people's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

M'Hamed Bougara University - Boumerdes



Faculty of science

Department of Computer Science

Domain: Mathematics and Computer Science

Field: Computer Science

Specialty: Information Systems

Final Year Project for the Award of Bachelor's Degree

Development of a Mobile Application for Client–Artisan Connection

Presented by:

Ms. Ikram Taftist

Ms. Meriem Chennouf

Ms. Nour El Houda Akhdari

Ms. Sofia Oubaziz

Supervisor: Ms. Samiya Hamadouche

Examiner : Mr. Nouasri Amine

Defended on : 26/05/2026

Academic Year: 2025/2026

Dedication

Dedicated by Meriem

بسم الله الرحمن الرحيم

Tout d'abord, je tiens à remercier Dieu de m'avoir donné la force et le courage d'accomplir ce modeste travail.

Je dédie ce travail :

À la petite fille que j'étais autrefois

Tu es aimée, tu es entendue, tu es vue. Je suis devenue ce refuge sûr que tu espérais, cette force tranquille que tu cherchais dans les tempêtes. Aujourd'hui, je te rends hommage avec tendresse et fierté, car je suis fière de la femme que je suis devenue, fière du chemin parcouru, des peurs affrontées et des rêves réalisés. Et surtout, je suis enfin heureuse.

Ce projet est pour toi

À ma très chère mère

Ce pilier de ma vie, cette femme au cœur immense dont l'amour inconditionnel et les prières silencieuses m'ont porté dans les moments les plus difficiles. Aucune dédicace, aucun mot, aucune phrase ne pourrait exprimer l'amour, la reconnaissance et le respect que j'ai pour toi. Tu es et tu resteras ma plus belle lumière.

À mon cher père

Cet homme de dignité et de sagesse, qui s'est sacrifié sans compter pour que je puisse atteindre mes rêves. Tu m'as appris que la persévérance est la clé de tout succès. Ta fierté est ma plus belle récompense et ton soutien mon plus grand trésor.

À ma chère sœur Bouchra

Ma confidente, ma douceur et ma complice de toujours. Merci pour ta présence, ton soutien sincère et ton amour fraternel qui m'ont accompagné tout au long de ce chemin.

À mon cher frère Azzedine

Mon ami, mon pilier et mon confident. Merci pour chaque mot d'encouragement, chaque geste de soutien et chaque moment partagé qui ont rendu ce voyage plus léger.

À mes chers amis

Vous qui avez partagé avec moi les rires et les larmes, les doutes et les victoires. Votre amitié sincère a été un véritable cadeau sur ce parcours. Merci d'avoir toujours cru en moi.

Dedication

Dedicated by Ikram

بِسْمِ اللهِ الرَّحْمٰنِ الرَّحِيم

الحمدُ لله الذي بنعمته تتمّ الصالحات، أحمده سبحانه حمدًا كثيرًا طيبًا مباركًا فيه، فقد منَّ عليَّ بإتمام مشوارٍ جامعيٍّ جديد تُوِّج بنيل شهادة الليسانس، فله الحمد أولًا وآخرًا على ما وفّق ويسّر. وأسأله عزّ وجلّ أن يرزقني علمًا نافعًا أُفيد به غيري، وأن يجعل ما تعلمناه وعملنا به في ميزان حسناتنا.

أتقدّم بخالص الشكر والامتنان إلى كلّ من كان سندًا لي وداعمًا خلال هذه السنوات الثلاث، وعلى رأسهم والداي العزيزان، اللذان كانا نعم العون والسند.

شكراً أبي، شكراً أمي، على كلّ ما بذلتماه من تعبٍ وصبرٍ ودعاء، وعلى دعمكما المتواصل الذي كان يدفعني للاستمرار وتجاوز كلّ الصعوبات، فجزاكما الله عنّي خير الجزاء وبارك في عمركما.

كما أتوجّه بالشكر إلى إخوتي وأختي الأعزاء، الذين كانوا مصدرًا للطمأنينة والابتسامة رغم التعب والمشقة، وإلى أخي الصغير الذي كان خير أنيسٍ ورفيقٍ لي في طريق العودة، دون كللٍ أو تذمّر، أسأل الله أن يرزقه من واسع فضله ويوفّقه لما يحب ويرضى.

ولا يفوتني أن أتقدّم بجزيل الشكر إلى أساتذتي الكرام، وكلّ من علّمني حرفًا أو قدّم لي علمًا أو نصيحة. كما أخصّ بالشكر الأساتذة المتفهّمين أصحاب الأخلاق الرفيعة، الذين جعلوا من طلب العلم متعةً وشغفًا لا ينطفئ.

وأشكر صديقاتي ورفيقات دربي، اللواتي كنّ خير سندٍ ورفقة طوال سنوات الدراسة، فقد خفّفن عنّي ضغوط الأيام الصعبة، وشاركني لحظات التعب والنجاح، فكانت الجامعة أجمل بوجودهن. أسأل الله أن يحفظكن ويوفّقكن في مستقبلكن العلمي والمهني.

وفي الختام، نحمد الله حمدًا كثيرًا على إتمام هذا المشروع الذي قدّمناه أنا وزميلاتي بكلّ إخلاصٍ واجتهاد، ووضعنا فيه جزءًا من جهدنا ووقتنا وطموحنا. نسأل اللهم أن تجعله علمًا نافعًا وسببًا للخير والمنفعة، وأن تبارك لنا فيما تعلمنا، وتزيدنا علمًا وتوفيقًا، إنّك وليّ ذلك والقادر عليه.

Dedication

Dedicated by Sofia

Je dédie ce travail de fin d'études à toutes les personnes qui ont occupé une place importante dans ma vie et qui m'ont accompagnée, de près ou de loin, durant ce long parcours rempli d'efforts, de doutes, de sacrifices et d'espoir.

À ma famille, pour votre amour inconditionnel, votre patience, vos encouragements et votre présence dans les moments les plus difficiles. Merci d'avoir toujours cru en moi même lorsque moi-même je doutais. Votre soutien a été ma plus grande force. Une pensée toute particulière à mon frère **Idir** et ma sœur **Katia**, pour votre présence, vos conseils, votre soutien moral et tous ces petits gestes qui ont compté plus que vous ne l'imaginez. Vous avez été une source de motivation et de réconfort tout au long de cette aventure. Savoir que je pouvais compter sur vous m'a aidée à avancer malgré la fatigue, le stress et les périodes compliquées.

À mes amis(es), merci pour les moments de rire, les discussions, l'entraide, les encouragements et la présence sincère durant les périodes de pression. Vous avez rendu ce parcours plus léger et plus beau.

À mon groupe de PFE, merci pour les efforts partagés, la patience, les idées, les difficultés traversées ensemble et tous les moments vécus durant ce projet. Ce travail représente aussi une aventure humaine faite de collaboration, de persévérance et d'apprentissage.

À mes grands-parents, paix à leurs âmes, même si vous n'êtes plus parmi nous aujourd'hui, votre amour, vos valeurs et vos prières continuent de m'accompagner chaque jour. J'aurais aimé que vous soyez présents pour voir ce moment si important de ma vie. Cette réussite vous est dédiée avec beaucoup d'amour et d'émotion.

Et enfin… je me dédie aussi ce travail **à moi-même**. À cette version de moi qui a continué malgré le stress, les nuits difficiles, la pression, les moments de découragement et la peur de ne pas y arriver. À toutes les fois où j'ai dû être forte, persévérer et me relever seule. Le chemin n'a pas été facile, mais je suis restée debout jusqu'au bout.

Aujourd'hui, je suis fière de moi, fière du chemin parcouru, fière de ne pas avoir abandonné malgré toutes les difficultés. Ce mémoire représente bien plus qu'un projet académique : il représente une preuve de ma détermination, de ma patience et de ma capacité à avancer même dans les moments compliqués.

Que ce travail soit le début d'un avenir rempli de réussite, de bonheur et de nouvelles ambitions.

Dedication

Dedicated by Nour

To my parents, thank you for your unwavering presence throughout my journey. I am deeply grateful for all the sacrifices you have made for me.

To my grandparents, whose home was always open to me and became the place where I spent much of my life, thank you for raising me with love, care, and guidance.

Particulièrement à ma grand-mère, qui n'a pas eu la chance de terminer ses études, mais qui a toujours profondément cru en la valeur de l'éducation et a encouragé toutes les femmes autour d'elle, y compris moi, à poursuivre leurs objectifs. J'ai choisi d'écrire cette partie en français afin qu'elle puisse la comprendre. Merci pour ta confiance constante en moi, ton soutien, ainsi que pour la force et la gentillesse qui continuent de m'inspirer.

To my grandfather on my father's side, although we were not in close contact throughout much of my life, I still carry your memory with me and hold it close to my heart.

To my aunts, thank you for your care, encouragement, and for always making me feel supported throughout the years. You have been like older sisters to me in so many ways.

And especially to my aunt Isma, whose kindness, support, and generosity have meant more to me than words can express. Thank you for always being there for me, believing in me, and caring for me.

And to someone very special to me, thank you for your constant support, encouragement, and for being my companion throughout this journey. Your belief in me, your love, and your companionship have meant more than you know.

This work is dedicated to all of you.

