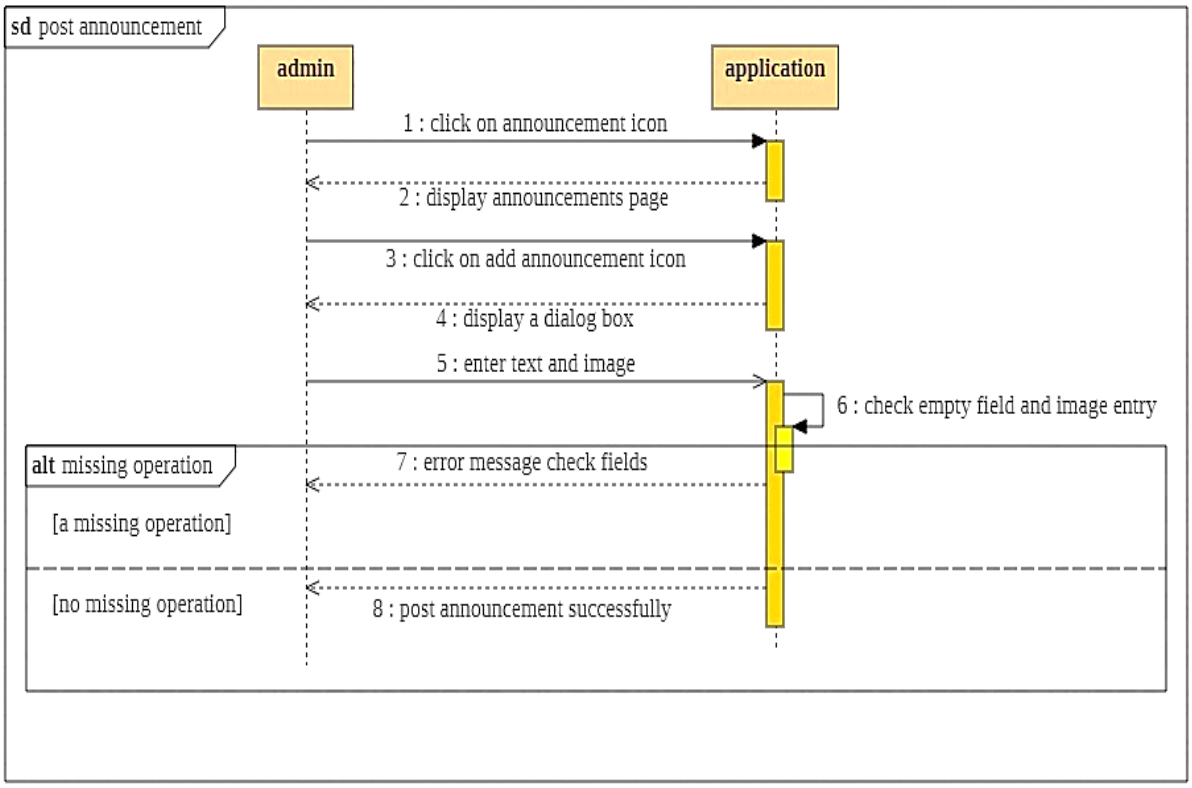


Figure 9. Post the announcement's sequence diagram.

2.3.4. Class diagram

The class diagram represents system class, attributes, and relationships among the classes [12]. This diagram is the UML version for databases. Our class diagram is represented in the next figure.

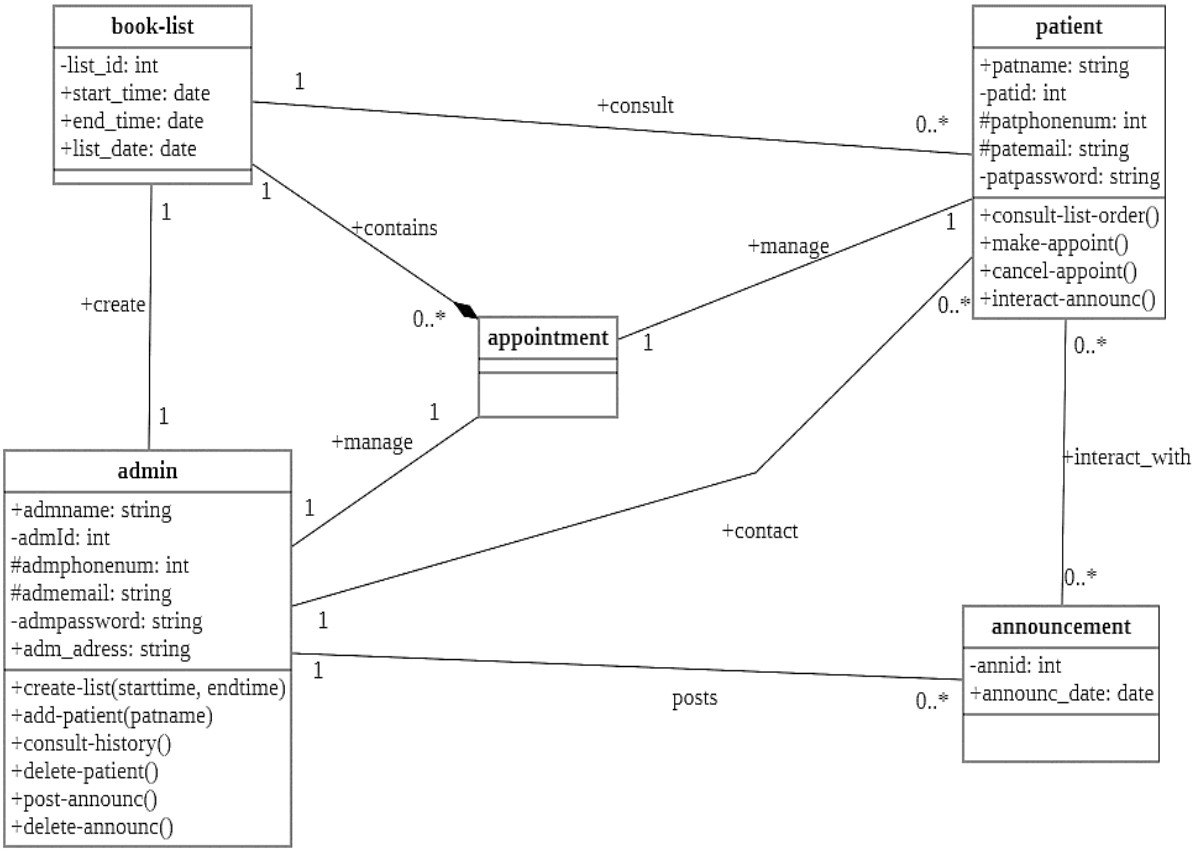


Figure 10.The class diagram of our application.

2.4. Conclusion

In this chapter, we had a presentation on UML diagrams used to build our application. We made a textual description of all interactions between the user and the system followed by the use case diagrams at the beginning of the chapter, then we completed the description with the sequence diagram of each use case, and we ended the chapter with the class diagram which allowed us to start the development of our applications.

CHAPTER 3
IMPLEMENTATION

3.1. Introduction

In this chapter, we will describe the implementation tools of our applications, starting from the development environment to the used libraries and APIs (Application Programming Interface), and finally, we will speak a little bit about the database and the security aspects then we conclude the dissertation with a presentation of some interfaces of our applications.

3.2. Development environment

The development environment could be divided into two parts, hardware, and software:

3.2.1. Hardware environment

We used a Toshiba computer for the implementation of our project, which is characterized by:

- Operating System: Windows 10, 64 bits.
- Processor: Intel i5 2nd generation.
- RAM: 4 GB.
- Hard disk: 120 GB SSD.

In addition, for the installation and running of the applications we used a Redmi smartphone with the following characteristics:

- Device name: Redmi Note 11 e.
- Android version: 12.
- Connection: LTE, 3G, 2G,5G.
- RAM: 4GO.
- Internal memory: 128 GO.

3.2.2. Software environment

This section contains the most important libraries used in our project.

Chapter 3: Implementation.

I. Android studio

Android Studio is an **integrated development environment** (**IDE**) that will take care of all the complexities of compiling our code and linking with the Android API. Once we have installed Android Studio, we can do everything we need inside this single application [13]. It is Based on the powerful code editor and developer tools from IntelliJ idea, and released by Google on 01/12/2014. We used the version 'Dolphin/2021.3.1 Patch 1' of Android Studio, and Java as a development language.

