People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

Echahid Hamma Lakhdar University El-Oued

Faculty of Exact Sciences

Computer Science department

Option Computer Systems

By:

- ***DEBBAR Abderrahmane***
- ***SOUFIA Mohammed***
- ***MERKHOUFI Seifeddine***

Title:

Development of mobile application for the management of lawyer office

Report Submitted in partial fulfilment of the

Requirements for the degree of

License in computer science

Sustained on 08-06-2022 in front of the jury composed from:

Mrs. Kholladi nadjoua houda	**Examiner**
Mrs. Guettas Chourouk	**Examiner**
Mr. Berdjouh chafik	**Supervisor**

ملخص

الهدف من مشروعنا هو إنشاء تطبيق للهاتف المحمول يساعد في إدارة مكتب محامي، وإبقاء العملاء على اطلاع بشأن وضعية قضاياهم وتاريخ المحاكمة.

لأجل تحقيق هدفنا اخترنا التصميم باستخدام UML بجانب 2TUP كدورة حياة لتطوير برنامج. ثم استخدمنا Flutter و Firebase لإنجاز التطبيق.

الكلمات المفتاحية:

تطبيق هاتف محمول، مكتب محامي ,2TUP,Flutter,Firebase,UML.

Abstract

The goal of our project is creating a mobile application that helps in managing a lawyer's firm, and keeping clients informed about the statue of their cases and trial date.

To achieve our objectives, we have chosen to model with UML aside to the 2TUP approach. In order to realize our application, we used flutter and Firebase.

Keywords:

Mobile application, Lawyer's firm, UML, 2TUP, Flutter, Firebase.

Table of Contents

List of tables and Figures

List of Tables

List of Figures

General Introduction

Today, with the advancement of new technologies, the human being is increasingly looking for more than just ways to make his life easier in any field, by utilizing the various types of technologies that have been developed to automate life wherever and at any time. While mobile applications are among the most popular and attractive innovations these days.

A mobile application, often known as a mobile app, is a computer program or software application that runs on a mobile device such a phone, tablet, or smartwatch. they are typically smaller, more ergonomic, and quicker to use.

So, among the fields that we are interested in is the field of lawyers.

A lawyer is a jurist whose primary responsibility is to advise and defend their clients in court, whether individual or legal entity, by defending for their rights to be protected and, more broadly, to represent them. The lawyer's job is to give legal advice and edit documents.

National legislation has made getting represented by a lawyer mandatory in specific cases, particularly to safeguard the rights of defense before certain tribunals.

Lawyers' profession has evolved into a delicate technique that necessitates continuous communication between the lawyer and his clients. A lawyer's primary job is to defend human rights litigants before the courts or any disciplinary body, such as a disciplinary authority. At the core of his criticism, the need to develop a mobile application will be a commitment, to facilitate the work of the lawyer in a flexible and timely manner.

The objective of this work is to develop a mobile application for the management of a lawyer office. To fulfill this objective, we have set the following aims:

– Facilitate communication between lawyer and client.

– Reduction of communication time between these actors.

– Data organization.

– Ensure better management and consistency of information.

Our report consists of three chapter:

- We start by introducing the various concepts related to a law firm, detailed problems and our proposed solution.
- Second, we present our software design process start collecting user needs and end by a detailed software design.
- Third, we describe our choice of tools and technologies, the motivations behind it and present our user interfaces.

Finally, we conclude the search with future vision

Chapter 01:

The preliminary study

Chapter01: The preliminary study

1-1-Introduction

The preliminary study is the first phase of our development process. it consists of identifying functional and operational needs, using mostly text, or very simple diagrams.

So, in this chapter we are going to study this idea in real world, and we will see the following points:

- Presentation of the project.
- Presentation of terminologies.
- The development method.
- Specification of needs.
- Identifying messages.
- Problem position.
- Conclusion.

1-2- Presentation of the project

Our goal for this project is to create a mobile application that allows you to manage a law firm. It's a matter of defining the management's responsibilities, updating data, organizing data collected from lawyers in order to design basic files, strengthening control and confrontation, and ensuring better information management and consistency. The following are the primary features of our application:

– Case management (Create, edit and archive cases).

– Invoice management (Create and send invoices).

– Client relationship management (send trial session).

– Requests management (View new requests).

1-3-Presentation of terminologies

- **Court**

Court, also called court of law, a person or body of persons having judicial authority to hear and resolve cases. The word court, which originally meant simply an enclosed place, or place where judicial proceedings are held.

- **Lawyer**

A lawyer is a person whose job it is to protect his clients' interests. He applies his expertise to the problem that his customer has presented. In general, he chooses his clientele absolutely independently. and is not bound by any rules or regulations. In this way, he serves as a liberal. The lawyer can also represent legal individuals, businesses, and other entities.**[1]**

- **Court case**

A court case is a problem between two parties that is handled by a judge or another legal process. A court case can be civil or criminal case. Every legal case has an accuser and one or more defendants.

- **Trial session**

A trial in law is a meeting of judgment between a number of disputing parties in a matter, to provide information in the form of legal evidence in a judicial session inside a court. A judge, jury, or another authority is entrusted with the task of deciding the dispute at hand.

- **Proxy**

A proxy document is a document that authorizes someone to perform the duty of another.

1-4- The development method

Development method It's one of the most essential tasks while working on a project that consider the project diagrams and the process that it follows.

We will utilize the Unified Modeling Language (UML) Diagrams for this project, which are ideal for this type of project, as well as the Two Track Unified Process (2TUP) for software development.

- **Two Track Unified Process (2TUP):**

2TUP, is a software development technique that uses the Unified Process method.

The 2TUP has a Y-shaped development cycle that divides the technical and functional components of the project. **[2]**

The process then revolves around three essential phases:

- **A technical branch:**

Capitalize on technical know-how and / or technical constraints. The techniques developed for the system are independent of the functions to be performed.

- **A functional branch:**

Capitalize on knowledge of the company's business. This brunch collection the functional needs, which produces a model focused on the “business” of end users.

- **A Realization phase:**

Consists in union the two branches together, which making it possible to carry out an application design and finally the delivery of a solution adapted to the needs. Figure1.1 explains the Y development process.