Dedication

To our supervisor, Madame Hamadouche Samya

We would like to express our sincere gratitude and sincere appreciation for your valuable guidance, patience, and continuous support throughout this project.

Your scientific rigor, insightful advice, and constant availability have been a great source of motivation and have significantly contributed to the success of this work.

Working under your supervision has been both an honor and a privilege. We are truly grateful for your trust, encouragement, and dedication.

Thank you for everything you have done for us.

Abstract

This project involves the design and development of a mobile application called Allo Artisan, aimed at connecting clients with artisans in Algeria. Following an analysis of existing solutions and their limitations, we proposed a solution built around three main axes: trust, functionality, and accessibility.

The application provides key features such as QR code identity verification, diploma validation, a multi-criteria search system, an availability calendar, and a post and notification mechanism for service requests.

Technically, the application was developed using Flutter for the mobile interface, Go (Golang) for the backend, and PostgreSQL as the database management system, deployed via Docker.

Keywords: Mobile application, client-artisan connection, Flutter, Go, PostgreSQL.

يتناول هذا المشروع تصميم وتطوير تطبيق هاتف محمول يُدعى «Allo Artisan»، يهدف إلى تسهيل عملية الربط بين الزبائن والحرفيين في الجزائر.

بعد دراسة وتحليل التطبيقات الموجودة في نفس المجال والتعرّف على حدودها ونقاط ضعفها، قمنا باقتراح حل يعتمد على ثلاثة محاور أساسية: الثقة، الوظائف، وسهولة الاستخدام.

يوفّر التطبيق مجموعة من الخصائص الأساسية، من بينها: التحقق من هوية الحرفي باستخدام رمز «QR»، التحقق من الشهادات والوثائق المهنية، نظام بحث متعدد المعايير، تقويم لتحديد فترات التوفر، بالإضافة إلى نظام للمنشورات والإشعارات الخاصة بطلبات الخدمات العادية والمستعجلة.

من الناحية التقنية، تم تطوير التطبيق باستخدام «Flutter» لتطوير واجهة الهاتف المحمول، و« (Golang) Go» لتطوير الجزء الخلفي وإنشاء واجهة «API REST»، إضافةً إلى «PostgreSQL» كنظام لإدارة قواعد البيانات، مع استخدام «Docker» لضمان نشر مستقر وقابل للنقل.

الكلمات المفتاحية: تطبيق هاتف محمول، الربط بين الزبائن والحرفيين، «Flutter»، «Go»، «PostgreSQL».

Contents

Abstract vi

List of Tables ix

List of Figures x

General Introduction xi

1 Existing System Analysis 1

1.1 Introduction . . . 1
1.2 Review of Existing Solutions and Applications . . . 1
1.3 Problem Statement . . . 3
1.4 Proposed an Adopted Solution . . . 3
1.5 Conclusion . . . 5

2 Application Modeling 6

2.1 **Introduction** . . . 6
2.2 Requirements expression . . . 6
2.2.1 Definition of Use Case Diagram . . . 6
2.2.2 Identification of the actors . . . 6
2.2.3 Identification of the use case . . . 7
2.2.4 Presentation of Use case diagram . . . 7
2.3 Analyse . . . 9
2.3.1 Definition of the Sequence Diagram . . . 9
2.3.2 Examples of the Sequence Diagram and their Textual Descriptions . . . 9
2.3.3 Definition of the Activity Diagram . . . 14
2.3.4 Appointment Management Activity Diagram and its Textual Description . . . 14
2.4 Design . . . 15
2.4.1 Definition of the Class Diagram . . . 15
2.4.2 Data Dictionary . . . 15
2.4.3 The Management Rules . . . 16
2.4.4 Presentation of the Class Diagram . . . 18

2.4.5 From the Class Diagram to the Relational Model 18
2.5 Conclusion 19

3 Implementation **20**
3.1 Introduction 20
3.2 Working Environment 20
3.2.1 Language and Framework 20
3.2.2 Development Tools 21
3.3 Deployment Diagram 21
3.3.1 Definition of the Deployment Diagram 21
3.3.2 Presentation of Deployment Diagram and its Textual Description 22
3.4 Logo of the application 24
3.5 Application Interfaces 25
3.6 Conclusion 30

General Conclusion **xiii**

Bibliography **xiv**

Webography **xv**

List of Tables

2.1 Use cases by actor 7
2.2 Textual Description of the Use Case «Register an Artisan Account» 10
2.3 Textual Description of the Use Case «Authentication» 11
2.4 Textual description of the use case ”Schedule an appointment” 12
2.5 Textual Description of the Use Case« Approve Artisan Account» 13
2.6 Data dictionary 16

3.1 Communication Flows and Protocols 23

List of Figures

1.1 A service-based application 5

2.1 Use Case Diagram 8
2.2 Sequence diagram of the «register an artisan account» use case 9
2.3 Sequence diagram of ‘Authentication‘ use case 10
2.4 Sequence diagram”Schedule an appointment” use case 11
2.5 Sequence diagram ”Approve artisan account” use case 13
2.6 Appointment Management Activity Diagram 14
2.7 Class Diagram 18

3.1 Logo of Go language 20
3.2 Logo of Flutter 20
3.3 Logo of Visual studio 21
3.4 Logo of Android Studio 21
3.5 Logo of docker 21
3.6 Logo of PostgreSQL 21
3.7 Deployment Diagram 22
3.8 Logo of the application 24
3.9 Login Interface 25
3.10 Create Artisan Account Interface 26
3.11 Client Home Page Interface 27
3.12 Normal Request 28
3.13 Urgent Request 28
3.14 Service Request Sharing Interface 28
3.15 Search for an artisan Interface 28
3.16 Artisan Private Profile Interface 29
3.17 Artisan Public Profile Interface 30

General Introducion

In today's rapidly evolving digital landscape, mobile applications have become essential tools for simplifying everyday tasks and improving access to services. Among the most pressing challenges faced by Algerian households is the difficulty of finding reliable, qualified, and available artisans for urgent needs such as plumbing repairs, electrical installations, or mechanical interventions. Despite the existence of numerous skilled professionals in the market, the lack of an efficient and trustworthy platform to connect them with potential clients creates a significant gap in the local service ecosystem.

A review of existing solutions such as Khedamni, De9de9, and Afriha reveals recurring limitations: absence of identity verification, lack of availability management, restricted search options, and insufficient trust mechanisms. These shortcomings motivated us to propose a more robust and user-centered alternative **Allo Artisan** a mobile application designed to bridge the gap between clients and artisans through a secure, transparent, and intuitive platform built around three core principles: **trust, functionality, and accessibility**.

This document is structured into three chapters.

Chapter 1 presents a comparative analysis of existing applications, defines the problem statement, and describes our proposed solution strategy.

Chapter 2 covers the complete UML modeling of the system, including use case, sequence, class, and activity diagrams.

Chapter 3 details the implementation phase, presenting the tools and technologies used alongside the main interfaces of the application.

Chapter 1

Existing System Analysis

1.1 Introduction

This chapter is dedicated to the study and analysis of **existing solutions** in the same domain as our project. Reviewing existing applications represents a **fundamental step** in the development of any new system, as it provides a clear understanding of current market solutions and their functionalities. Through this analysis, we aim to examine how these applications operate, evaluate the **services they provide to users**, and identify their **strengths** as well as their **limitations**.

This study will enable us to determine the aspects that require improvement and to define the **requirements** necessary for designing a more **effective**, **reliable**, and **competitive application**.

1.2 Review of Existing Solutions and Applications

To better understand the current state of existing solutions, we will analyze a selection of popular applications in this domain, namely ***Khedamni***, ***De9de9***, and ***Afriha***. Each application will be examined based on its **main features**, **strengths**, and **weaknesses** in order **to highlight their limitations** and **identify opportunities for improvement**.

Khedamni

Khedamni is a mobile application available on the Google Play Store, with over 10,000 downloads and a rating of 4.4 stars based on 207 reviews.



Khedamni Logo

Weaknesses:

- Search limited to filtering by wilaya and commune only

- Many professionals have limited feedback, making it hard to assess service quality
- No diploma required during account creation, profiles lack professionalism
- No calendar feature, so users cannot check artisan availability
- Profiles display very little information overall

Strengths:

- Clean and professional interface
- Home page displays client posts filtered by wilaya, commune, and category
- Account creation is simple and straightforward
- Good use of post status indicators (e.g., “en attente”, replies)

De9de9

De9de9 is a mobile and web application available on the Google Play Store, with over 50,000 downloads and a rating of 4.1 stars based on 120 reviews.

De9de9 Logo

Weaknesses:

- No consistent standards in account creation; mix of professional and irrelevant profiles, which is unfair to skilled users
- No verification framework to validate identities or qualifications; no diploma required
- Guest access is very restricted, preventing visitors from building enough trust to register

Strengths:

- Richly detailed artisan profiles: questionnaire, work gallery, itemized pricing, availability calendar, and ratings
- Wide diversity of service categories with many subcategories

Afriha

Afriha is a mobile and web application available on the Google Play Store, with over 10,000 downloads and a rating of 4.0 stars based on 24 reviews.