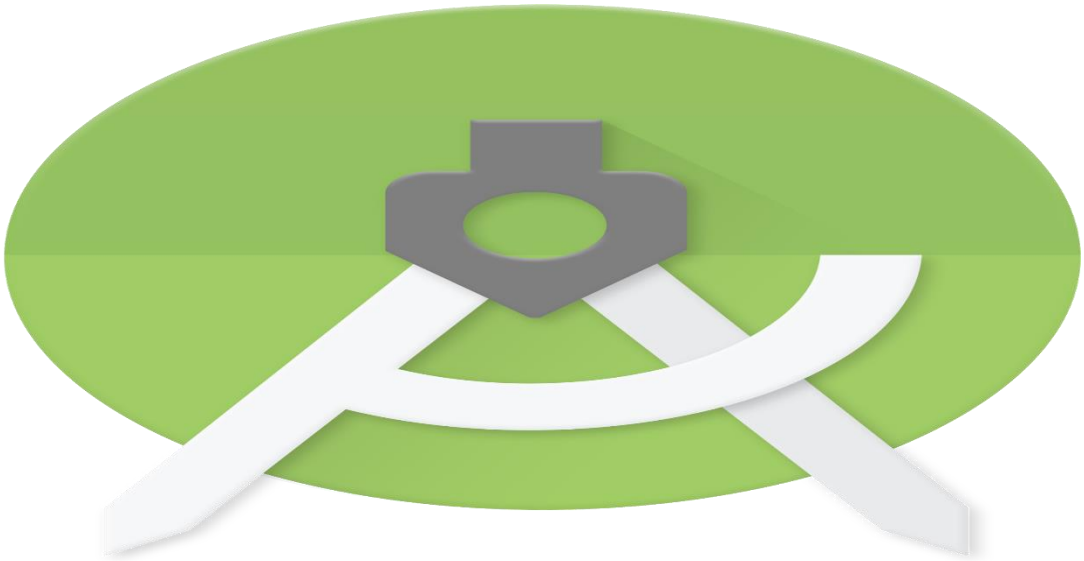


Figure 11 Android studio official logo.

II. Java development kit

Java development kit (JDK) is a development environment for building applications and components using the Java programming language, which includes tools useful for developing, testing, and monitoring programs written in Java and running on the Java platform [14].

Figure 12.Java Logo.

III. The eXtensible Markup Language

The eXtensible Markup Language (XML) is a simple text-based format for representing structured information: documents, data, configuration, books, and much more. It was derived from an older standard format called SGML (ISO 8879), to be more suitable for web use. XML is one of the most widely-used formats for sharing structured information today: between programs, between people, and between computers and people, both locally and across networks [15].

IV. Java

Java is a programming language and computing platform first released by Sun Microsystems in 1995. It has evolved from a humble beginning to power a large share of today's digital world, by providing a reliable platform upon which many services and applications are built. New, innovative products and digital services designed for the future continue to rely on Java, as well [16].

3.3. Used Libraries and APIs

Here are some libraries and APIs used in our project:

3.3.1. Real-time API

We used the real-time API to have the real-time get the real-time in the area.

```
public class timeclass extends AsyncTask<Void,Void,Void> {
    Handler h=new Handler();
    String result;
    String date,time,second;
    mm my;
    int boo;
    timeclass(mm my ,int boo){
        this.my=my;
        this.boo=boo;
    }
    @Override
    protected Void doInBackground(Void... voids) {
        String strurl = "https://www.timeapi.io/api/TimeZone/zone?timeZone=Africa/Algiers";
        InputStream inputStream = null;
        try {
            URL url = new URL(strurl);
            URLConnection connection = url.openConnection();
          //  connection.setConnectTimeout(10000);
            inputStream = new BufferedInputStream(connection.getInputStream());

            BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream, charsetName: "iso-8859-1"), sz: 8);
            StringBuilder sb = new StringBuilder();
            String line= "";
            while ((line = reader.readLine())!=null) sb.append(line+"\n");
            inputStream.close();
            result = sb.toString();
        } catch (Exception e){
```

Figure 13.Time API.

3.3.2. Circle image view

A Circle image view is a UI component used in Android application development that displays an image in a circular shape instead of the default rectangular shape. It is a subclass of the Image View class and provides a way to display profile pictures, icons, and other images in a circular shape.

3.3.3. Recycler view

Recycler view is a UI component in Android application development that is used to display large sets of data in a list of a grid format. It is an improvement over the older List View and Grid View components, providing better performance and more customization options.

3.4. Database Implementation

We used Google Firebase to build our project.

3.4.1. Google Firebase

Google Firebase is a set of cloud-based development tools that helps mobile application developers build, deploy and scale their applications [17]. To add Firebase to our project in Android Studio, we must do the following steps:

- Create a Gmail account in Google.
- Register the application in the Firebase platform by filling in the required fields.
- Download the file that contains Google services, and place it in the project package.

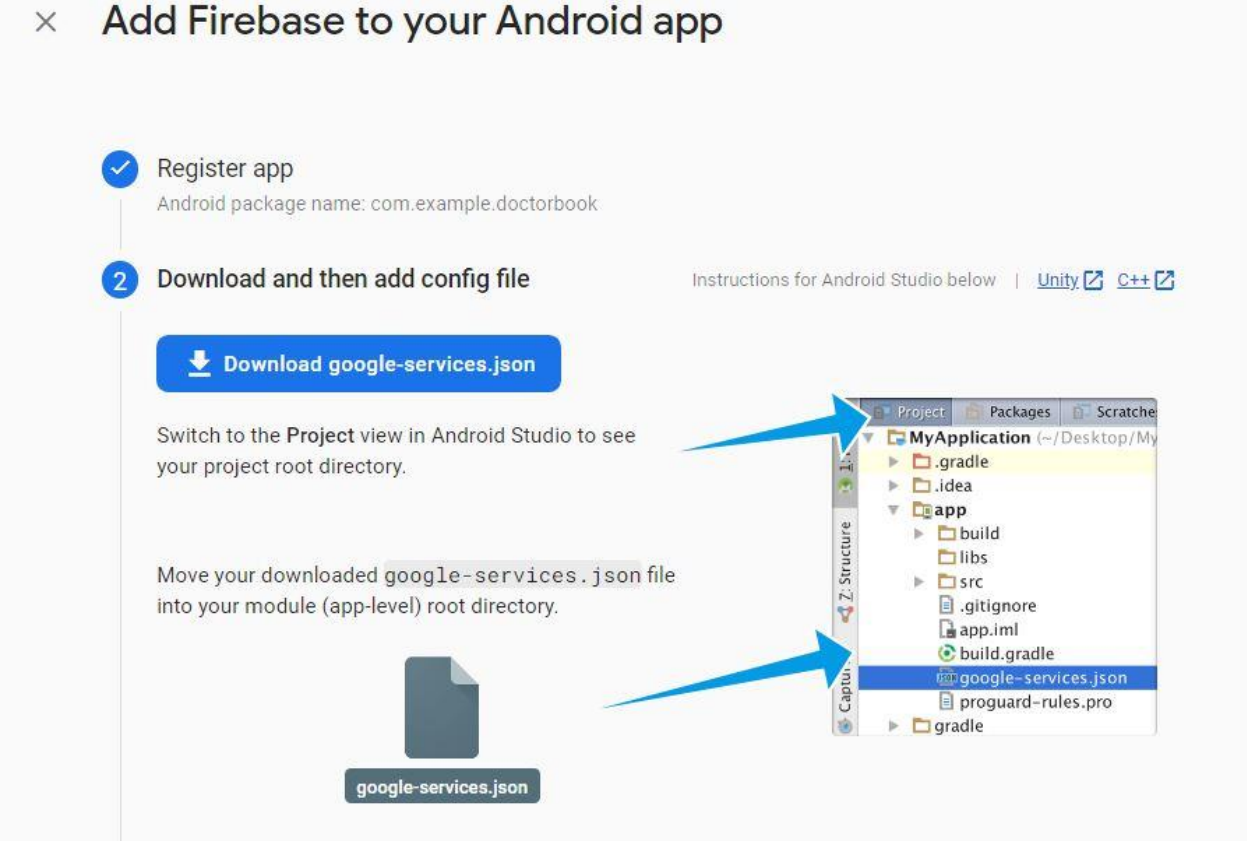
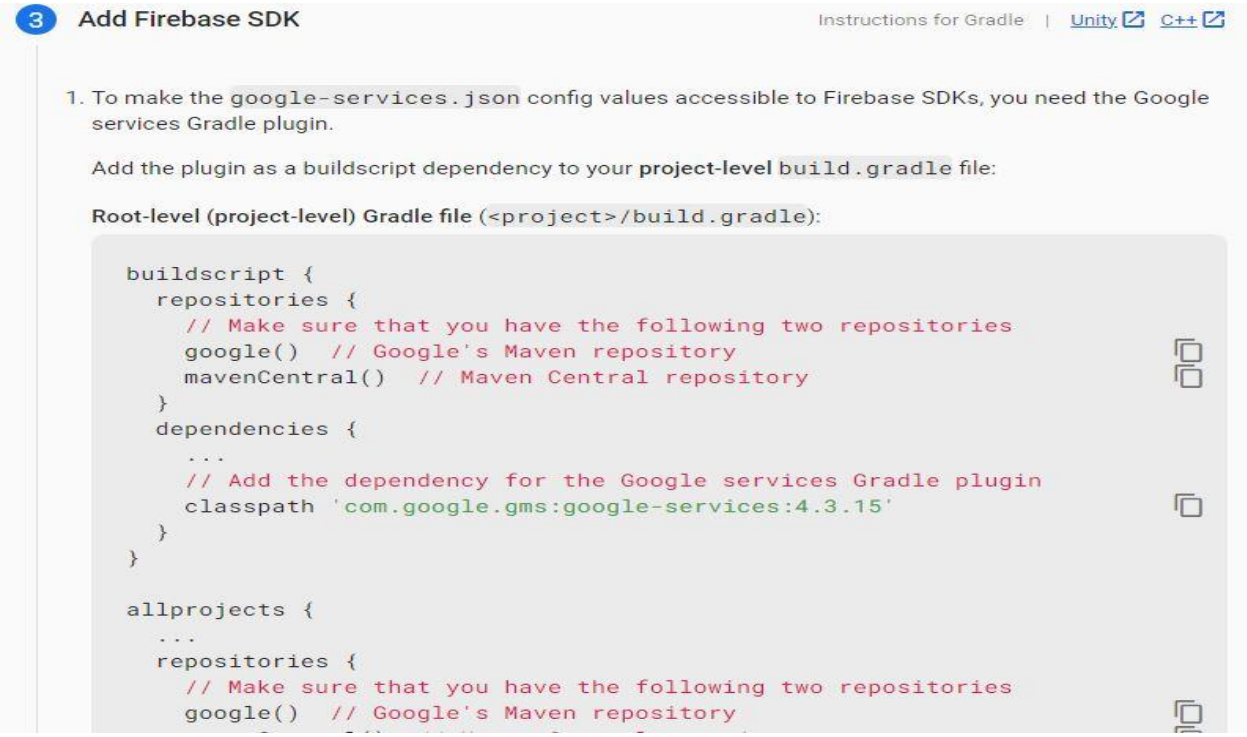