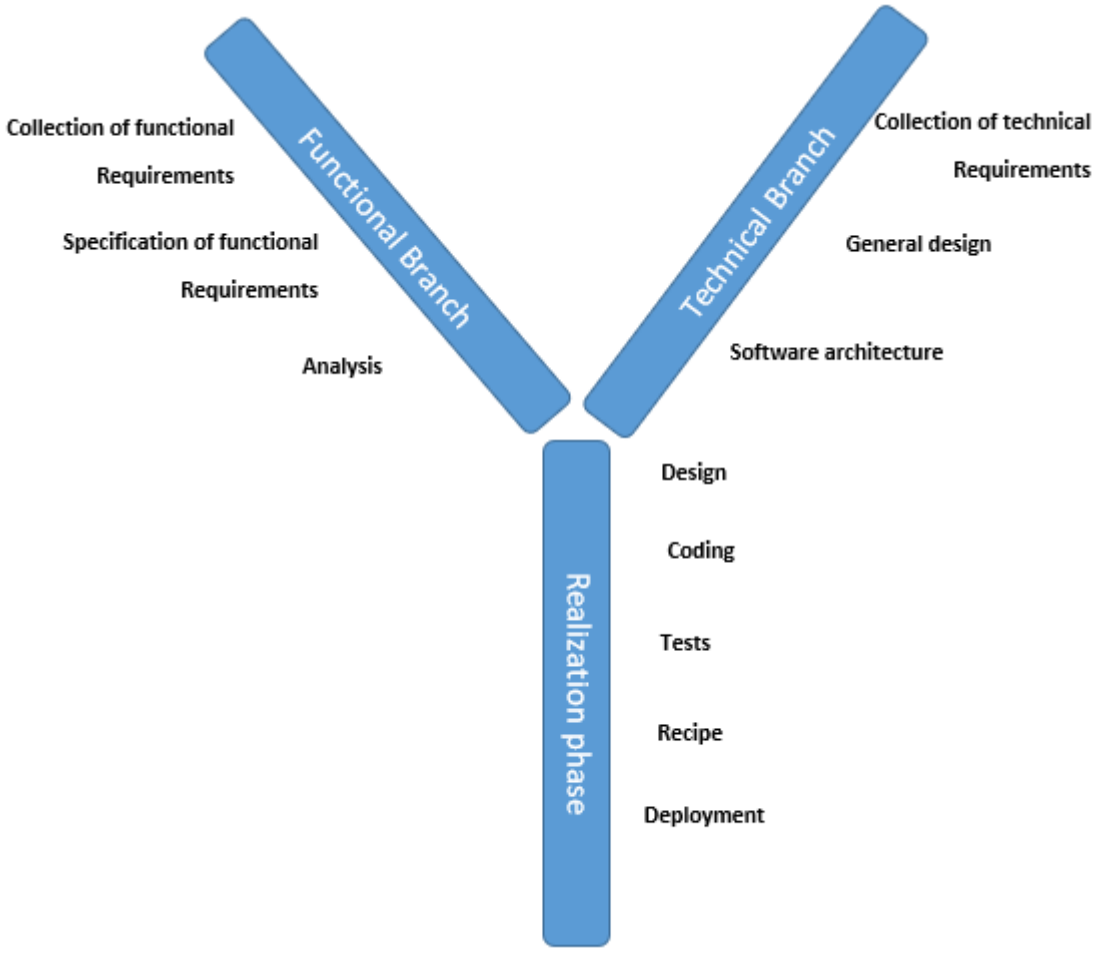

Figure 1.1 The Y development process

1-5- Specification of needs

1-5-1- Specification of functional requirements

After a study of the system, this part is reserved for the description of the requirements functions of the different actors of the application. These needs are grouped into use case diagrams.

- **Consult Request**

The lawyer consults the list of client requests.

- **Confirm Request**

The lawyer after the consultation the requests of the will either send a message of confirmation or a message of refuse.

- **Check Clients**

The lawyer can check all the clients of the system.

- **Add court session**

The lawyer can send a date of a court sessions to the Client.

- **Manage cases**

The lawyer can check and manage cases (send court session, create invoice, edit and archive).

- **Manage Invoices**

The lawyer can create the invoices and send it to clients as a pdf file.

- **Perform statistics**

Statistics are a necessary requirement for gaining a global perspective. We use the following examples to illustrate the various scenarios:

-The total number of cases.

- The total number of clients.

- The number of requests.

- **Request for a case**

The client requests for a new case.

- **Establish proxy**

The clients can create a proxy as a pdf file.

- **Consult cases**

The client consults the judgment date.

1-5-2- Specification of non-functional requirements

Non-functional needs refer to all of the technological, ergonomic, and aesthetic restrictions that the system must overcome in order to be realized and perform properly. **[3]**

In terms of our application, we have satisfied the following requirements:

- **Authentication**

Each user of the application must authenticate with a username and a password, so that he can use the system.

- **Graphical interfaces**

This application's user interface must be clear and simple, as well as well-organized in terms of graphic design, color selection, and style.

- **Speed of access**

The system must be able to respond to user requests in real time.

- **Security**

User data must be protected.

1-6- Identification of actors

An actor is a typical system user. Rather being a natural person, it reflects a responsibility relationship to the system or a position. As a result, it is an external device that interacts directly with the system by sending and receiving messages:

- **The lawyer**

– Supervisor and system administrator.

– The lawyer ensures the consultation of the requests and confirms these requests and consults each client's file.

– Manage court session, invoice and Manage case history.

– Perform statistics.

- **The client**

He has the right to follow the progress of his case, know at any time financial situation. Thus, to print his invoice, Consult the date of the session.

1-7- Identifying messages

A message is a one-way communication specification between objects that transport data with the goal of initiating an activity in the receiver. Normally, a message is related with two types of events a send event and a receive event **[4]**:

• **Messages issued by the system**

– Display of the list of requests.

– Display of the list of customers.

– The form (to add,) court session.

– The form (to add) invoice.

– The form (to add, modify, archive) case.

– Statistics result.

• **Messages received by the system**

– Entering authentication information.

– Requests the list of customer requests.

– Requests the list of customers.

– Request to establish proxy.

– Request (to add) court session.

– Request (to add) invoice.

– Request (modify, archive) Case.

– Request to view invoice.

– Request to consult case.

– Request to consult a court session

1-8-problem position

1-8-1- Issue

Lawyers' profession has evolved into a delicate operation that necessitates continual communication between the lawyer and his clients.

Clients typically want to consult legal procedures as soon as possible after they are carried out in their business, lawyers prefer direct communication with customers and a flexible, real-time approach to ensure the success of operations. This type of client-lawyer communication is typically conducted over the phone through direct contact, which causes a delay in information transfer between these two actors.

1-8-2- Objectives

The goal of our project is to design and implement a mobile application for the management of a law business that is both fast and efficient in order to solve the challenges mentioned above on the one hand, and to integrate the use of new techniques in our daily lives, which can be done by everyone. As a result, our application must address the following requirements:

– Facilitate communication between lawyer and client.

– Reduction of communication time between these actors.

– Data organization.

– Ensure better management and consistency of information

1-9- Conclusion

At the end of this chapter, we discussed the various requirements our application's functionalities, whether functional or non-functional, as well as the various actors interacting with the system. We can say that this step is absolutely essential for progressing in the development of our project and preparing for the next step.

Chapter 02: Analyzing and Conception

Chapter 02: Analyzing and Conception

2-1- Introduction

After defining the functional needs necessary for the operation of our application, in this chapter we present it's needs in form of use case diagram and each use case as a sequence diagram, and we are going to identify the general class diagram and explain it as follows:

- Identifying the use case diagram.
- Detailing each use case.
- The class diagram.

2-2- Identifying the use case diagram

2-2-1-The use case diagram

Use case diagram is a graphical representation of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has and will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or ellipses. The actors are often shown as stick figures.**[5]**

Our general use case diagram is presented in the Figure 2.1 .