Afriha Logo

Weaknesses:

- Mandatory GPS activation undermines trust and discourages new users
- Home page displays artisan accounts in a random and unclear order
- No diploma required, profile names appear unprofessional
- Search filtering limited to category only, which can raise safety concerns
- No calendar, profiles lack pricing and scheduling details
- Post functionality is very limited

Strengths:

- Search uses selection-based navigation instead of free-text input
- Wide variety of service categories with many associated services

1.3 Problem Statement

After conducting research by reading user reviews of existing applications, we collected their needs. Based on this, we identify the problem that should be addressed through our proposed application :
The identified shortcomings of current applications reveal a fundamental gap in addressing the needs of both end users and skilled artisans, highlighting the need for a more robust, scalable, and user-centered solution adapted to modern service ecosystems

1.4 Proposed an Adopted Solution

This section presents the main design directions of the proposed application, which are based on the analysis of existing systems and identified user needs. The proposed solution is structured around three main axes: **trust**, **functionality**, and **usability & accessibility**, which together define the core principles of our application.

1. Trust

- Each artisan is assigned a **QR code** linked to their profile, allowing clients to verify their identity before receiving any service.
- Artisan registration requires a **formal verification process**, including personal information and supporting documents such as diplomas or professional licenses. Accounts are reviewed before activation.
- A **reliable review system** is implemented, where ratings can only be submitted after service confirmation from both the client and the artisan, ensuring authenticity.
- A **progressive rating system** is used to reflect the artisan's most recent performance more accurately.
- The system includes a **Visitor Mode**, allowing users to browse the application without registration, improving accessibility and transparency.
- The application does not rely on GPS tracking; instead, users manually select their location (wilaya and commune), with an optional **distance-based search (km zone)** to find nearby artisans.

2. Features

- An **advanced search system** allows users to filter artisans by category and location (wilaya and commune), with an additional **distance-based filter (km zone)**.
- Artisans can switch between **Active and Inactive modes**, controlling their visibility and availability.
- A **post system** enables clients to publish requests (urgent or non-urgent), which are sent as notifications to active artisans in the same category and area. Artisans can also share their work and interact through comments and reactions.
- Each artisan profile includes a **calendar showing availability and booked days**, without exposing detailed appointment information to preserve privacy.

3. Usability & Accessibility

- The platform supports **three languages (Arabic, French, English)** to serve a wide range of users.
- Users can **follow and save artisans** for quick and easy access later.
- Only **active artisans** are displayed in search results to improve efficiency.

- The interface is designed to be **simple, clear, and intuitive**, suitable for users of all technical levels.

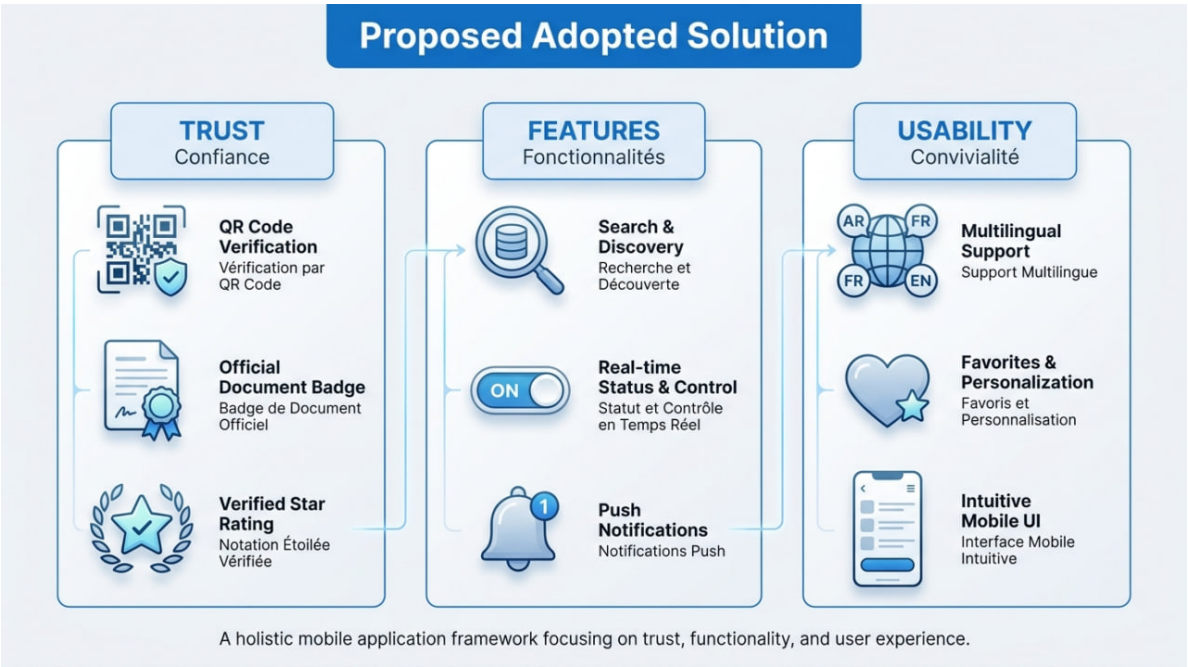


Figure 1.1: A service-based application

1.5 Conclusion

In this chapter, we conducted a **comprehensive study** of existing applications by analyzing their **strengths and weaknesses** from the users' perspective. This evaluation enabled us to identify **relevant improvements** and **innovative solutions** that will be integrated into our application in order to enhance its **performance**, **usability**, and **competitiveness** in the market.

In the next chapter, we will proceed to the implementation of the proposed solutions through the **System Modeling** phase using **UML diagrams**.

Chapter 2

Application Modeling

2.1 Introduction

In this chapter, we will study several UML (Unified Modeling Language) models of a system. Some models present the system from the users' point of view, others illustrate its internal structure, while some provide a global or detailed vision of the system. These models complement each other and can be combined. They are developed throughout the entire system development life cycle, from requirements expression to the design phase, **The application brings together five main actors**: the Visitor, the Client, the Artisan, the Moderator, and the Administrator

2.2 Requirements expression

2.2.1 Definition of Use Case Diagram

Use case diagram are usually referred to as behavior diagram used to describe a set of actions (use cases) that some system should or can perform in collaboration with one or more external users of the system (actors). Each use case should provide some observable and valuable result to the actors or other stakeholders of the system[1].

2.2.2 Identification of the actors

There are five actors who can use our application:

1. **Visitor:** an unauthenticated person who can access limited public features of the application.
2. **Client:** a registered user who can access the full feature of the application.
3. **Artisan:** a registered professional who offers services on the platform.
4. **Moderator:** a system manager responsible for controlling and supervising the application.

5. **Administrator:** a system manager responsible for adding or rejecting a moderator and to help the moderator in its tasks.

Each actor interacts with the system according to their role and permissions, which will be reflected in the use case diagram.

2.2.3 Identification of the use case

In this table 2.1, each actor has been defined with their own use cases:

Actor	Use Case
Visitor	View a post
Client	Create an account Interact with an artisan's post Search an artisan View artisan's profile Manage an appointment Check the artisan profile by QR code Rate an artisan after service Exchange message with artisan
Artisan	Register an account Process request View the notification Set the availability in a calendar Set status Publish a new post Exchange message with client
Moderator	Approve post Manage reports Manage accounts
Administrator	Manage moderator

Table 2.1: Use cases by actor

2.2.4 Presentation of Use case diagram

The following figure 2.1 illustrates the use case diagram of the application:

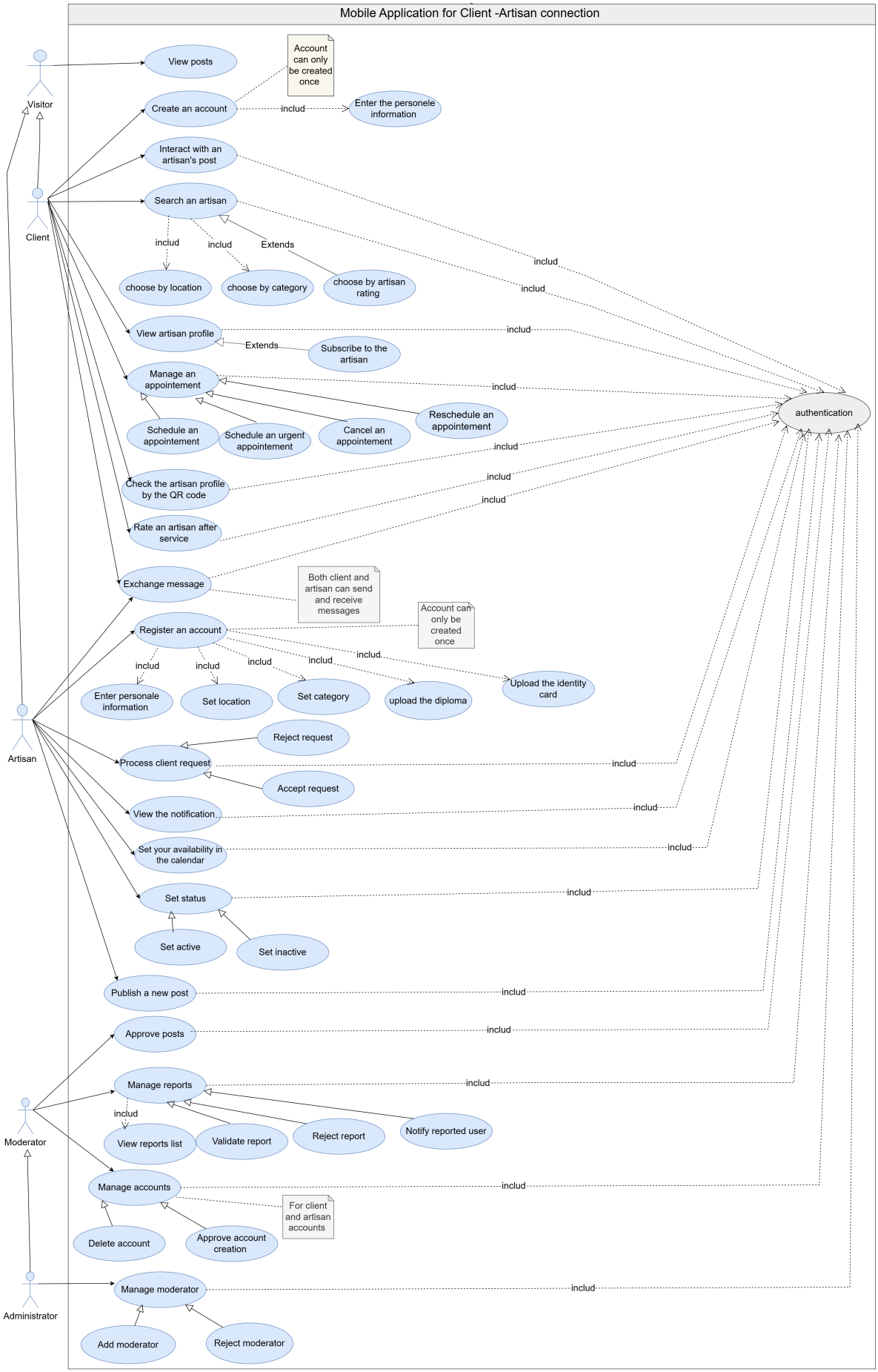


Figure 2.1: Use Case Diagram

2.3 Analyse

This point describes in detail the behavior and interactions between the different components, and for this purpose, we use:

2.3.1 Definition of the Sequence Diagram

A sequence diagram represent interactions between lifelines ,and illustrates interactions from a temporal perspective, particularly the chronological sequence of messages exchanged between lifelines[1]

2.3.2 Examples of the Sequence Diagram and their Textual Descriptions

In this part, we will present the sequence diagrams along with their textual descriptions for certain use cases of our system.

Use Case :'Register an Account Artisan'

The following figure(figure:2.2)represents the sequence diagram of the «Register an Artisan account» use case:

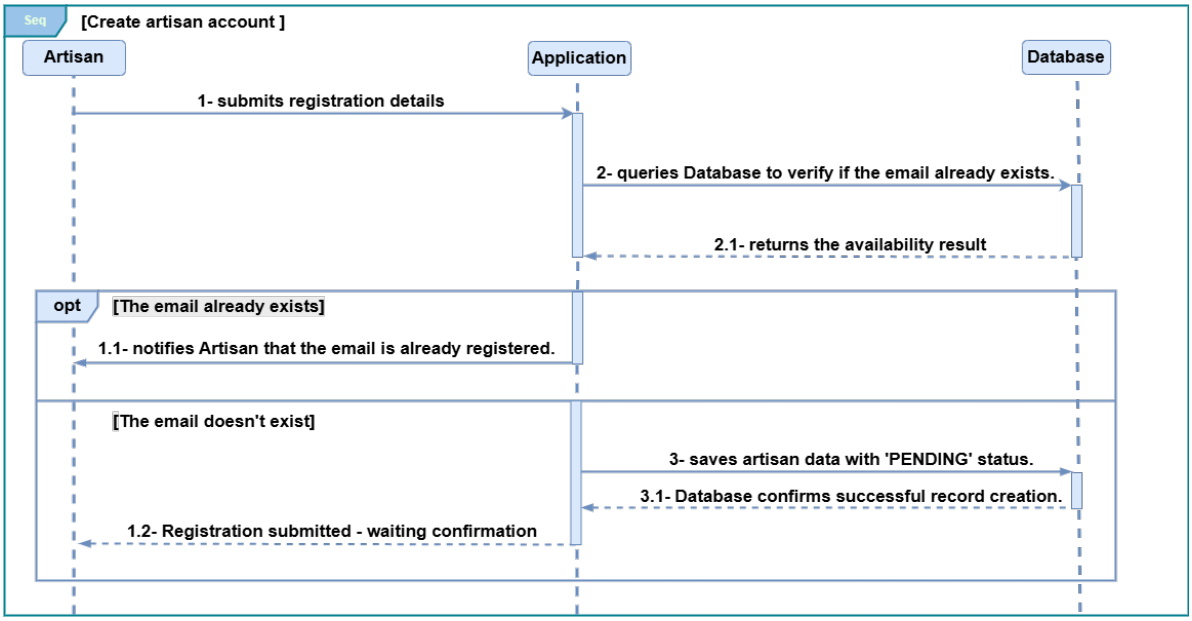


Figure 2.2: Sequence diagram of the «register an artisan account» use case

The following table (Table:2.2) represents the textual description corresponding to the previous sequence diagram(figure:2.2):

Register Artisan Account	
Title	Register Artisan Account
Actor	Artisan
Objective	Register a new artisan account in the system.
Precondition	The artisan must have a valid email address.
Nominal Scenario	1.The Artisan submits registration details. 2. The Application queries the Database to verify if the email already exists. 1.1. The Database returns that the email is available. 3. The Application saves the artisan data with ”Pending” status. 1.1. The Database confirms successful record creation. 1.2. The Application notifies the Artisan that the registration has been submitted and is awaiting confirmation.
Alternative Scenario	**A1: If the email already exists.** 1.1. The Application notifies the Artisan that the email is already registered.

Table 2.2: Textual Description of the Use Case «Register an Artisan Account»

Use Case :’authentication’

The following figure (figure:2.3) represents the sequence diagram of the “Authentication” use case

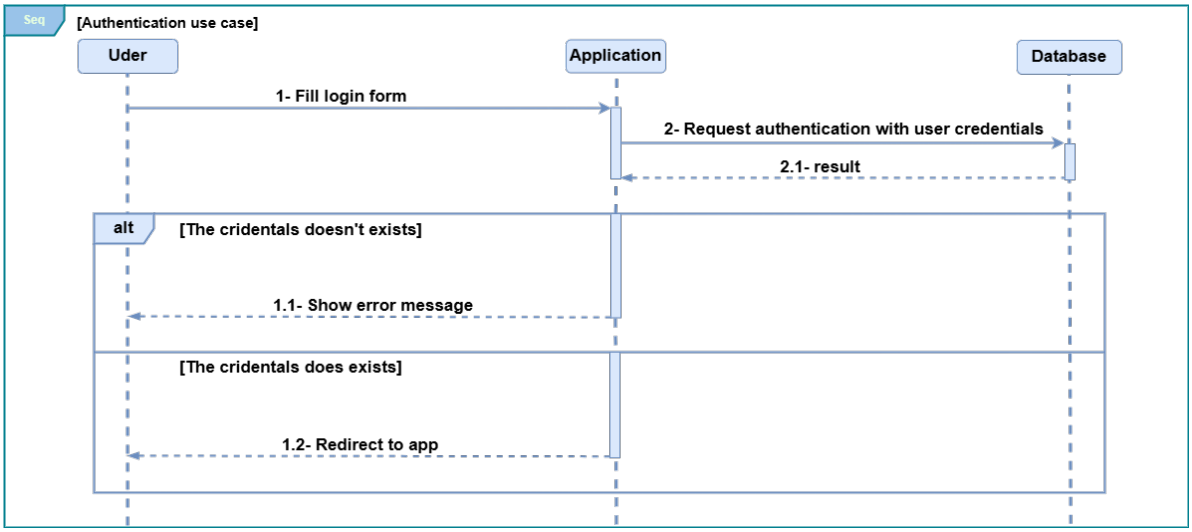


Figure 2.3: Sequence diagram of ‘Authentication‘ use case

The following table (Table:2.3) represents the textual description corresponding to the previous sequence diagram(figure:2.3):

User Authentication	
Title	User Authentication
Actor	User (Artisan or Admin)
Objective	Log into the system using credentials.
Precondition	The user has a previously created and validated account.
Nominal Scenario	1. The User fills in the login form. 2.The Application sends the credentials to the Database for verification. 2.1.The Database returns a successful result. 1.2.The Application redirects the User to the main application.
Alternative Scenario	**A1: Invalid credentials.** 1.1. The Application displays an error message.