Figure 14. The first step of implementation.

- Add the Firebase SDK to my project then synchronize from the IDE.
- Execute the application to test the SDK's installation.
- Enter the Firebase console.



3 **Add Firebase SDK** Instructions for Gradle | Unity C++

1. To make the `google-services.json` config values accessible to Firebase SDKs, you need the Google services Gradle plugin.

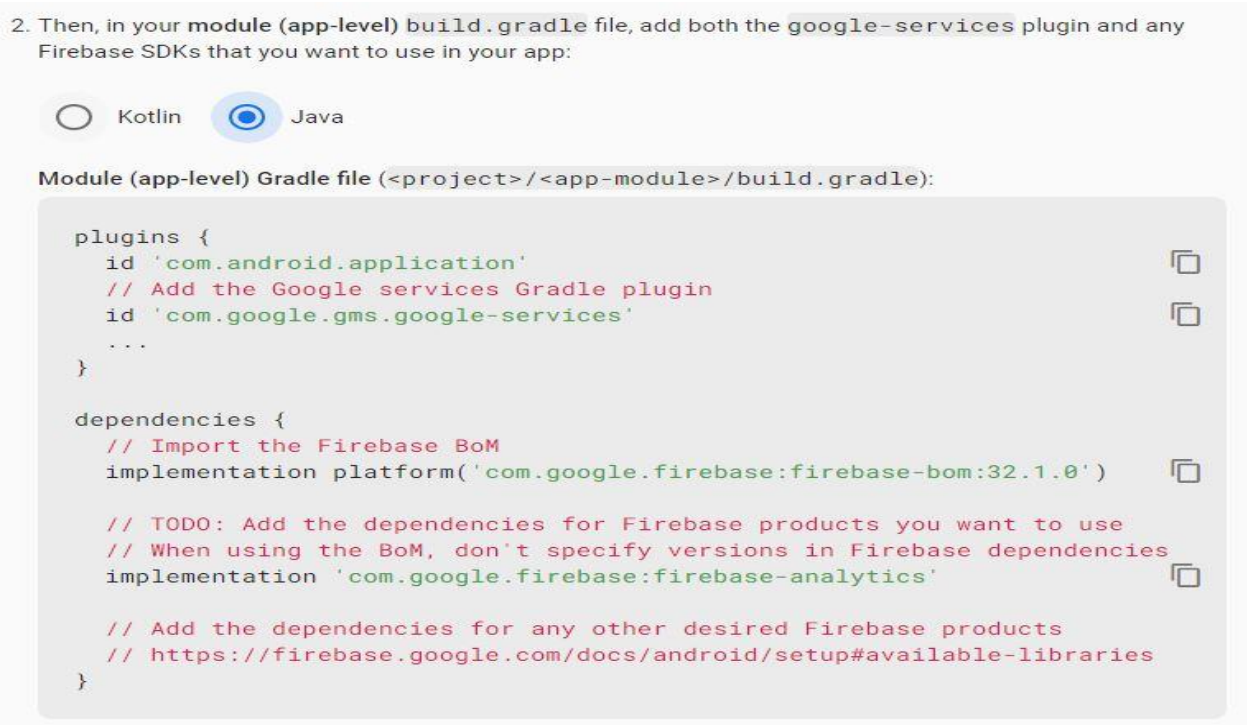
   Add the plugin as a buildscript dependency to your **project-level** `build.gradle` file:

   Root-level (project-level) Gradle file (`<project>/build.gradle`):

```
buildscript {
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
    mavenCentral()  // Maven Central repository
  }
  dependencies {
    ...
    // Add the dependency for the Google services Gradle plugin
    classpath 'com.google.gms:google-services:4.3.15'
  }
}

allprojects {
  ...
  repositories {
    // Make sure that you have the following two repositories
    google()  // Google's Maven repository
```

Figure 15. The second step of implementation.



2. Then, in your **module (app-level)** `build.gradle` file, add both the `google-services` plugin and any Firebase SDKs that you want to use in your app:

   ○ Kotlin ◉ Java

   Module (app-level) Gradle file (`<project>/<app-module>/build.gradle`):

```
plugins {
  id 'com.android.application'
  // Add the Google services Gradle plugin
  id 'com.google.gms.google-services'
  ...
}

dependencies {
  // Import the Firebase BoM
  implementation platform('com.google.firebase:firebase-bom:32.1.0')

  // TODO: Add the dependencies for Firebase products you want to use
  // When using the BoM, don't specify versions in Firebase dependencies
  implementation 'com.google.firebase:firebase-analytics'

  // Add the dependencies for any other desired Firebase products
  // https://firebase.google.com/docs/android/setup#available-libraries
}
```

Figure 16. Build. gradle file.

3.4.2. Firebase Authentication

Firebase provides a secure and easy way for users to sign into their applications. Developers can use Firebase Authentication to support email and password login, Google sign-in, Facebook login, and more [17].

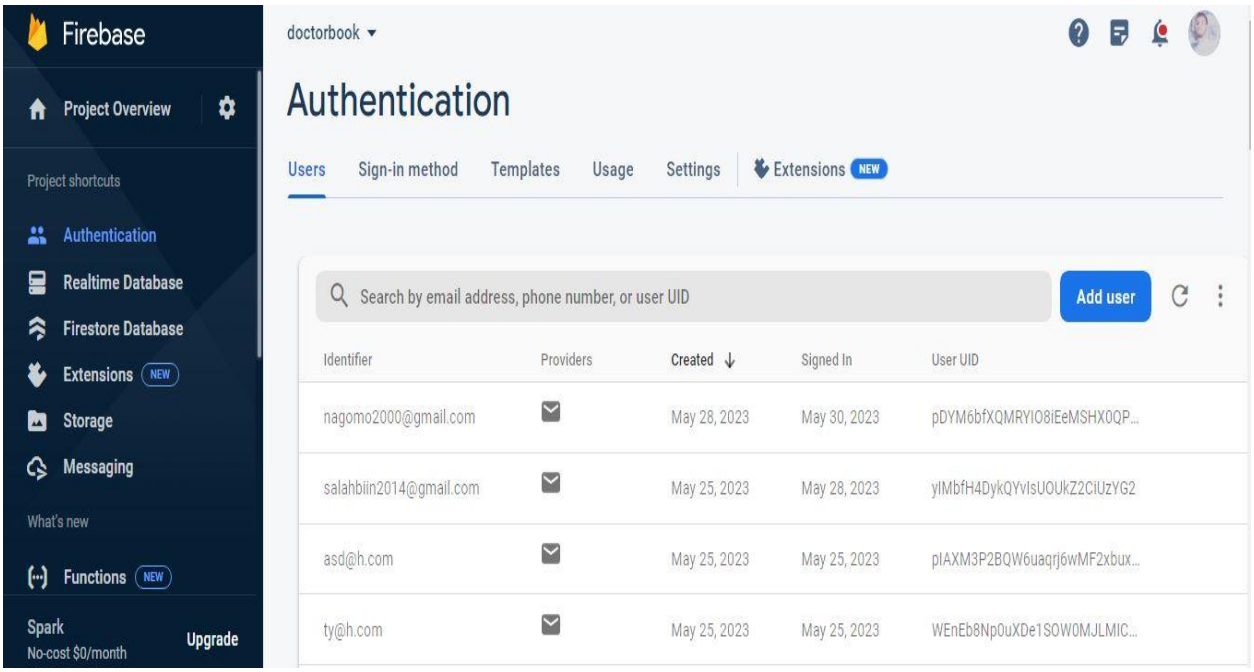


Figure 17.Firebase Authentication.