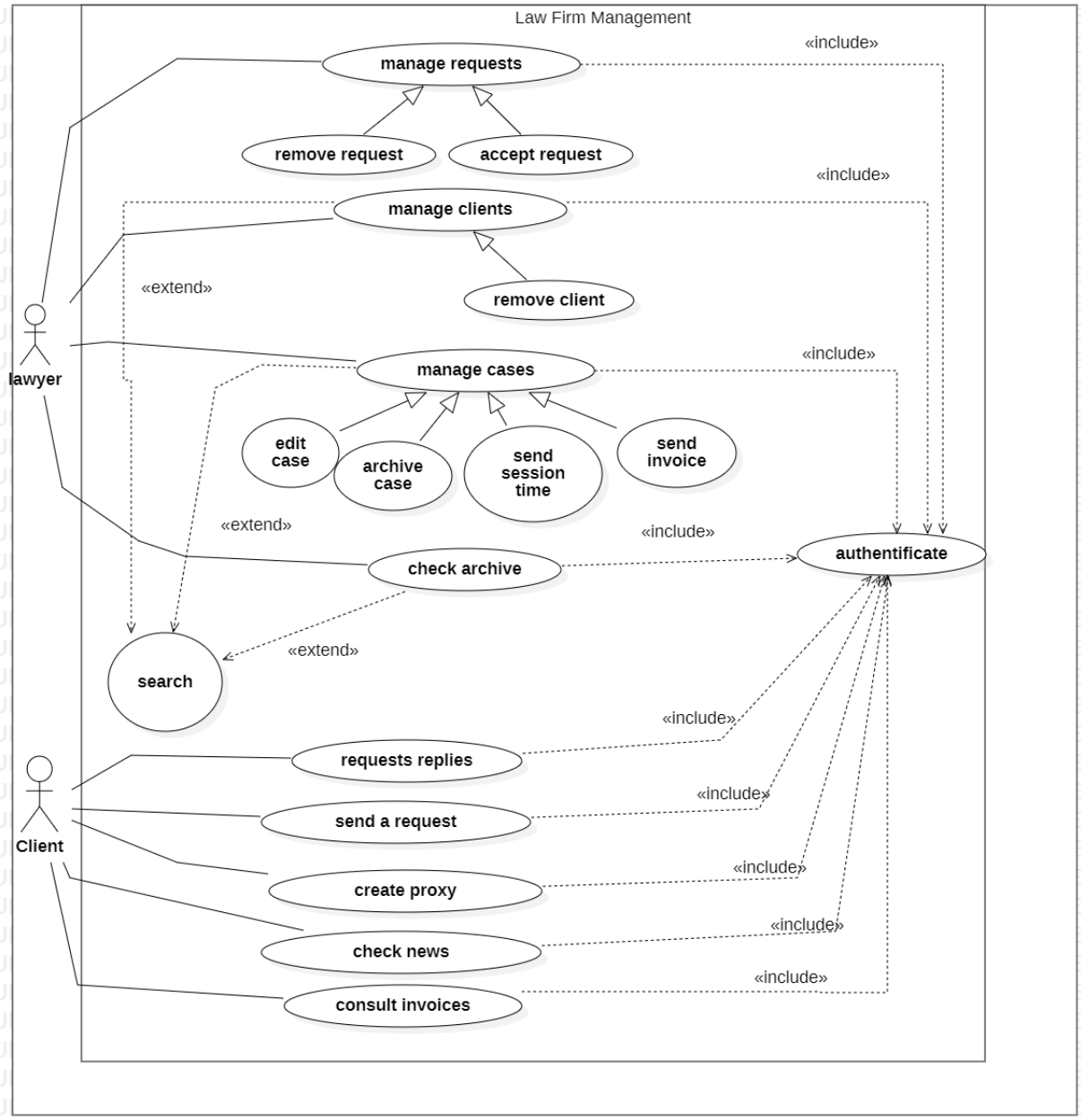

Figure 2.1 The general use case diagram.

2-2-2-The table of use cases

As shown in Table 2.1

Use cases	Actors
Login(authentication)	Lawyer, client
Manage requests	Lawyer
Manage clients	Lawyer
Manage cases	Lawyer
Check archive	Lawyer
Search	Lawyer
Create proxy	Client
Check news	Client
Send request	Client
Consult invoices	Client
Requests replies	Client

Table 2.1 The table of use cases

2-3- Detailing each use case

- **Login (authentication) use case** (As shown in Figure 2.2 and Figure 2.3)

The goal here is to access to the system

the actions

- User enter email, password
- The system verifies the information valid or not

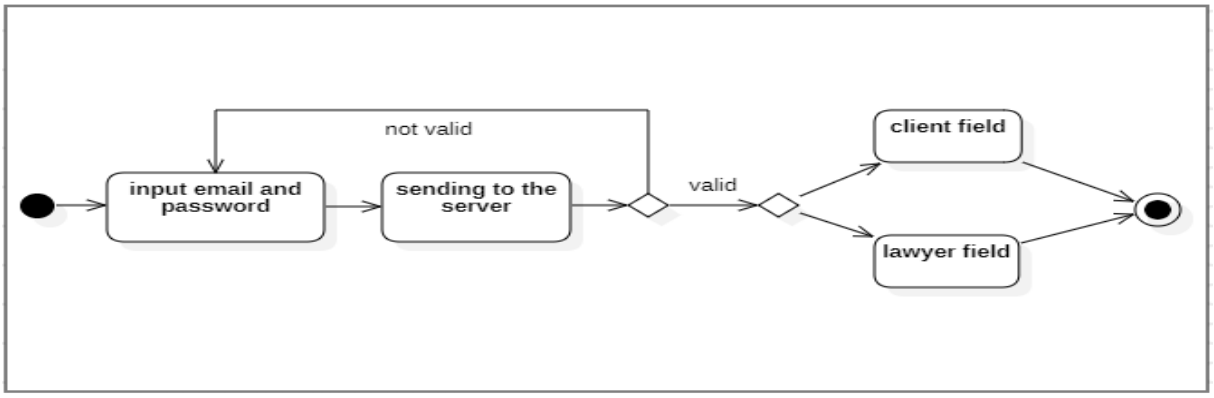

Figure 2.2 Login activity diagram

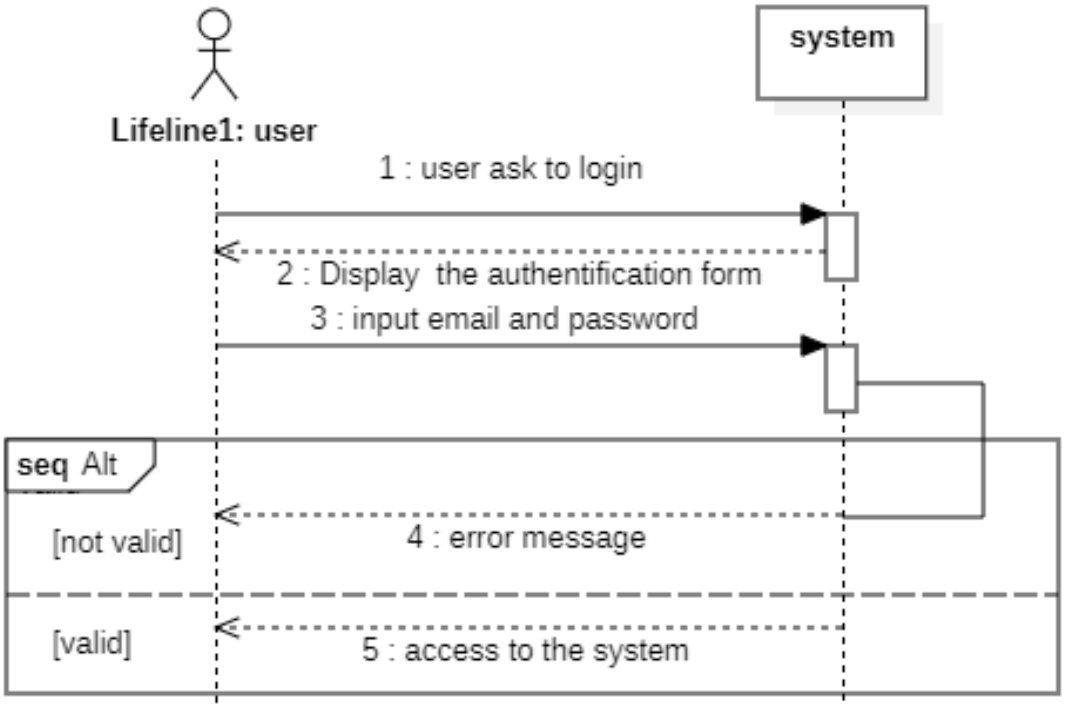

Figure 2.3 Login sequence diagram

- **Managing requests use case** (As shown in Figure 2.4 and Figure 2.5)