Table 2.3: Textual Description of the Use Case «Authentication»

Use Case :'Schedule an appointment'

The following figure (Figure2.4) represents the sequence diagram of the«Schedule an appointment»use case

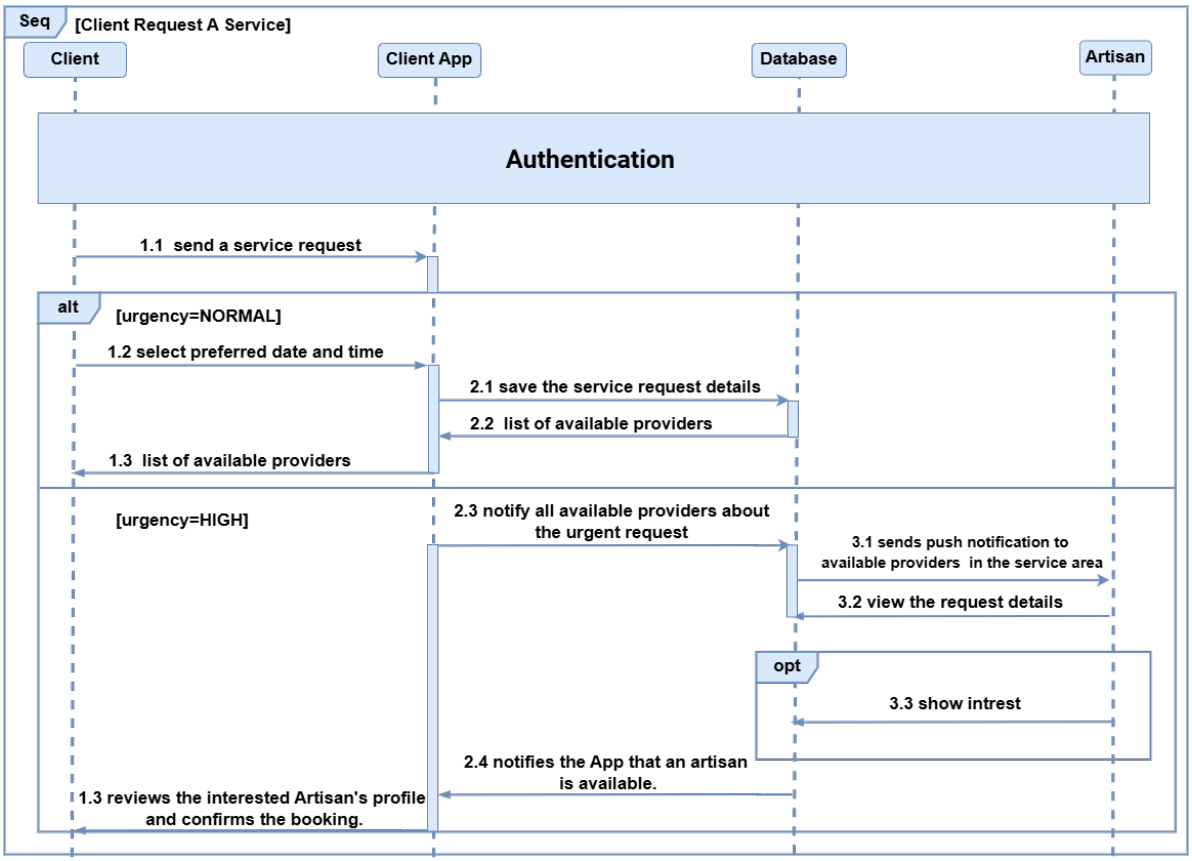


Figure 2.4: Sequence diagram"Schedule an appointment" use case

The following table (Table2.4) represents the textual description corresponding to the previous sequence diagram(Figure2.4):

Schedule an appointment	
Title	Schedule an appointment
Actor	Client
Objective	Allow a client to request a service and book an available artisan.
Precondition	• The Client is authenticated. • There are available artisans in the system.
Nominal Scenario	1.1. The Client sends a service request. 1.2. The Client selects preferred date and time. 2.1. The Client App saves the service request details in the Database. 2.2. The Database returns a list of available providers. 2.4. The Client App notifies the Client that an artisan is available. 1.3. The Client reviews the interested Artisan's profile and confirms the booking.
Alternative Scenario	**A1 (High Urgency):** 2.3. The Client App notifies all available providers about the urgent request. 3.1. The system sends push notification to available Artisans. 3.2. The Artisan views the request details. 3.3. The Artisan shows interest (optional). 1.3. The Client reviews the interested Artisan's profile and confirms the booking.

Table 2.4: Textual description of the use case "Schedule an appointment"

Use Case :approve artisan account creation

The following figure (figure:2.5) represents the sequence diagram of the «Approve artisan account creation» use case

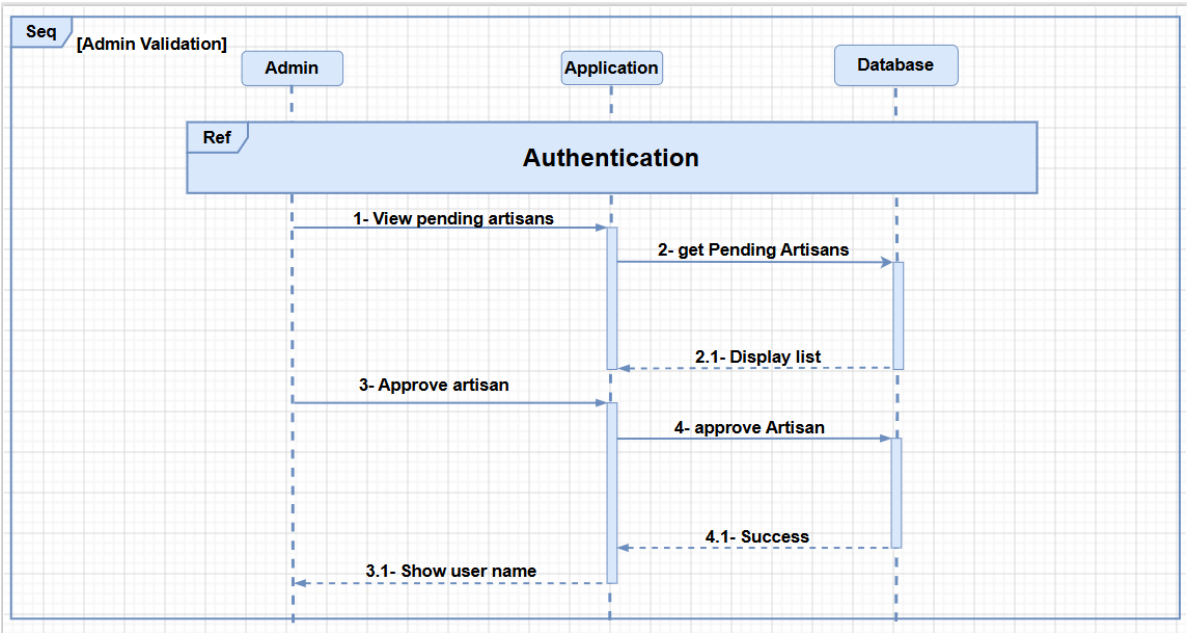


Figure 2.5: Sequence diagram ”Approve artisan account” use case

The following table (Table:2.5) represents the textual description corresponding to the previous sequence diagram(Figure2.5):

Approve Artisan Account	
Title	Approve Artisan Account
Actor	Admin
Objective	Approve a pending artisan account.
Precondition	• The Admin must be authenticated. • There are pending artisan accounts.
Nominal Scenario	1.The Admin views the list of pending artisans. 2.The Application retrieves pending artisans from the Database. 2.1 The Application displays the list. 3. The Admin selects and approves an artisan. 4.The Application sends the approval request to the Database. 4.1. The Database confirms success. 3.1. The Application shows a success message to the Admin.
Alternative Scenario	**A1: Error during approval (e.g., connection issue).** 3.2. The Application displays an error message.

Table 2.5: Textual Description of the Use Case« Approve Artisan Account»

2.3.3 Definition of the Activity Diagram

The activity diagram must represent all the actions performed by the system, including all conditional branches and possible loops. It is a directed graph of actions and transitions. Transitions are crossed when actions are completed; activities can be performed either in parallel or sequentially[2]

2.3.4 Appointment Management Activity Diagram and its Textual Description

The following figure (Figure2.6) represents the Activity Diagram of the System :