3.4.3. Firebase Database

The Firebase Real-Time Database is a cloud-hosted NoSQL database that lets organizations store and synchronize data in real-time across all their users' devices. This makes it easy to build applications that are always up to date, even when users are offline [17].

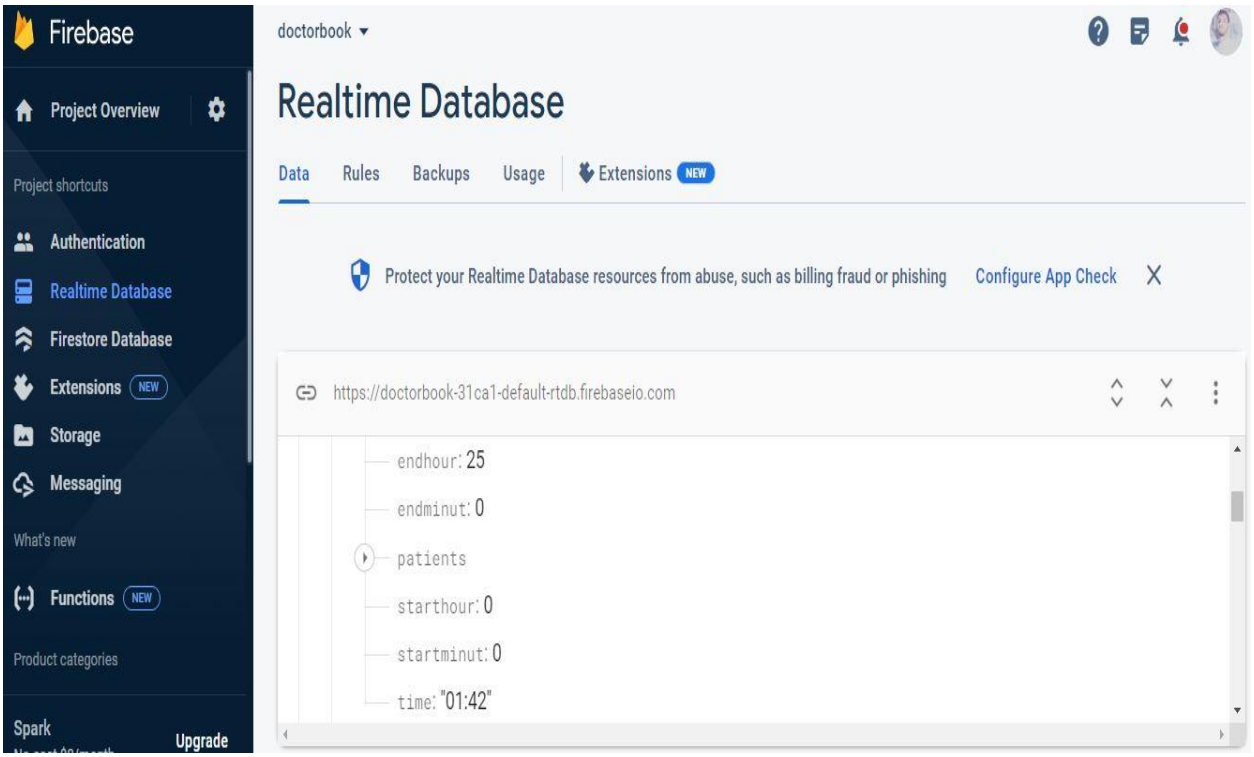


Figure 18.Firebase database.

3.4.4. Firebase Cloud Messaging

Firebase Cloud Messaging (FCM) is a service that lets businesses send messages to their users' devices, even if they aren't using the application. Developers can use FCM to send push notifications, update app content, and more [17].

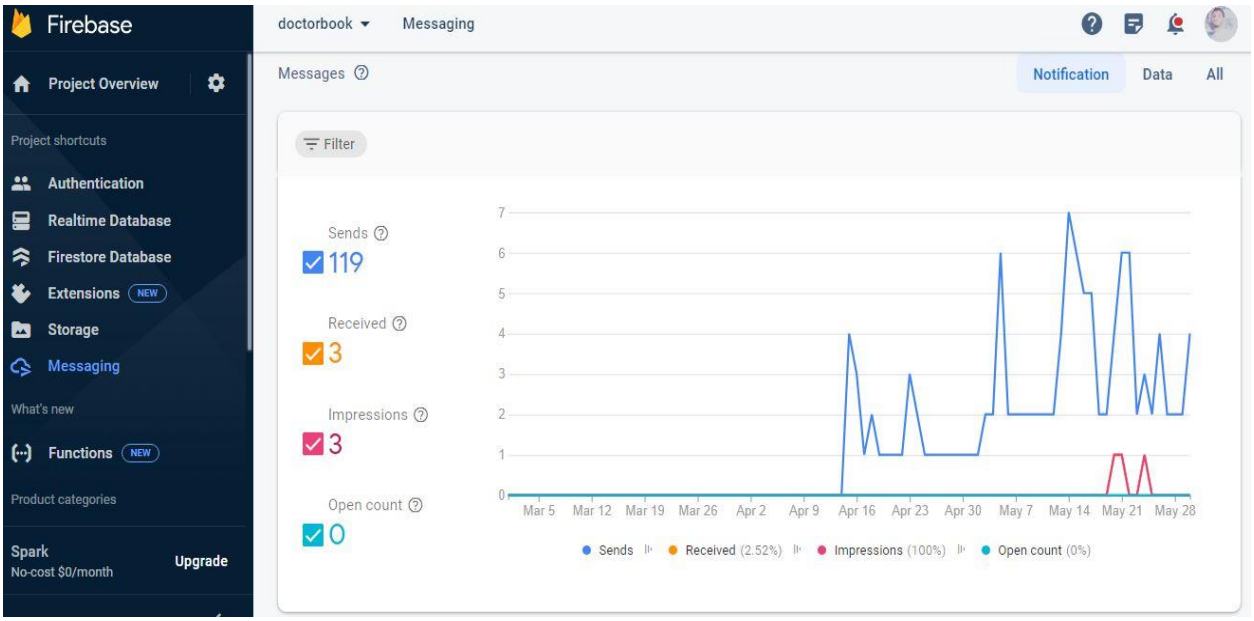


Figure 19.Firebase cloud messaging.

3.4.5. Firebase Storage

Firebase Storage is a Cloud Storage built for application developers who need to store and serve user-generated content, usually big files like photos or videos, it is mostly used or developed for photos and videos, but we might use this for other things like text files [18].