The goal here that the lawyer can check the requests of clients

the actions

- Accept request
- Remove request

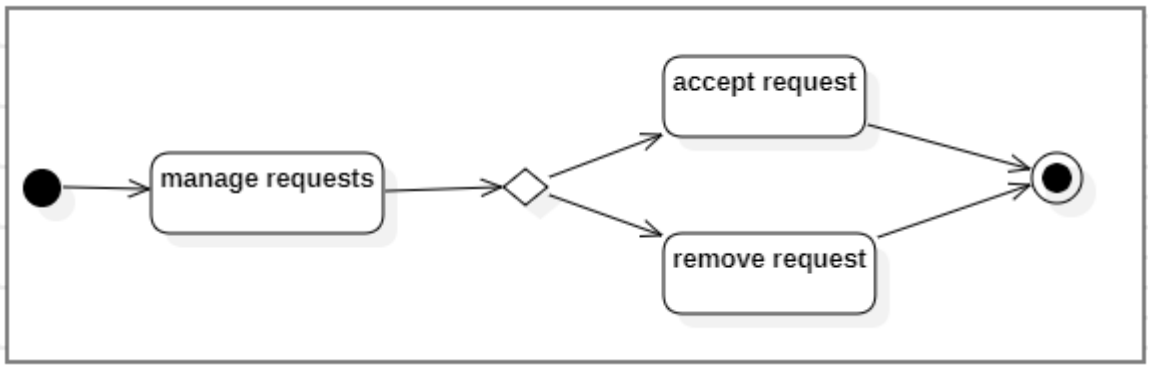

Figure 2.4 Managing requests activity diagram

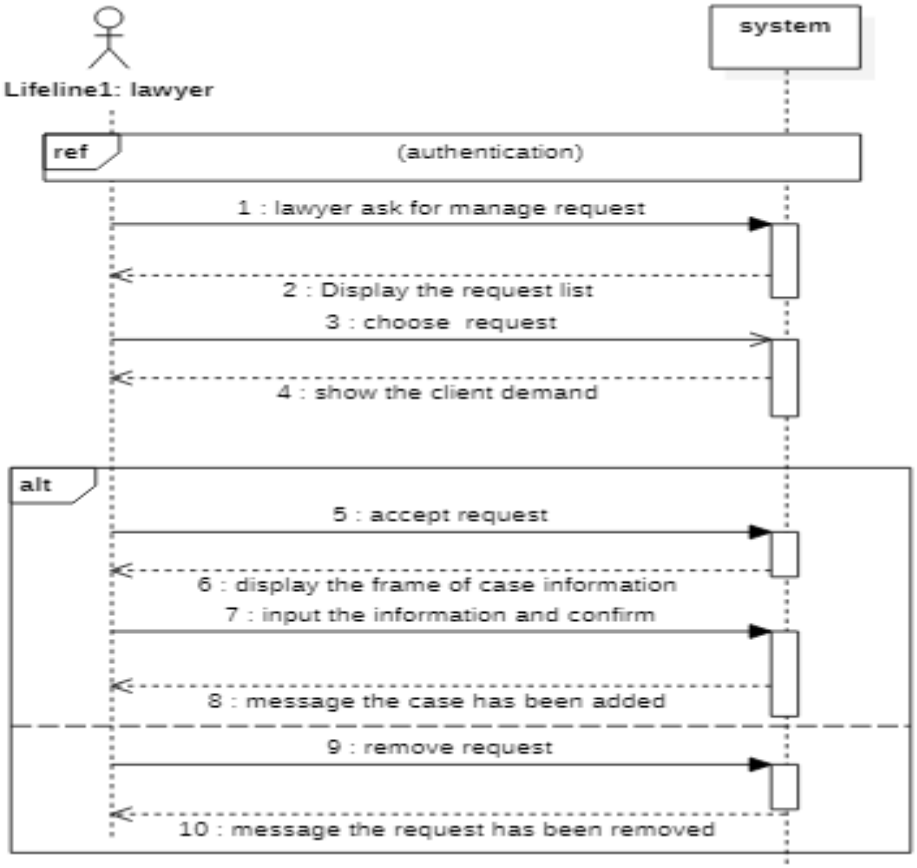

Figure 2.5 Managing requests sequence diagram

- **Managing Clients use case** (As shown in Figure 2.6)

The goal here the lawyer(admin) can check all the user of the system from this use case

the action

- remove user from system
- check client profile

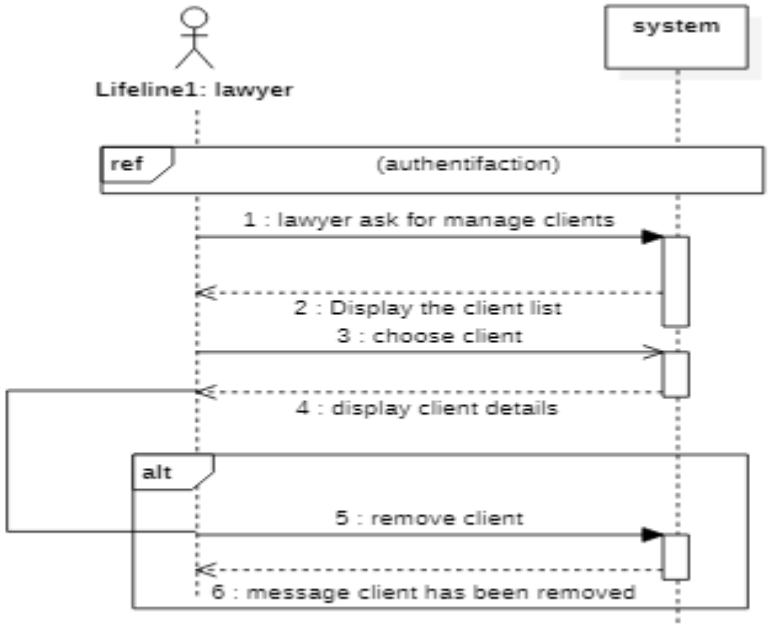

Figure 2.6 Managing clients sequence diagram

- **Managing cases use case**

The goal here is to manage the activities that belongs to each case

the action

- Send session date
- Archive case
- Edit case
- Send invoice
- History

❖ **Managing case activity diagram** (As shown in Figure 2.7)

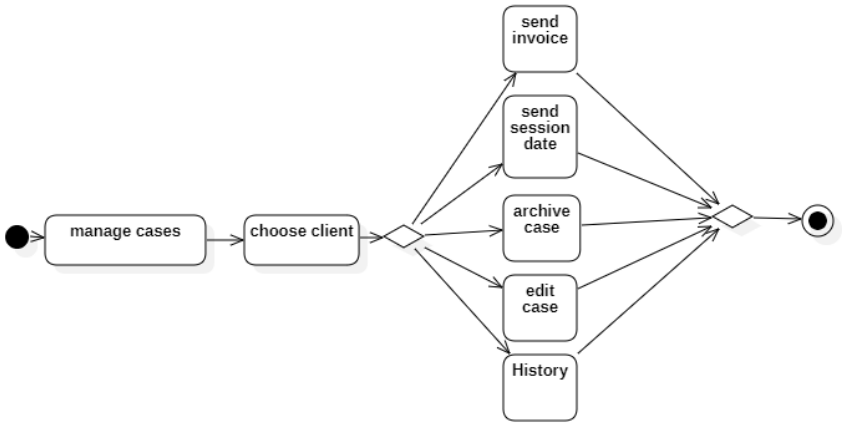

Figure 2.7 Managing cases activity diagram

- **Sending session date** (As shown in Figure 2.8)

The goal here that the lawyer can send session time from this action

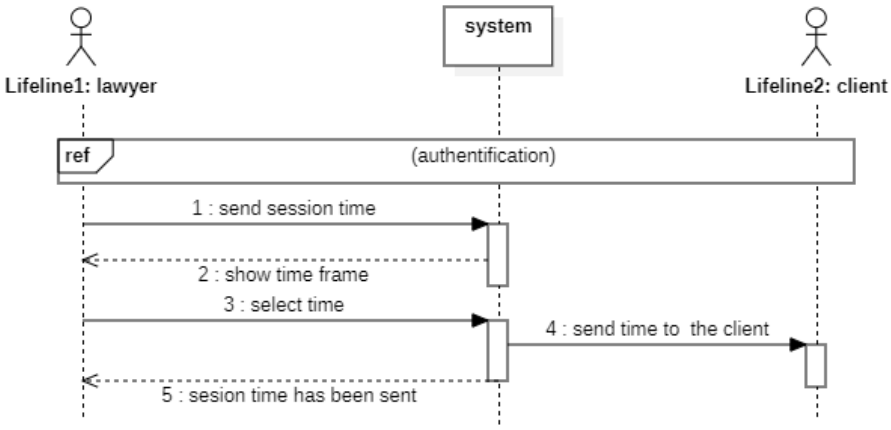

Figure 2.8 Sending session date sequence diagram

- **Archiving case** (As shown in Figure 2.9)