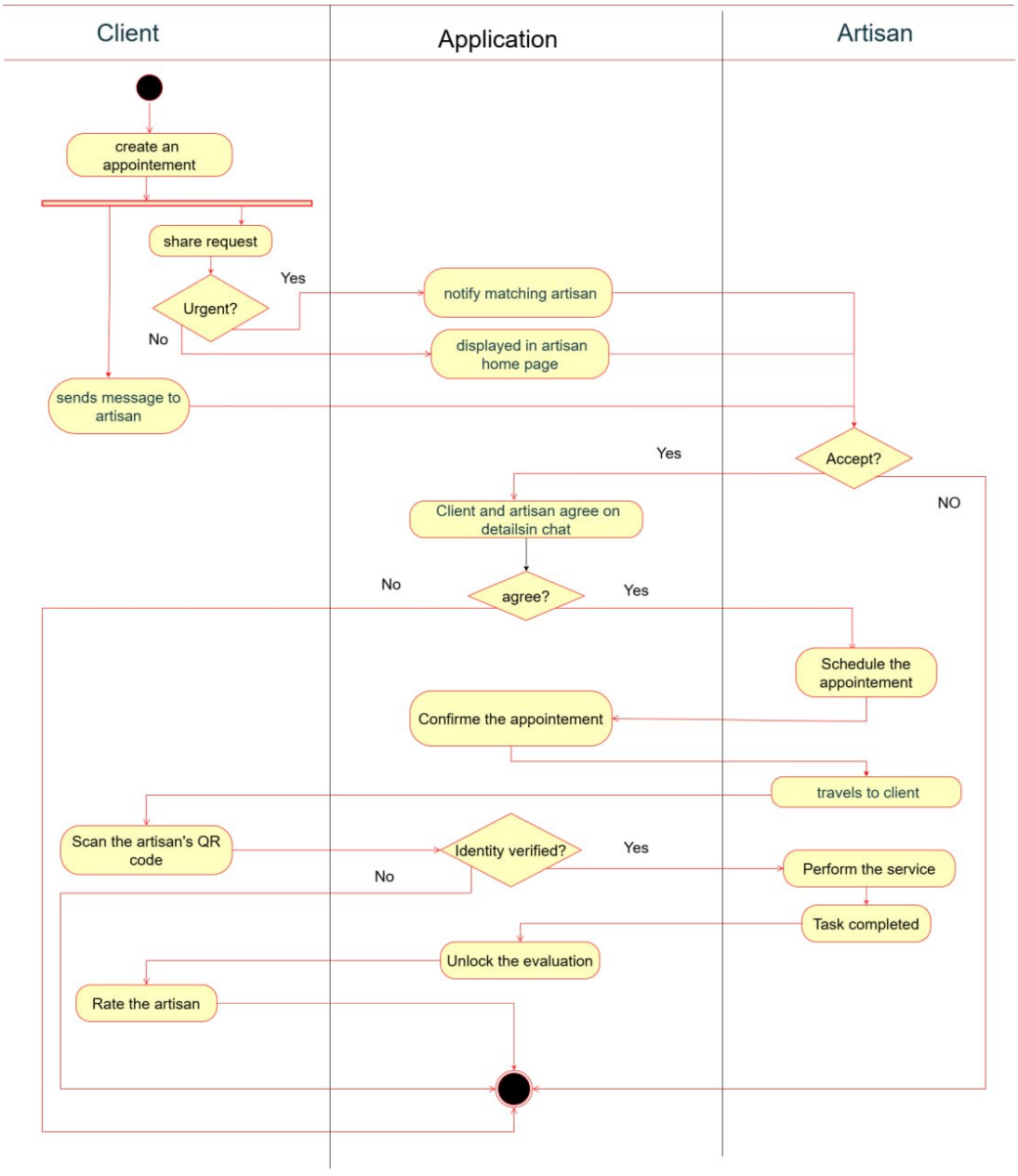


Figure 2.6: Appointment Management Activity Diagram

We present the textual description of the previous activity diagram(Figure2.6) :
The service request process revolves around three main actors: the **Client**, the **Application**, and the **Artisan**. It unfolds as follows:

Service Request Initiation: The client initiates the process by creating a service request through the application. This request can be of three types: **urgent**, **normal**, or in the form of a **direct message** sent to the artisan. Depending on the selected type, the application notifies the matching artisans (urgent case) or displays the request on the artisans' home page (normal case).

Artisan Request Processing: The artisan reviews the request and decides whether to **accept** or **decline** it. In the event of a refusal, the process ends. If accepted, both parties enter a negotiation phase through the integrated messaging system in order to agree on the details of the service. If no agreement is reached, The process is completed.Otherwise, an appointment is scheduled.

Confirmation and Verification: The appointment must be confirmed through the application. On the day of the appointment, the artisan travels to the client's location. Upon arrival, the client verifies the artisan's identity by **scanning their QR code**. If the identity is not confirmed, the process stops. Otherwise, the artisan proceeds with the execution of the service.

Service Completion: Once the service is completed, the application automatically unlocks the evaluation module. The client then **rates the artisan**, which marks the end of the process.

2.4 Design

2.4.1 Definition of the Class Diagram

The class diagram is generally considered **the most important diagram** in object-oriented development. It represents the conceptual architecture of the system: it describes the classes used by the system, as well as the relationships between them, whether they represent a **conceptual hierarchy** (inheritance) or an **organic relationship** (aggregation)[2]

2.4.2 Data Dictionary

- The following table (Table2.6)represents the data dictionary of our system:

Classe	Attributs				Methodes
	Name	Description	Type	Size	
User	username	Unique name of the user	A	50	Login() Logout() UpDateProfile() ChangePassword()
	email	Email address of the user	AN	100	
	password	Encrypted password	AN	255	
	creationDate	Date creation of user account	Date		
	phoneNumber	Phone number of the user	A	15	
Client	Id_client	Unique identifier of the client	A	10	UpDatePayement()
Administrator	Id_admin	Unique identifier of the client	AN	10	
Moderator	Id_moderator	Unique identifier of the moderator	AN	10	ValidationArti-sanAccount() BanUser()
Artisan	Id_artisan	Unique identifier of the artisan	AN	10	UpDateBio() SetAvailability(Status) UploadDiploma() UploadID_card()
	location	Geographic location of the artisan	A	100	
	category	Trade category of the artisan	A	50	
	activeStatus	Indicates id artisan is active	B		
	diploma	Diploma or certification name	A	100	
	bio	Short biography of the artisan	AN	255	
	ID_card	Identity card document of the artisan	AN	50	
Report	Id_repot	Unique identifier of the report	AN	10	Submit_Report() Process_Report() Close_Report() sendNotification()
	description	Description of the reported issue	AN	255	
	Status	Current status of the report	A	20	
Review	Id_review	Unique identifier of the review	AN	10	Create_Review() CalculateRatingScore()
	comment	Comment left by the client	AN	255	
	creatAT	Date the review was created	Date		
	ratingScore	Score given to the artisan	N	1	
Request	Id_Request	Unique identifier of the request	AN	10	
	requestDate	Date the request was submitted	Date		
	status	Current status of the request	A	20	
	description	Description of the request	AN	255	
Urgent	deadline	Deadline date of the urgent request	Date		sendNotification(post)
	priorityLevel	Priority level of the request	N	1	
Simple	messageContent	Content of the simple request message	AN	255	
Appointement	Id_appointement	Unique identifier of the appointment	AN	10	Confirm_Appointe() Cancel_Appointe() Complete_Appointe() sendNotification()
	scheduleTime	Scheduled date and time of the appoint-ment	Date-Time		
	status	Current status of the appointement	A	20	
Message	Id_message	Unique identifier of the message	AN	10	Sendmsg() DeleteConversation() sendNotification(message)
	text	Text content of the message	AN	255	
	timestamp	Date and time the message was sent	Date-Time		
Post	Id_post	Unique identifier of the post	AN	10	Publish_Post() edit_post(newContent) remove_post() approve_post() sendNotification()
	Content	Content of the post	AN	255	
	approvalStatus	Approval status of the post	A	20	
	creatAT	Date the post was created	Date		
	image	URL or path of the image	AN	255	
Follow	Id_follow	Unique identifier of the follow	AN	10	Follow() Unfollow()
	followingDate	Date of the following	Date-Time		

Table 2.6: Data dictionary

2.4.3 The Management Rules

The following business rules govern the behavior and constraints of the proposed system:

1. **A user** must have a unique username, email address, password, creation date, and phone number.

2. A user can hold one of the following roles: **Client, Artisan, Moderator, or Administrator**. These roles are mutually exclusive and are implemented through inheritance from the abstract class User.

3. A Moderator is **responsible for validating** artisan accounts and is **authorized** to ban users who violate platform policies.

4. A Moderator manages one or more Reports submitted by clients. Each Report is associated with exactly one Moderator.

5. **A Client** may submit one or more Requests. Each Request is associated with exactly one Client.

6. A Request is specialized into one of two subtypes: **Urgent**, characterized by a deadline and a priority level, or **Simple**, characterized by a text message. These subtypes are mutually exclusive.

7. Only **Urgent requests** trigger a push notification, invoked via the `sendNotification()` operation defined in the Urgent class.

8. A Client may follow zero or more Artisans. Conversely, an Artisan may be followed by zero or more Clients. This many-to-many relationship is reified through the association class Follow, which records the follow date and active status.

9. A Client may schedule zero or more Appointments with an Artisan. Each Appointment is associated with exactly one Client and exactly one Artisan, and is characterized by a scheduled time and a status.

10. An Artisan may publish one or more Posts. Each Post must receive an administrative approval before becoming publicly visible, as indicated by the `approvalStatus` attribute.

11. A Client may submit a Review for an Artisan, comprising a textual comment and a numerical rating score. The Review records the creation date and contributes to the Artisan's overall rating, computed via the `calculateRatingScore()` operation.

12. A Client and an Artisan may exchange Messages. This many-to-many interaction is mediated through the Message class, which stores the message text and timestamp.

13. An Artisan must upload a professional diploma and a valid identity card document as part of the registration process. These documents are subject to verification by a Moderator.

14. An Artisan's account status (`ActiveStatus`) must be set to `APPROVED` before the Artisan is permitted to accept Appointment requests or have Posts published.