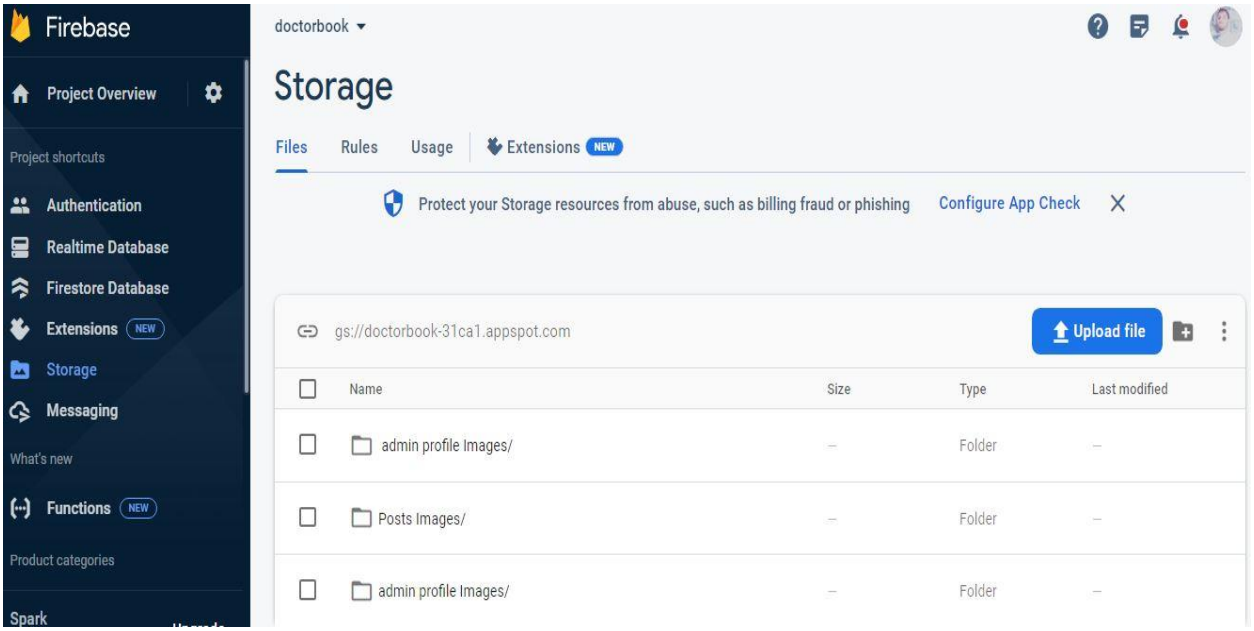


Figure 20.Firebase storage.

3.5. Presentation to the project interfaces

This section contains the presentation of our applications' main interfaces, starting with the logos of our two applications.

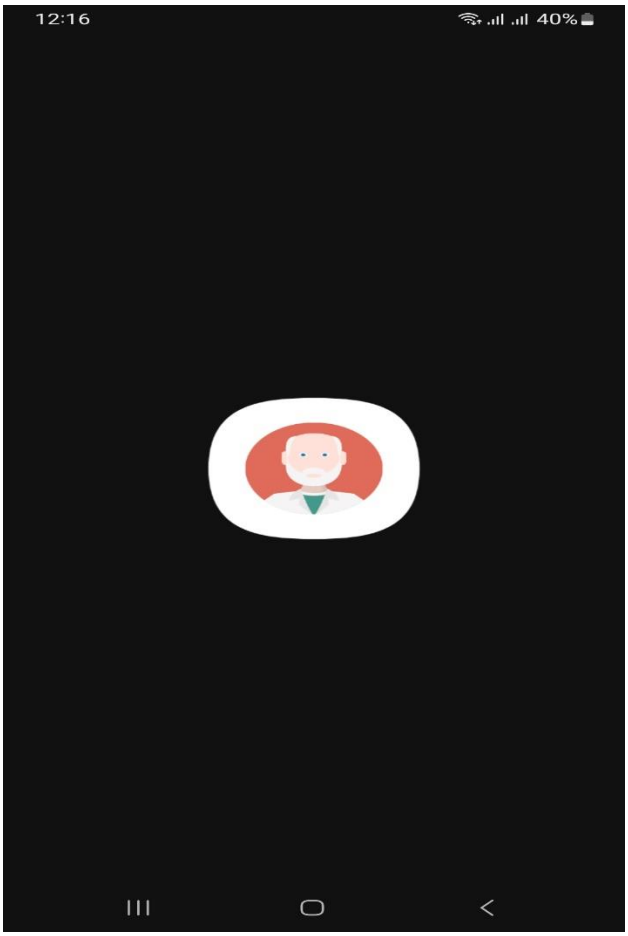


Figure 21.The logo of the admin's application.

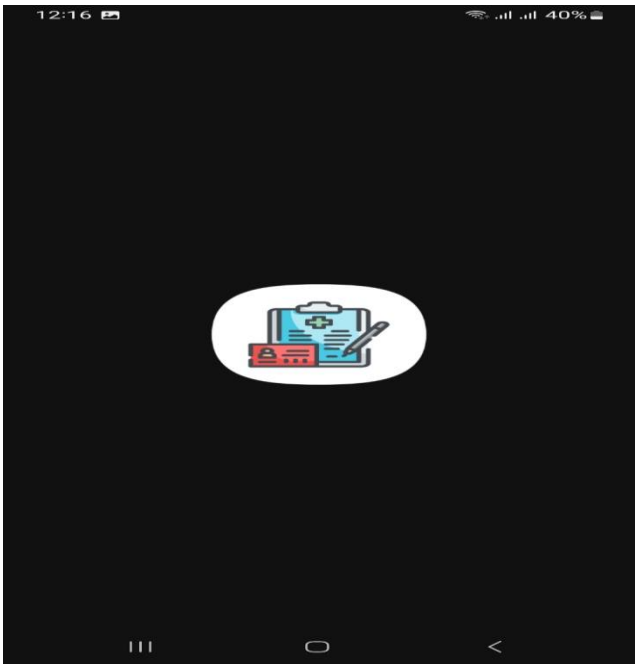


Figure 22.The logo of the patient's application.

3.5.1. Splash Screen

It is the displayed interface while loading the application, and contains a background image with loading animation.

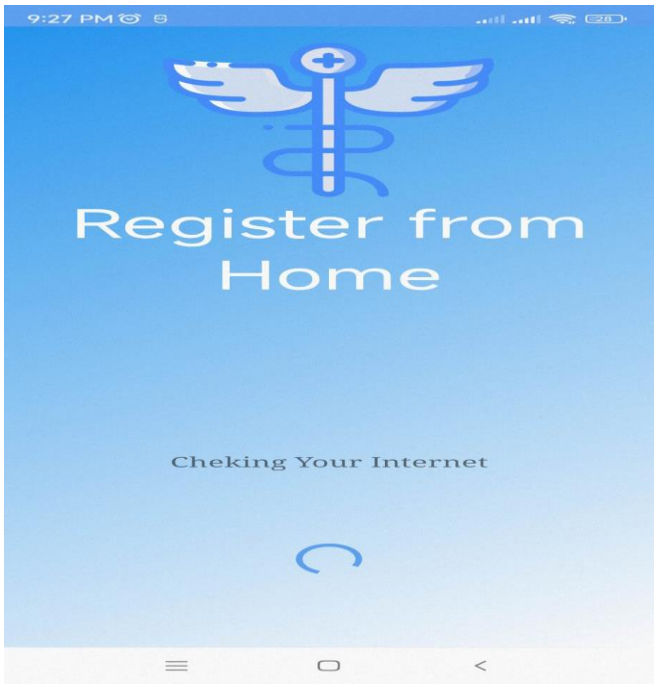


Figure 23.Splash screen of our applications.

3.5.2. Registration page

This page contains the application's logo with the required fields to create a new account in the application for both the user and admin.

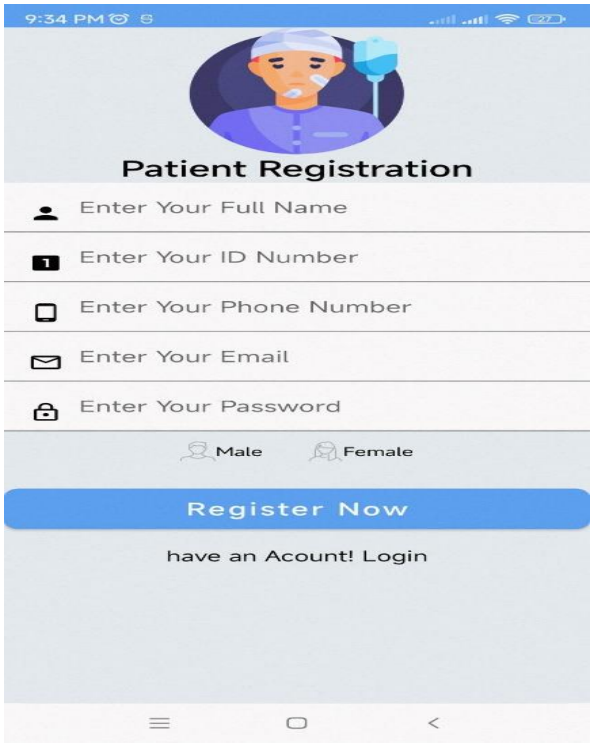


Figure 24.Registration page for patients.

3.5.3. Login page

For previously registered members.