The lawyer can add a case to the archive

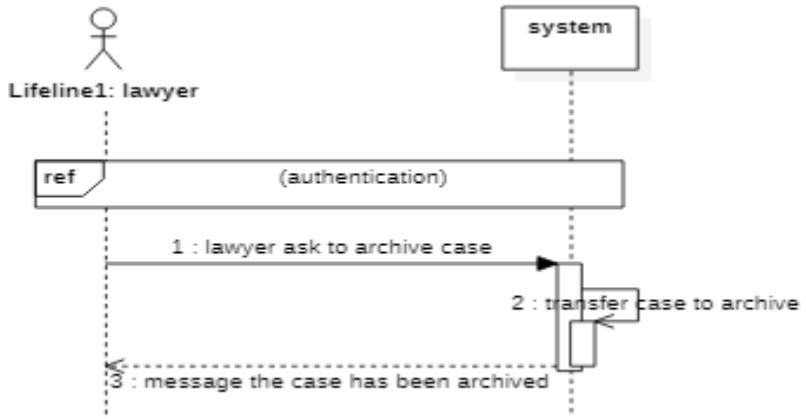

Figure 2.9 Archiving case sequence diagram

- **Editing case** (As shown in Figure 2.10)

The lawyer can edit on the case information

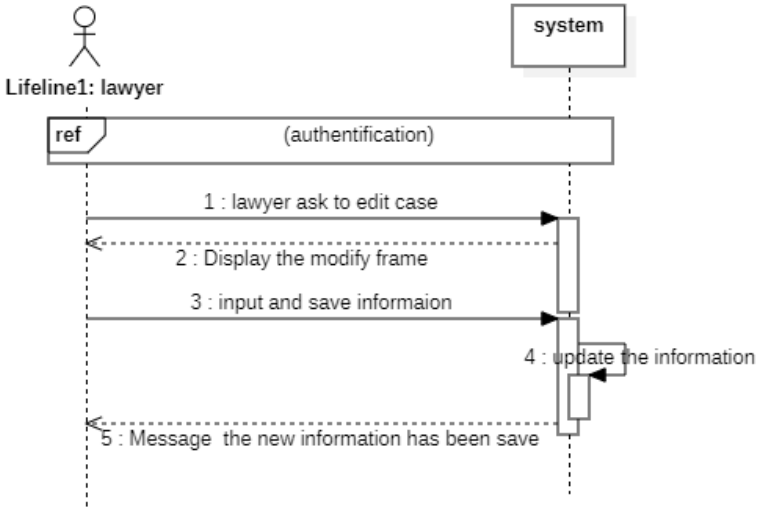

Figure 2.10 Editing case sequence diagram

- **Sending invoice** (As shown in Figure 2.11)

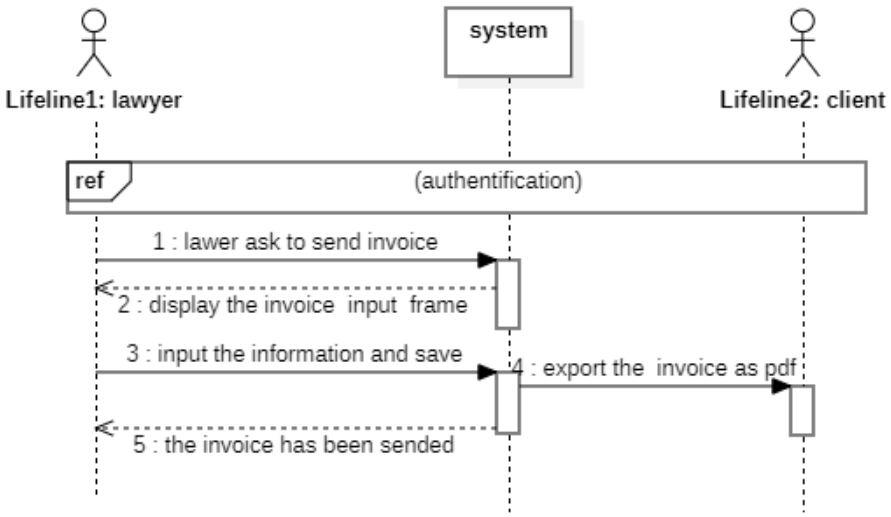

Figure 2.11 Sending invoice sequence diagram

- **Checking history** (As shown in Figure 2.12)

The lawyer can consult all the data (session time, notes) that he sends to the client

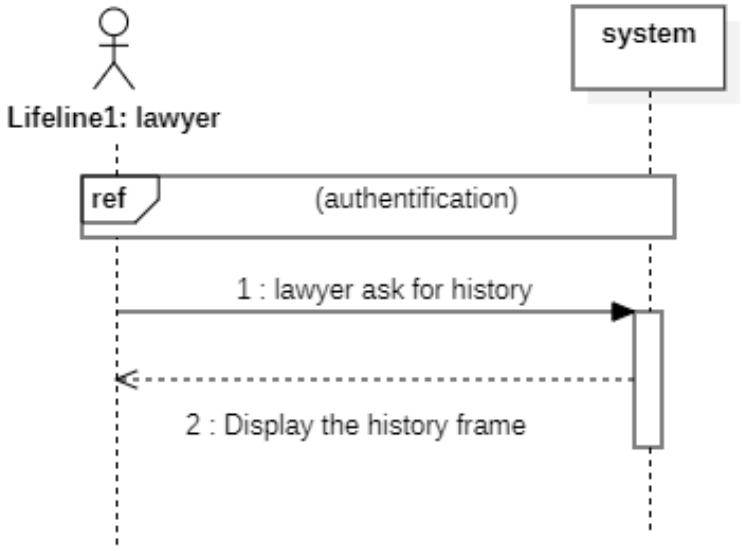

Figure 2.12 Checking history sequence diagram

- **Checking Archive use case** (As shown in Figure 2.13)

The goal here the lawyer(admin) can check all the previous cases

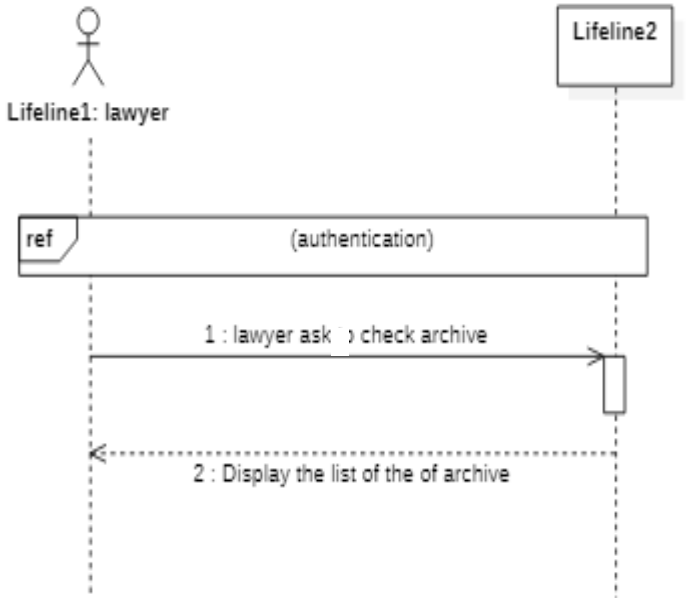

Figure 2.13 Checking Archive sequence diagram

- **Searching** (As shown in Figure 2.14)