2.4.4 Presentation of the Class Diagram

The following figure(Figure2.7)illustrates the class diagram of the application:

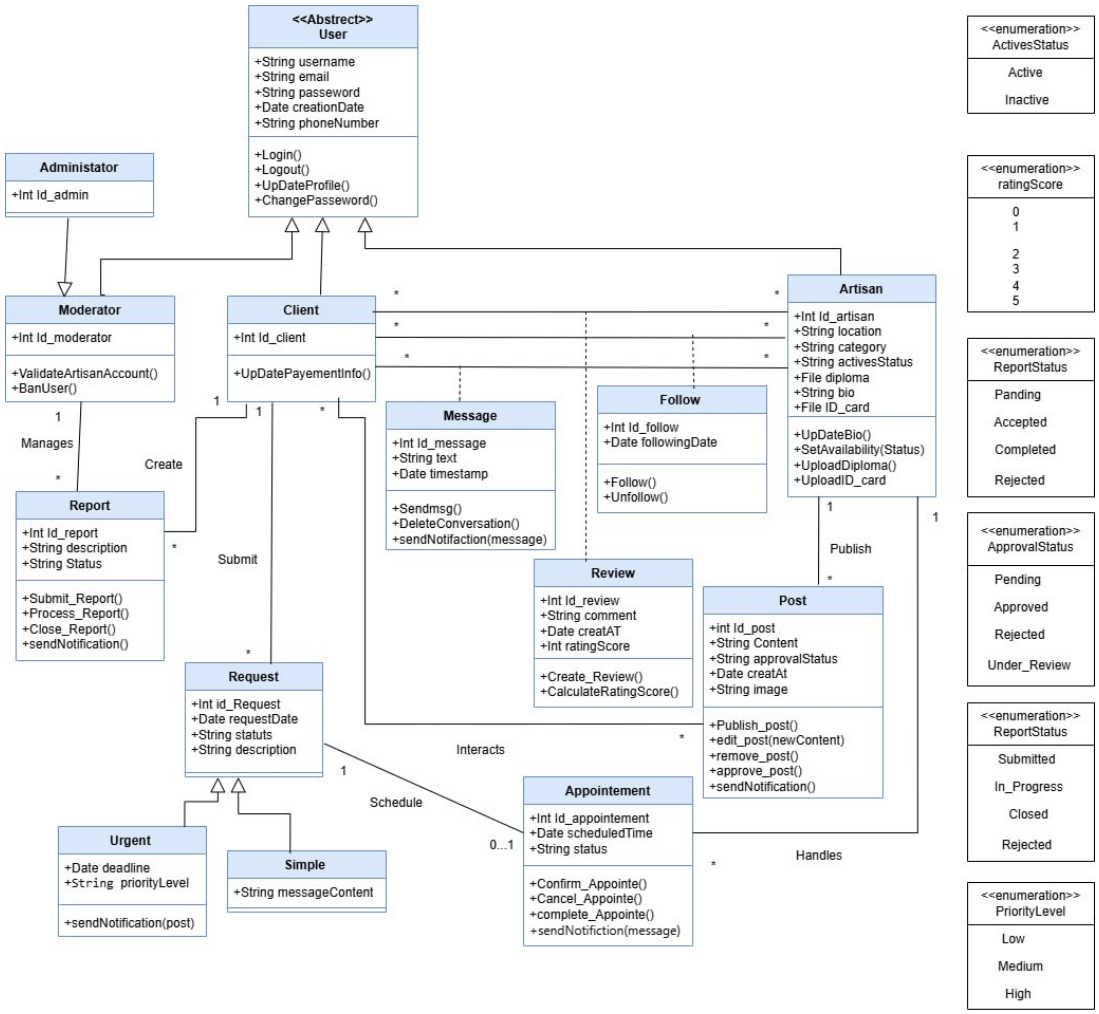


Figure 2.7: Class Diagram

2.4.5 From the Class Diagram to the Relational Model

The transformation from the object-oriented class diagram to the relational model is governed by a set of standard mapping rules, applied as follows:

- **Rule 1 — Class mapping:** Each class is mapped to a relational table. The attributes of the class become the columns of the table, and the identifier attribute becomes the primary key.
- **Rule 2 — One-to-one association:** Each one-to-one (1:1) association is handled by migrating the primary key of one relation into the other as a foreign key.
- **Rule 3 — One-to-many association:** Each one-to-many (1:N) association is handled by migrating the primary key of the relation on the ”one” side into the relation on the ”many” side as a foreign key.

- **Rule 4 — Many-to-many association:** Each many-to-many (M:N) association is handled by creating a new relation whose primary key is the concatenation of the primary keys of the participating relations. Any attributes belonging to the association class are also included in this new relation.

Applying the above mapping rules to the class diagram(Figure:2.7), the resulting relational schema comprises the following relations:

User(id_user, username, email, password, creationDate, phoneNumber)

Client(id_client, #id_user)

Artisan(id_artisan, location, category, activesStatus, bio, diploma, ID_card, #id_user)

Moderator(id_moderator, #id_user)

Administrator(id_admin, #id_user)

Report(id_report, description, status, #id_moderator)

Request(id_request, requestDate, status, description, #id_client)

Post(id_post, content, approvalStatus, creatAt, image, #id_artisan)

Urgent(id_urgent, deadline, priorityLevel, #id_request)

Simple(id_simple, messageContent, #id_request)

Appointment(#id_client, #id_artisan, scheduledTime, status)

Follow(#id_client, #id_artisan, followingDate, isActive)

Message(id_message, text, timestamp, #id_client, #id_artisan)

Review(id_review, comment, creatAT, ratingScore, #id_client, #id_artisan)

2.5 Conclusion

This chapter covered the **three essential stages** of the development of our system.**The requirements expression** made it possible to identify the actors and their functionalities through **the use case diagram**. **The analysis phase** described the dynamic behavior of the system using **sequence and activity diagrams**. Finally, **the design phase** formalized the system architecture through **the class diagram**. At the end of this chapter, we now have a solid and coherent foundation that will guide **the implementation phase** discussed in **the following chapter**.

Chapter 3

Implementation

3.1 Introduction

This chapter is dedicated to the **implementation phase** of our **Allo Artisan** application. We will begin by presenting the working environment by describing the languages, frameworks, and tools used during the development process, as well as the system deployment architecture.

Subsequently, we will highlight the graphical interfaces of the application through commented screenshots, thus reflecting the final result of the development work carried out for the two main actors: the **client** and the **artisan**.

3.2 Working Environment

In this part, we will present the tools that we will use to develop our application.

3.2.1 Language and Framework

Go language

An open-source programming language supported by Google, Easy to learn and great for teams, Its Built-in concurrency and a robust standard library,it is used to develop the backend and expose REST API. [4]



Figure 3.1: Logo of Go language

Flutter

Flutter is an open-source framework a Google based on the Dart language, allowing the development of mobile applications multi-platform applications from a single code base, it was used to design the user interface of our application [5]



Figure 3.2: Logo of Flutter

3.2.2 Development Tools

VS Code

A standalone source code editor that runs on Windows, macOS, and Linux, Mainly used for backend development in Go. [6]



Figure 3.3: Logo of Visual studio

Android Studio

The official IDE for Android application development, The emulation and testing of the Flutter application on virtual device. [7]



Figure 3.4: Logo of Android Studio

Docker

A containerization tool that allows the backend and its dependencies to be packaged into isolated containers, ensuring stable and portable deployment. [8]



Figure 3.5: Logo of docker

PostgreSQL

PostgreSQL is a powerful, open source object-relational database system, it was used to store and manage all the data of our application. [9]



Figure 3.6: Logo of PostgreSQL

3.3 Deployment Diagram

3.3.1 Definition of the Deployment Diagram

A deployment diagram describes the physical arrangement of the hardware resources that make up the system and shows the distribution of components across this hardware. Each resource is represented by a node, and the deployment diagram specifies how the components are distributed

on the nodes as well as the connections between the components or the nodes. Deployment diagrams exist in two forms: specification and instance.[3]

3.3.2 Presentation of Deployment Diagram and its Textual Description

The following figure 3.7 presented the class diagram of the application:

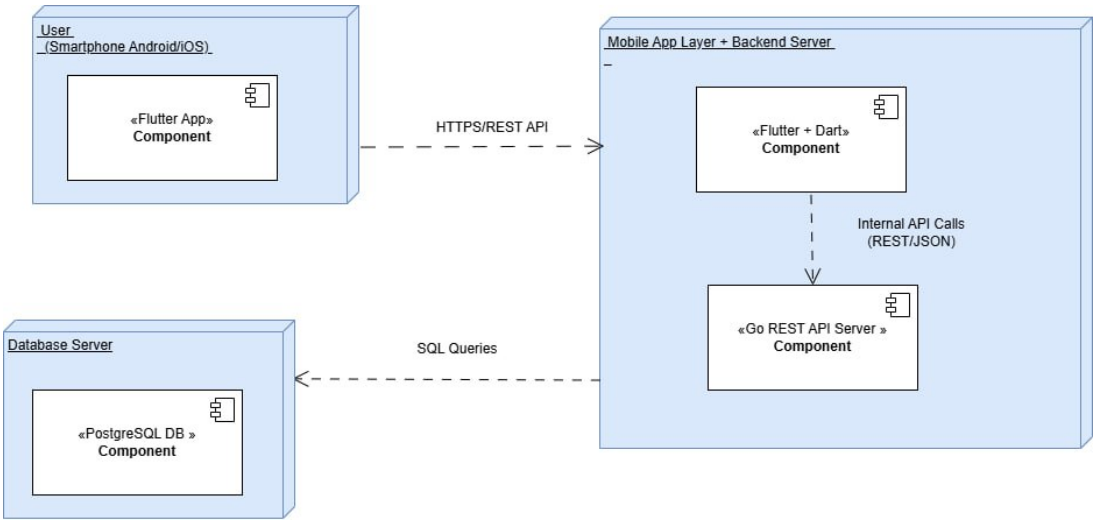


Figure 3.7: Deployment Diagram

The diagram is structured around three distinct deployment nodes, each grouping one or more software components that fulfill well-defined roles within the system.