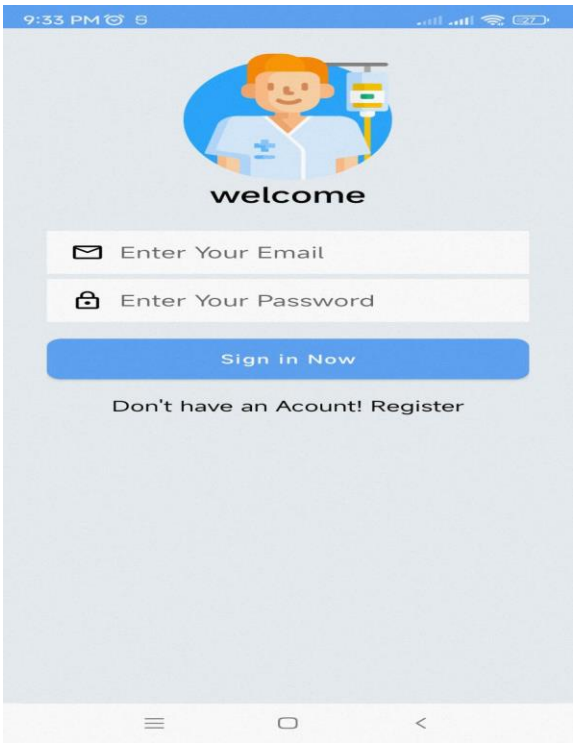


Figure 25.Login page for the patient.

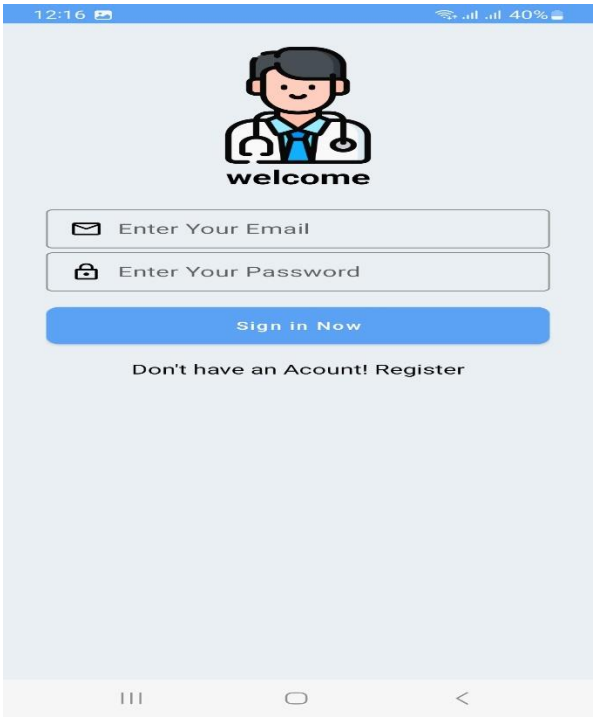


Figure 26.Login page for the doctor.

3.5.4. make appointment page

Contains the list of registered patients and the registration button to register in the list or cancel the registration.

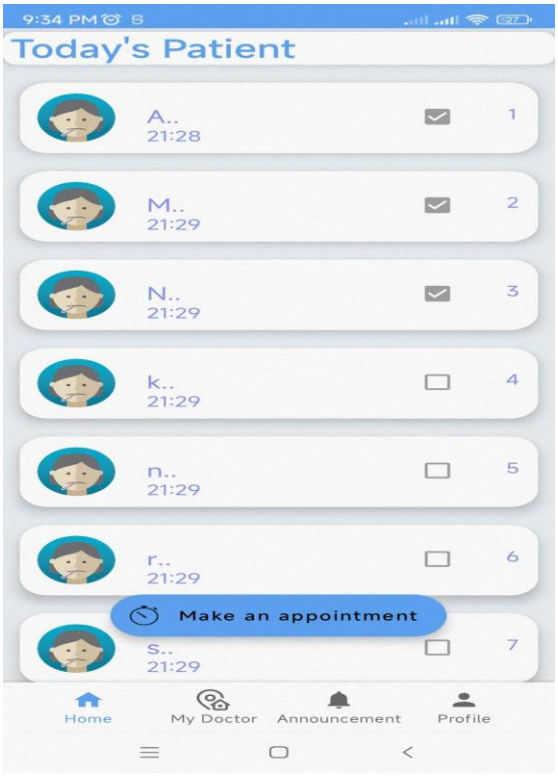


Figure 27.Patient home page.

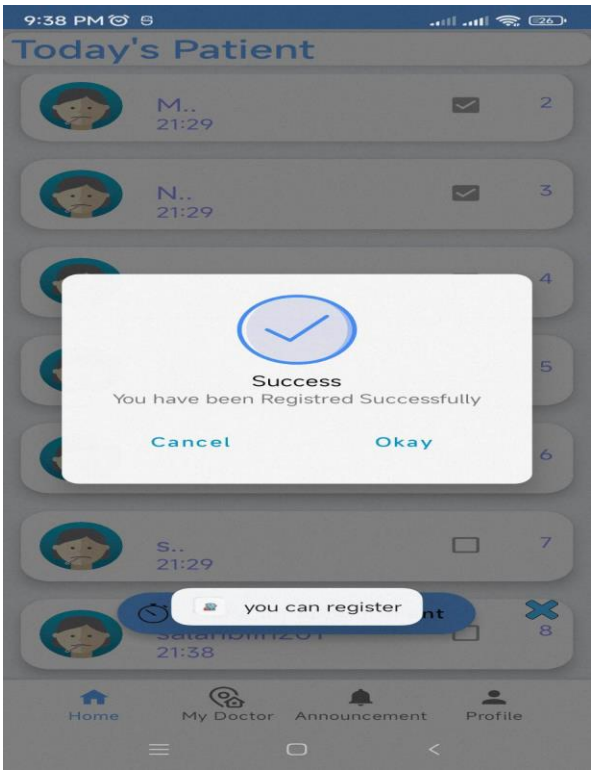


Figure 28. Make an appointment from the patient app.

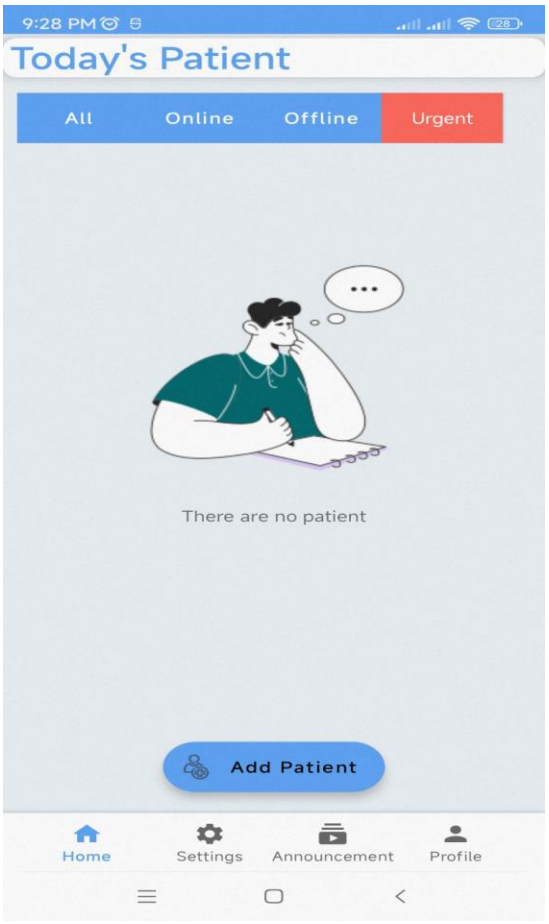


Figure 29.The admin home page.

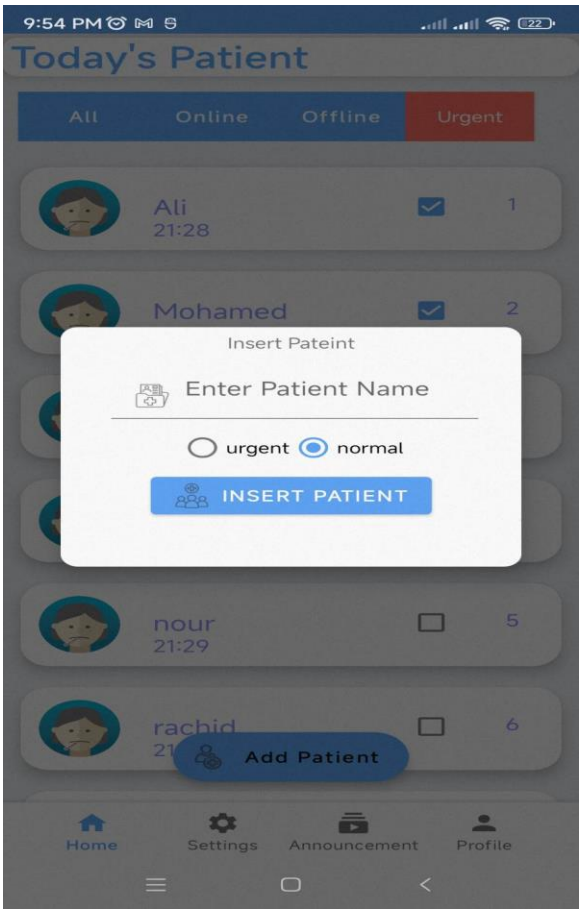


Figure 30. Make an appointment by admin.