The goal here is to help the lawyer to find clients and cases

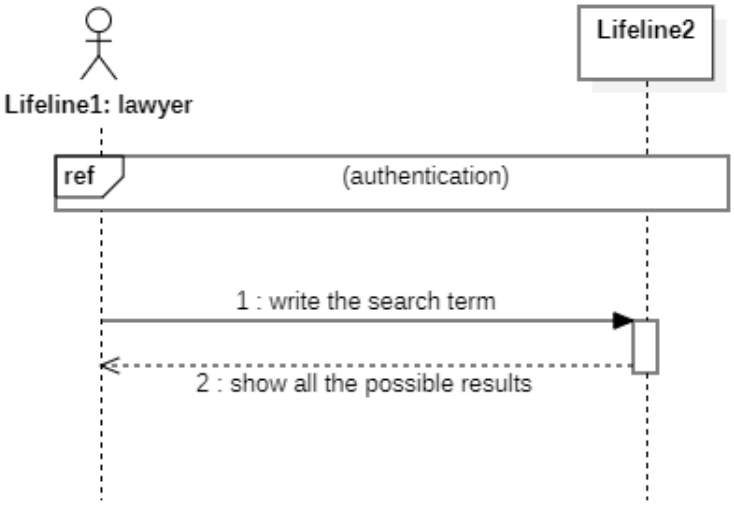

Figure 2.14 Searching sequence diagram

- **Creating proxy** (As shown in Figure 2.15)
  The goal here that the client can create a proxy as a pdf file
  Action
  Convert the entered information to a pdf file

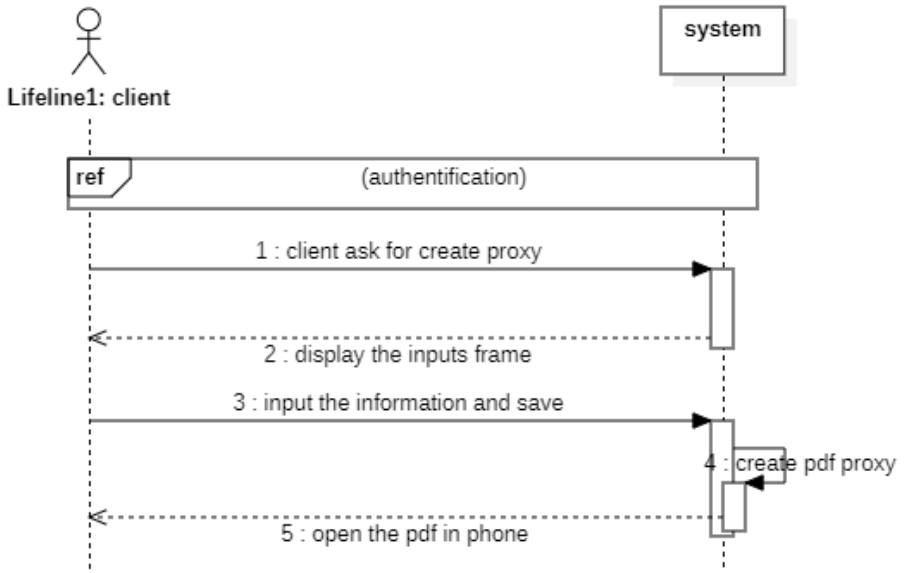

Figure 2.15 Creating proxy sequence diagram

- **Checking news** (As shown in Figure 2.16)
  The goal here is the client can check the news that the lawyer sent it

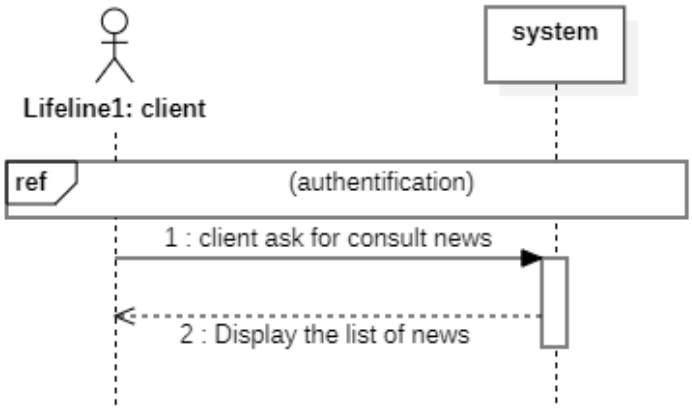

Figure 2.16 Checking news sequence diagram

- **Consulting invoices** (As shown in Figure 2.17)
  The goal here is the client can consult the invoices

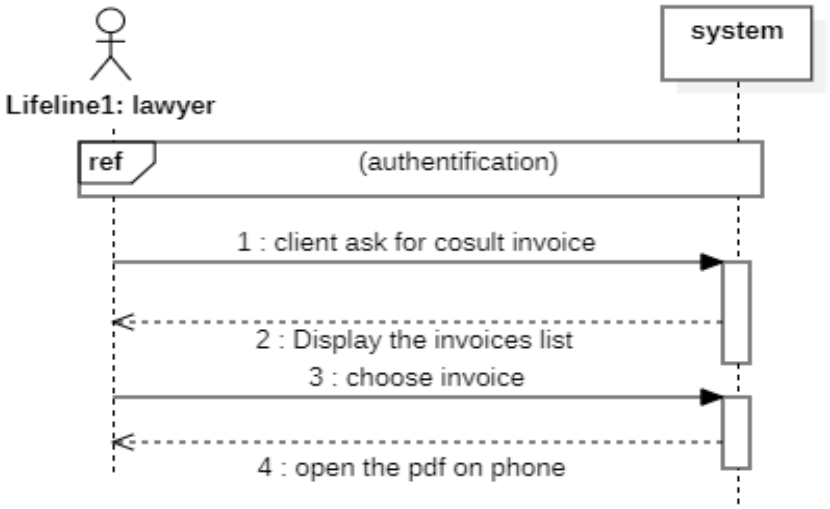

Figure 2.17 Consulting invoices sequence diagram

- **Checking request replies use case** (As shown in Figure 2.18)

The goal here the Client can check all his requests replies

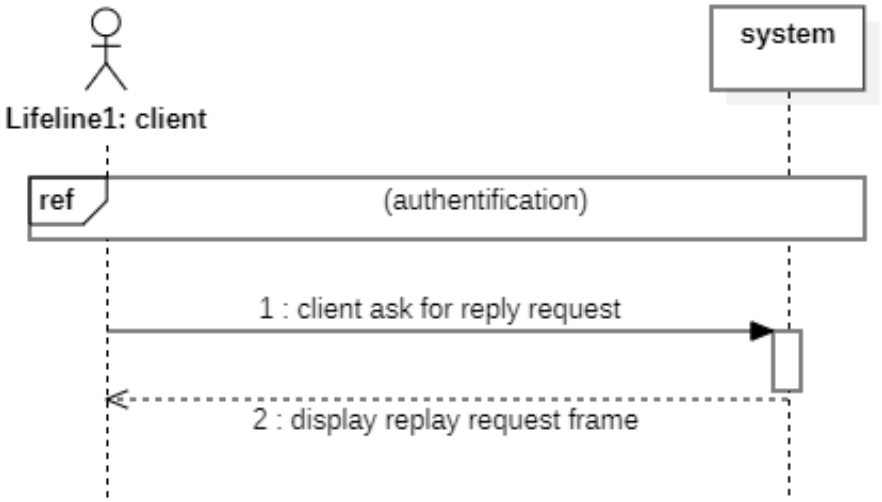

Figure 2.18 Checking request replies sequence diagram

- **Sending request use case** (As shown in Figure 2.19)