3.5.5. Profile page

This page contains information about the admin.

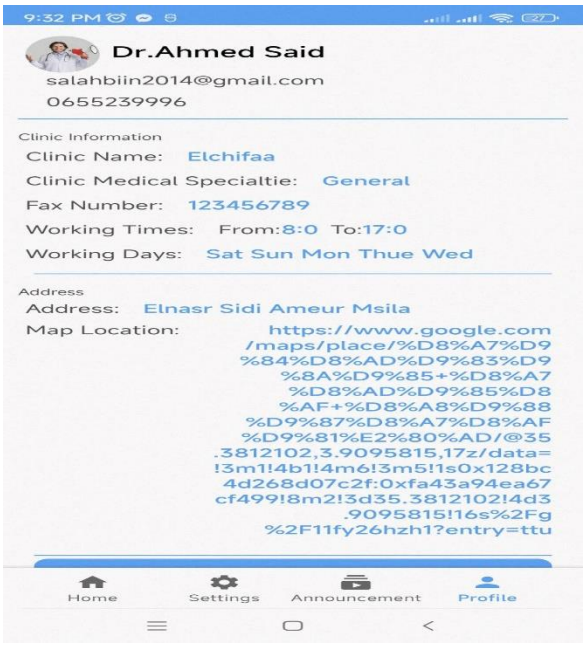


Figure 32.Doctor profile from admin app.

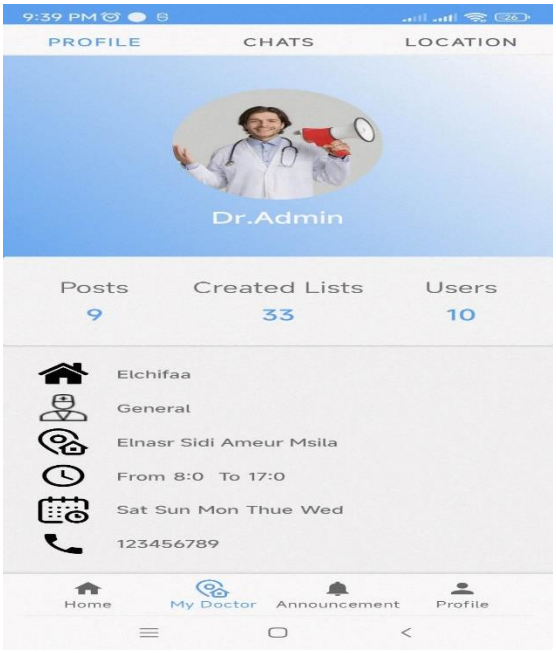


Figure 31.Doctor profile from patient app.

3.5.6. Announcement page

Where the admin can post and consult the announcements, the user can only consult them.

Figure 33.Announcements from Admin app.

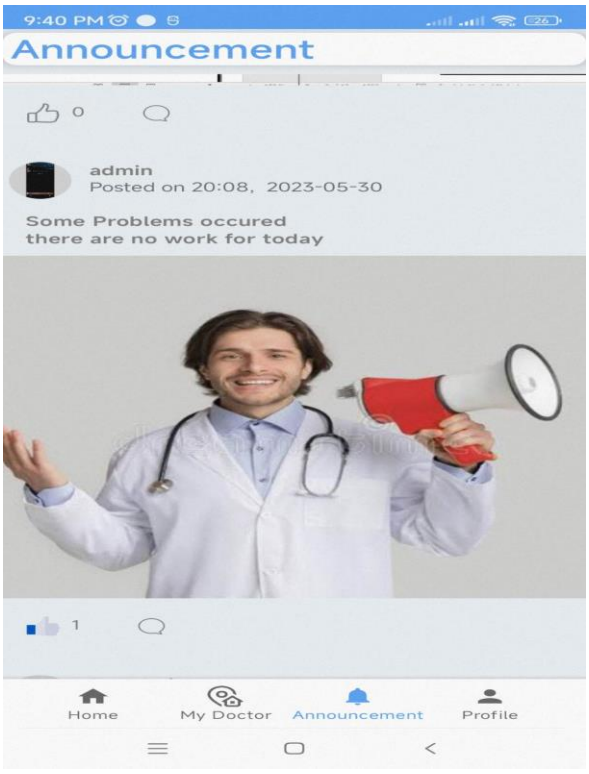


Figure 34.Announcements from the patient app.

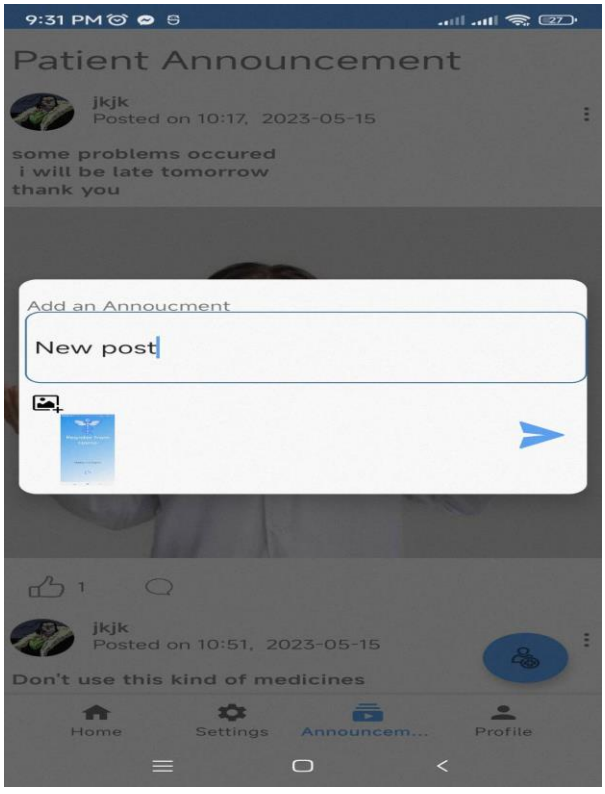


Figure 35. Posting an announcement.

3.5.7. Setting page

The admin has a setting page where he can choose the starting and ending time of the booking list, block users or turn on/off notifications … etc.

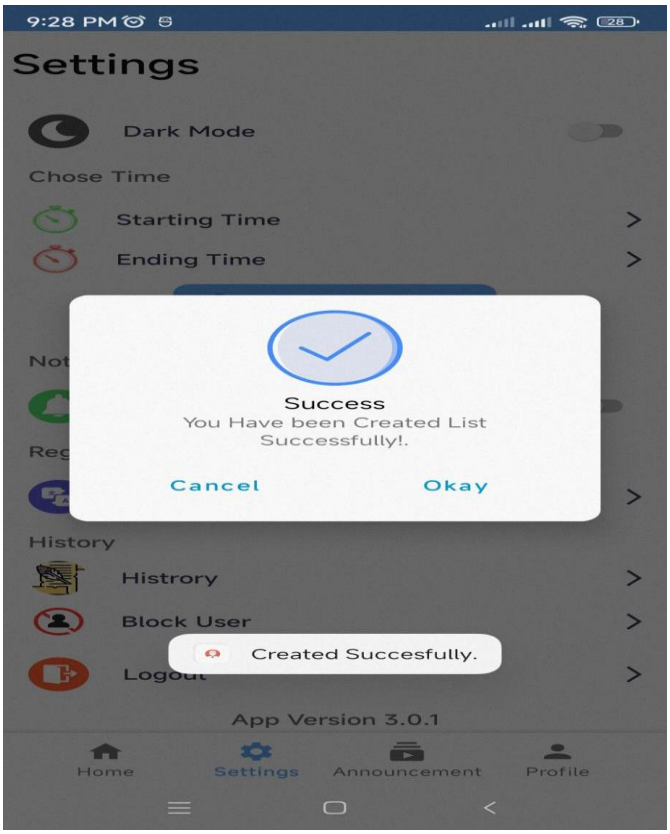


Figure 36. Settings of admin.

3.5.8. History page

Where the admin can find old booking lists.

Figure 37. History of booking lists.

3.5.9. Location page

The user can find the location of the clinic on this page.

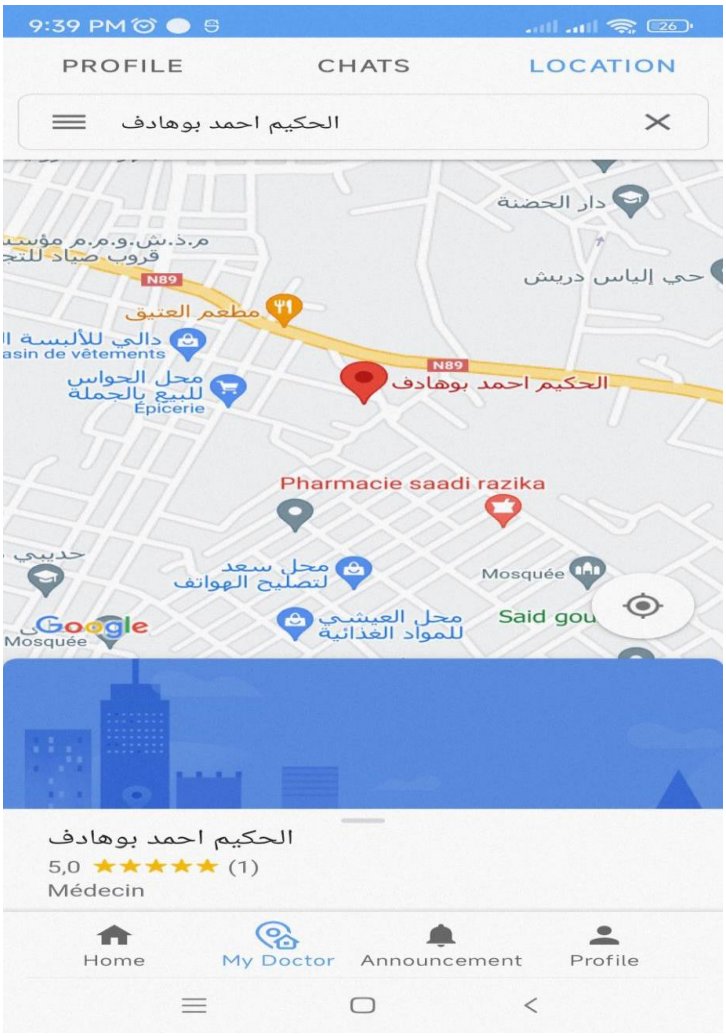


Figure 38.Location of the clinic for the patient.

3.6. Conclusion

In this last chapter, we have presented the development environment of our project and some used libraries, then a presentation of the most important interfaces of our applications. We tried to respect the IHM standard roles for the realization of the project.

GENERAL CONCLUSION

In today's fast-paced world, managing appointments efficiently is essential for businesses and individuals alike. With the advent of smartphones and the increasing reliance on mobile technology, the need for a convenient and user-friendly solution has become paramount. Introducing our mobile application for an online appointment booking system, a cutting-edge platform designed to streamline the process of scheduling appointments, saving time, and enhancing productivity

This work began by providing an overview of smartphone applications, discussing their types, categories, and various platforms. We also introduced the Android universe and its different versions. Following that, we presented the problem of appointment scheduling and its solution through mobile applications. In the second chapter of this dissertation, we delved into the analysis and design process of the project, which included specifying requirements and creating UML diagrams. The last chapter focused on the practical tools utilized to develop our project, encompassing the development environment, libraries used, and the implementation of the database. Lastly, we showcased several application interfaces accompanied by descriptive explanations.

Our project may seem unhelpful or less important, but as long as it touches human souls it will be very important and very useful to save people’s lives. We tried a lot to facilitate the use of our applications.

At the end of the time, we reached 100% of our goals; because of the short time, we did not add some more features like the option of changing the language to Arabic and dark mode. If we had time in the future, we would add some features like more languages and more doctors from different specialties. We will seek to expand the application to other ways in the future.

Bibliography

[1] https://www.techtarget.com/whatis/definition/mobile-app, viewed the 12/03/2023.

[2] https://www.techtarget.com/whatis/search/query?q=definition+of+native+app, viewed the 12/03/2023.

[3] https://www.techtarget.com/whatis/search/query?q=definition+of+web+app, viewed the 12/03/2023.

[4] https://www.techtarget.com/whatis/search/query?q=definition+of+hybrid+app, viewed the 12/03/2023.

[5] https://www.ecommerce-nation.com/various-categories-types-of-mobile-applications, viewed the 15/03/2023.

[6] Calimag, J. N., et al. "Ubiquitous learning environment using android mobile application." *International Journal of Research in Engineering & Technology* 2.2 (2014): 119-128.

[7] https://www.techtarget.com/whatis/search/query?q=apple+IOS, viewed the 16/03/2023.

[8] https://www.techopedia.com/definition/25196/blackberry-os, viewed the 16/03/2023.

[9] https://www.techopedia.com/definition/26924/windows-mobile, viewed the 17/03/2023.

[10] https://www.tutorialspoint.com/android/android_overview.htm, viewed the 23/03/2023.

[11] https://www.tutorialspoint.com/android/android_architecture.htm, viewed the 23/03/2023

[12] https://www.techopedia.com/definition/3243/unified-modeling-language-uml, viewed the 19/04/2023.

[13] J. Horton, Android Programming with Kotlin for Beginners, BIRMINGHAM - MUMBAI: Packt Publishing, 2019.

[14] https://www.oracle.com/java/technologies/jdk9-readme.html, viewed the 12/05/2023.

[15] https://www.w3.org/standards/xml/core, viewed the 12/05/2023.

[16] https://www.java.com/en/download/help/whatis_java.html, viewed the 12/05/2023.

[17] https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase, viewed the 18/05/2023.

[18] https://www.javatpoint.com/firebase-cloud-storage, viewed the 18/05/2023.

[19] J. R. J. Grady Booch, The Unified Modeling Language User Guide, USA,Canada: Addison-Wesley Publishing Co, 1998.

Abstract

This dissertation aims to create a smartphone application for making doctor's appointments online for both doctor and patient. The evolution of technology helped us to apply it in online appointment-making and take advantage of mobile use in this domain. We first introduced some theoretical concepts related to mobile application development and the universe of Android, and then, we used the UML modeling method to describe the main features and advantages of our project. Finally, we have reached the realization phase where we used integrated developing environments, which are android studio, and Google Firebase services for the database implementation, these tools made the development easier, and we concluded the dissertation with screenshots of some interfaces from our application.

Keywords: Android, UML, Firebase, Mobile application, Doctor's appointments making.

Résumé

Cette thèse a été écrite pour créer une application mobile permettant de prendre des rendez-vous en ligne chez un médecin, l'application est destinée pour le médecin et le patient. L'évolution de la technologie nous a permis de l'appliquer dans la prise de rendez-vous en ligne et de profiter des avantages de l'utilisation mobile dans ce domaine. Nous avons d'abord introduit quelques notions théoriques liées au développement d'applications mobiles et à l'univers Android, puis nous avons utilisé la méthode de modélisation UML pour décrire les principales fonctionnalités et avantages de notre projet. Enfin, nous sommes arrivés la phase de réalisation où nous avons utilisé l'environnement de développement intégré Android studio, et les services de 'Google Firebase' pour l'implémentation de la base de données ce qui a facilité le développement, nous avons conclu ce mémoire avec quelques interfaces de notre application.

Mots clées : Android, UML, Firebase, Application mobile, Prise de rendez-vous chez le médecin.

ملخّص

تمت كتابة هذه المذكرة لإنشاء تطبيق محمول لحجز موعد لدى الطبيب عبر الانترنت لفائدة كل من الطبيب والمريض. وقد ساعدنا التطور التكنولوجي السريع على تطبيقها في مجال الحجوزات عن بعد والاستفادة أكثر من مزايا الهواتف المحمولة في هذا المجال.

قدمنا في بداية الرسالة بعض المفاهيم النظرية عن تطوير تطبيقات الهواتف الذكية بالإضافة إلى عموميات حول نظام أندرويد، ثم استخدمنا لغة النمذجة الموحدة "UML" لوصف مميزات وخصائص مشروعنا، كما شهدنا بعد ذلك مرحلة الإنجاز حيث استخدمنا برنامج اندرويد ستوديو"Android studio"، كما قمنا بالاستعانة بخدمات "Google Firebase" لضبط قاعدة البيانات مما جعل التطوير أسهل. وأخيرا قمنا بعرض بعض واجهات التطبيق الخاص بنا.

كلمات مفتاحية: اندرويد، UML، تطبيق محمول، Firebase، حجز موعد لدى طبيب.