The goal here the Client can send a request to the lawyer

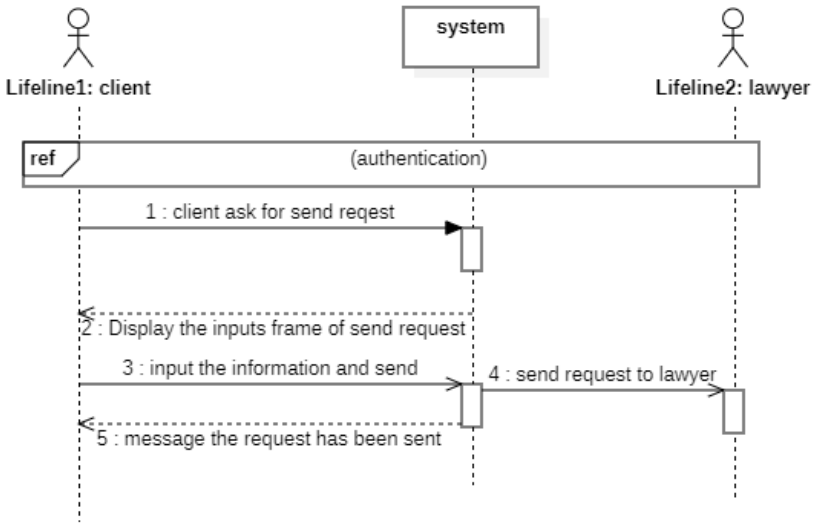

Figure 2.19 Sending request sequence diagram

2-4-The General Class diagram

This step will allow us to illustrate the main constructions of the class diagram. Class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes and the relationships among objects **[6],** as explained in the (Figure 2.20)

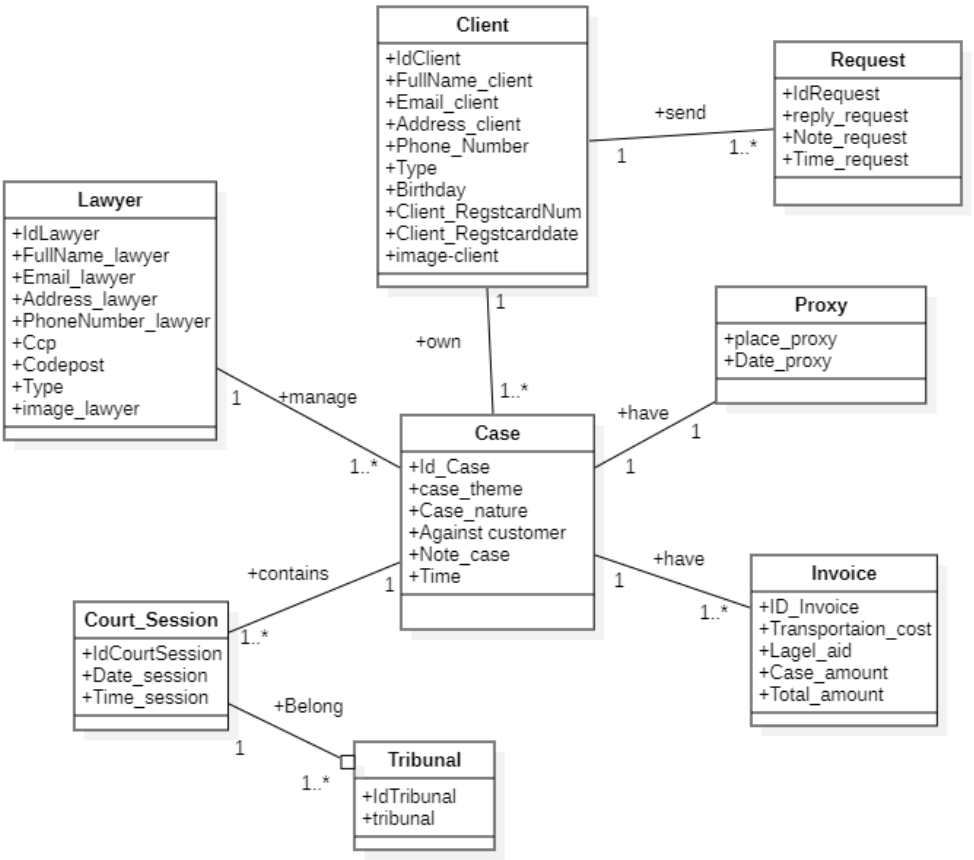

Figure 2.20 The General class diagram

- **data dictionary:**(As shown in Table 2.2)

Class	Attribute	Description	Type
Lawyer	IdLawyer FullName_lawyer Email_lawyer Address_lawyer Phone_Number_lawyer Ccp Codepost Type Image_lawyer	Lawyer Identificatory Lawyer full name Lawyer Email Lawyer Address Lawyer Phone number Lawyer ccp account Lawyer Code postal Account type Lawyer image	varchar(6) varchar(50) varchar(30) varchar(30) int int int varchar(1) image
Client	IdClient FullName_client Email_client Address_client Phone_Number_client Type Birthday Image_client Client_RegstcardNum Client_Regstcarddate	Client Identificatory Client full name Client email Client address Client phone number Account type Client birthdate Client image Client registration card Number Client registration card Creation date	varchar(6) varchar(30) varchar(30) varchar(30) int varchar(1) int image int int
Case	IdCase Case_Theme Case_Nature Against customer Notes_case Time	Case Identificatory Case Theme Case Nature Client Against customer Case Notes Case Time of Creation	varchar(6) varchar(30) varchar(30) varchar(30) varchar(50) time
Proxy	Place_proxy Date_proxy	Proxy redaction place Proxy redaction date	varchar(30) date
Request	IdRequest Notes_request Time_request Reply_request	Request Identificatory Request notes Creation time of the request The reply of the client request	varchar(6) varchar(30) time varchar(30)

Tribunal	IdTribunal tribunal	tribunal Identificatory tribunal name	varchar(6) varchar(30)
Court_session	IdCourtSession Date_session Time_session	Courtsession Identificatory Court session date Court session time	varchar(6) date time
Invoice	Id_Invoice Case_Amount Transportation_cost Legal_Aid Total_amont	invoice Identificatory Case amount transportation cost legal advice cost Total amount	varchar(6) int int int int

Table 2.2 The class diagram data dictionary.

2-5-Conclusion

In this chapter, we covered the analysis stage, which allowed us to go from functional structure via use cases to object structuring via classes and categories. We have, on the other hand, described the system's several classes.

Meanwhile, the next chapter will focus on the application's development.

Chapter 03: Realization

Chapter 03: Realization

3-1-Introduction

After making the appropriate design for our application, we will detail the process of creating our application in this chapter. This is done by describing the development environment, database implementation, and an overview of our application's interfaces.

3-2-Development Environment

3-2-1-Flutter SDK

Flutter is Google's portable UI toolkit for crafting beautiful, natively compiled applications for mobile, web, and desktop from a single codebase. Flutter works with existing code, is used by developers and organizations around the world, and is free and open source. flutter includes: Heavily optimized, mobile-first 2D rendering engine with excellent support for text. Modern react-style framework. Rich set of widgets implementing Material Design and iOS-style. APIs for unit and integration tests. Interop and plugin APIs to connect to the system and 3rd-party SDKs. Headless test runner for running tests on Windows, Linux, and Mac. Dart DevTools for testing, debugging, and profiling your app. Command-line tools for creating, building, testing, and compiling your apps.**[7]**

3-2-2- Dart Language

Dart is a client-optimized language for developing fast apps on any platform. Its goal is to offer the most productive programming language for multi-platform development, paired with a flexible execution runtime platform for app frameworks. Dart also forms the foundation of Flutter. Dart provides the language and runtimes that power Flutter apps, but Dart also supports many core developer tasks like formatting, analyzing, and testing code.**[7]**

3-2-3-Backend (Firebase)

Firebase is a Backend-as-a-Service (Baas). It provides developers with a variety of tools and services to help as develop quality apps, grow their user base, and earn profit. It is built on Google's infrastructure. **[8]**

- **Authentication**

It supports authentication using email passwords, phone numbers, Google, Facebook, Twitter, and more. The Firebase Authentication can be used to manually integrate one or more sign-in methods into an app.

- **Firestore (NoSQL)**

Cloud Firestore is a flexible, scalable database for mobile, web, and server development from Firebase and Google Cloud. Like Firebase Realtime Database, it keeps your data in sync across client apps through real time listeners and offers offline support for mobile and web so you can build responsive apps that work regardless of network latency or Internet connectivity. Cloud Firestore also offers seamless integration with other Firebase and Google Cloud products, including Cloud Functions.

- **Storage**

Cloud Storage for Firebase is built for application developers who need to store and serve user-generated content, such as photos or videos. Cloud Storage for Firebase is a powerful, simple, and cost-effective object storage service built for Google scale. The Firebase SDKs for Cloud Storage add Google security to file uploads and downloads for your Firebase apps, regardless of network quality.

3-2-4-VScode text editor

VScode is the fastest editor for flutter users. Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It is created and maintained by Microsoft Corporation. It has a rich set of plugins and extensions to speed up development. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to full-featured IDEs. **[9]**

3-3- Project interfaces presentation

In this part of the report, we will present the main functionalities of our application which will be done by the presentation of some interfaces.

- **"authentication" user interface**

At the first launch of the application, the first window displayed is the window "login" that represents the authentication (As shown in Figure 3.1).

And if it is the first time that he uses the application he should create a new account by heading to "Sign-in" page (As shown in Figure 3.2). and compete the process.

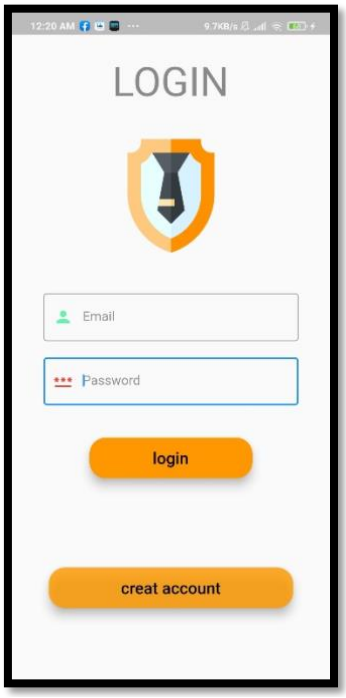

Figure 3.1 Log-in user interface

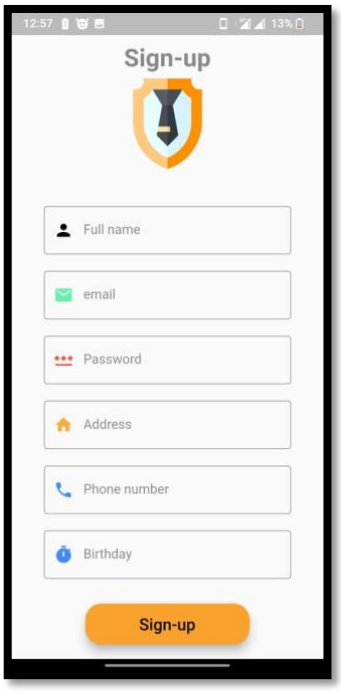

Figure 3.2 Sign-up user interface

- **“home” user interface**

We have two different home pages for both lawyer (As shown in Figure 3.3). And client (As shown in Figure 3.4).

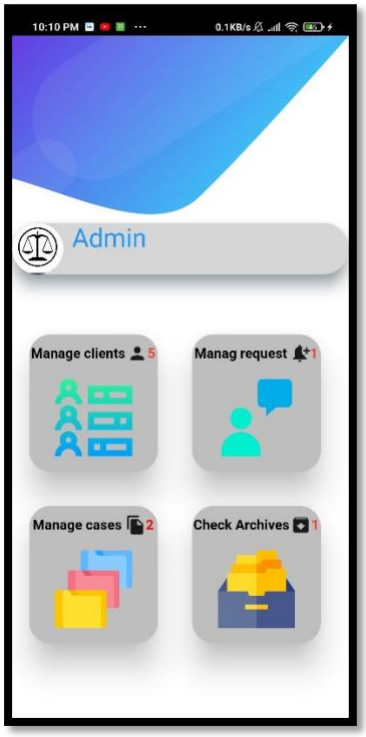

Figure 3.3 Lawyer home user interface

Figure 3.4 Client home user interface

- **Different set of user interfaces**

We present the “Manage clients” page (As shown in Figure 3.5). Then “manage cases” page (As shown in Figure 3.6). And after “Manage requests” page (As shown in Figure 3.7). And lastly “Profile” page (As shown in Figure 3.8).

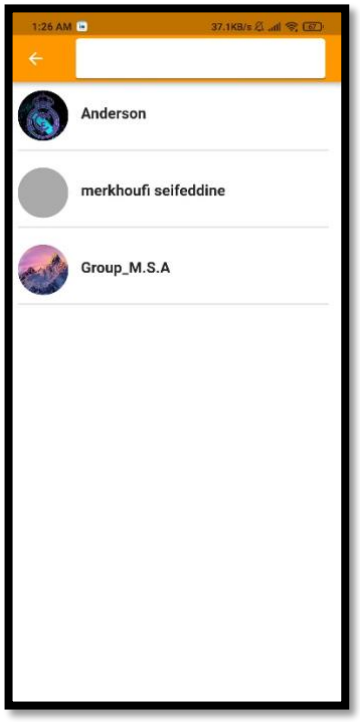

Figure 3.5 Manage clients user interface

Figure 3.6 Manage cases user interface

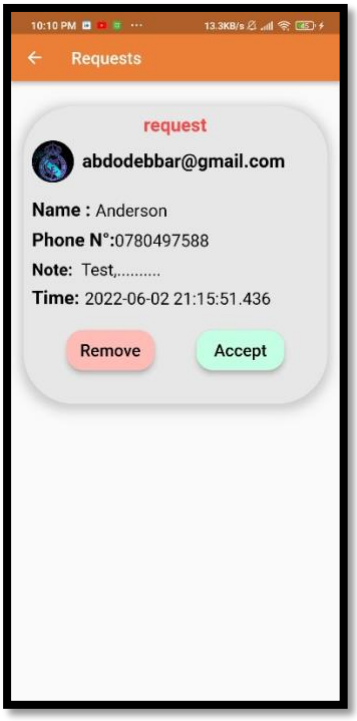

Figure 3.7 Manage requests user interface

Figure 3.8 Profile user interface

3-4- Conclusion

The realization phase is the most delicate stage in the development of our application. In this chapter, we covered the practical parts of completing our project, such as the development tools required. Finally, we've shown some of the user interfaces available in our application.

General Conclusion

The purpose of this report was to build and implement a mobile app for law firm management.

We began our job by introducing the lawyer firm, the issues, and our proposed solution. We progressed to the software design process in the second chapter by attempting to collect user needs and finishing with a detailed design of our product. The third chapter covered implementation specifics as well as the display of key user interfaces.

Perspectives:

Our report could be improved in various ways:

- Easy access to your documents and scans.
- Easy access for both lawyers and clients and make it an open application depends on supply and demand.
- Giving monthly statistics on the number of cases, Amounts of money and debts
- Managing appointemnts with clients

References

[1] https ://www.renson-avocats.be/role-et-les-missions-avocat.html

[2] http://dspace.univ-tlemcen.dz/bitstream/112/5500/5/chapitre1.pdf

[3] pascal Roques et Franck Vallée. UML en action de l'analyse des besoins a la conception en java,2000.

[4] pascal Roques et Franck Vallée. UML en action De l'analyse des besoins à la conception 4éme édition.

[5] https://en.wikipedia.org/wiki/Use_case_diagram

[6] https://en.wikipedia.org/wiki/Class_diagram

[7] https://docs.flutter.dev/resources/faq

[8] https://firebase.google.com/docs

[9] https://code.visualstudio.com/docs/supporting/faq