1.1 Client Node - Smartphone Android/iOS This node represents the execution environment on the end-user side, namely a mobile device running an Android or iOS operating system. It hosts the following component:

- **«Flutter App»Component:** This is the mobile application developed using the Flutter framework. It constitutes the presentation layer of the system and is responsible for rendering the user interface, handling user interactions, and forwarding requests to the remote backend via the HTTPS/REST API interface.

1.2 Server Node - Mobile App Layer + Backend Server This node groups all server-side application components. It acts as an intermediary between the client layer and the persistence layer and hosts two distinct components:

- **«Flutter + Dart»Component:** This component represents the server-side application logic layer developed in Dart. It receives requests from the mobile application through the HTTPS/REST interface, processes them, and routes them to the internal API server via Internal API Calls using the REST/JSON format.

- **«Go REST API Server»Component:** This component constitutes the core of the business logic layer. Developed in the Go programming language (Golang), it exposes a RESTful API consumed by the Dart component and is responsible for executing SQL queries against the database management system.

1.3 Persistence Node - Database Server This node corresponds to the data persistence layer of the system. It hosts the following component:

- **«PostgreSQL DB»Component:** This is the PostgreSQL relational database management system (RDBMS).It ensures the storage, persistence, and integrity of application data. It receives SQL queries issued by the Go REST API Server and returns the corresponding results.

2. Communication Flows and Protocols

The interactions between the different system components are represented by dashed connectors, consistent with the UML notation for dependency relationships. The table below summarizes all identified communication flows:

Source	Destination	Protocol	Description
Flutter App (Client)	Flutter + Dart (Server)	HTTPS / REST API	Transmission of user requests to the backend via a secured API
Flutter + Dart (Server)	Go REST API Server	Internal API Calls (REST/JSON)	Routing of requests from the presentation layer to the business logic layer
Go REST API Server	PostgreSQL DB	SQL Queries	Execution of CRUD operations on the relational database

Table 3.1: Communication Flows and Protocols

3.4 Logo of the application



Figure 3.8: Logo of the application

Allo Artisan is a mobile application developed with the aim of facilitating the connection between clients looking for services and artisans offering their skills in various fields. The logo of Allo Artisan has been carefully designed to reflect the values and mission of the application. Each element carries a specific meaning: The house silhouette represents the home environment, which is the primary space where clients seek artisan services such as plumbing, electricity, painting, and general repairs. The wrench and screwdriver symbolize craftsmanship, technical expertise, and the wide range of skills that artisans offer through the platform. The speech bubble surrounding the tools illustrates communication and direct interaction between clients and artisans, emphasizing that the application serves as a bridge connecting the two parties. The circular arrow evokes reliability, continuity, and trust core values that the application strives to guarantee in every service exchange. The blue color was chosen to convey professionalism, confidence, and transparency, qualities that both clients and artisans can expect from the platform.

The name Allo Artisan was deliberately chosen to be simple, memorable, and meaningful.

3.5 Application Interfaces

In this part, we will present the interfaces of our application:

Login Interface

The adjacent figure (Figure 3.9)represents the Login interface of our Allo Artisan application:

This interface is the **first page displayed** to the user when launching the application. It offers **two login modes** through a ”Client” and ”Artisan” selector, allowing each type of user to authenticate according to their role. The user must enter their **email address or username**, as well as their **password**, to access their account. A **”Login”** button is provided to validate the authentication process. The application also offers the possibility to **”Continue as a Visitor”** in order to **access certain features without an account**. For new users, two links are available at the bottom of the page: **”Client Registration”** and **”Artisan Registration”**, each redirecting to the corresponding registration form.

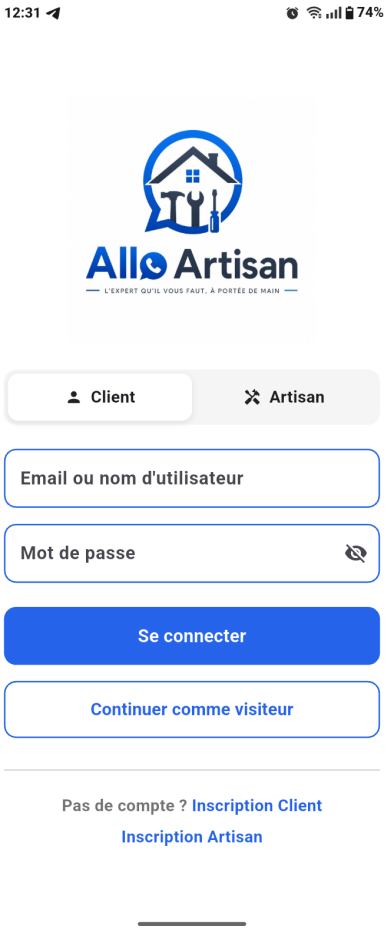


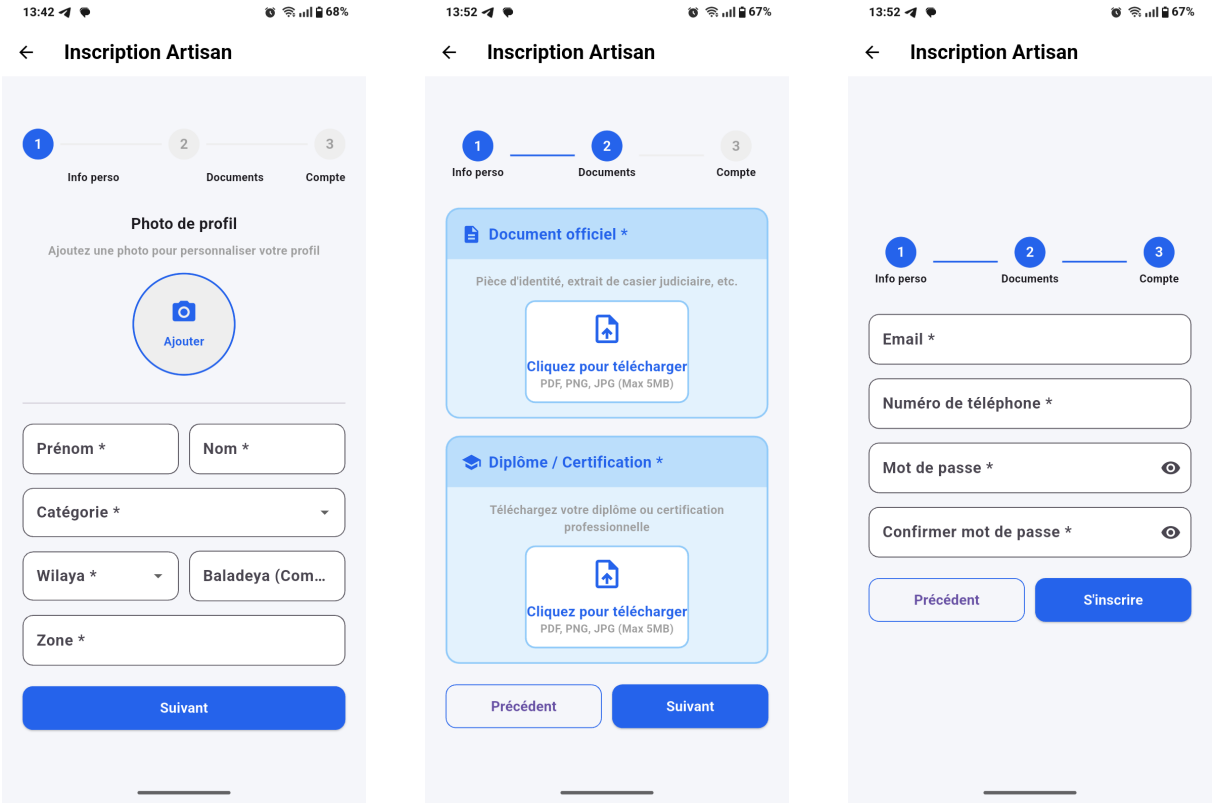
Figure 3.9: Login Interface

Create Artisan Account Interface

The figure above(Figure3.10) presents the artisan account creation interface in our Allo Artisan application :

- **Step 1** – Personal Information: The artisan must add a profile picture and fill in their personal information, including their first name, last name, professional category, wilaya, baladeya,municipality and intervention area. A “Next” button allows them to proceed to the next step.
- **Step 2** – Documents: The artisan must upload two required documents: an official document such as an identity card or criminal record extract, as well as a diploma or professional certification.A “Next” button allows them to continue.

- **Step 3** – Account: The artisan must enter their email address, phone number, password, and confirm the password. A “Sign Up” button completes the account creation process.



(a) Step 1 (b) Step 2 (c) Step 3

Figure 3.10: Create Artisan Account Interface

Client Home Page Interface

The figure adjacent (Figure 3.11) represents the client home interface of the Allo Artisan application. It displays the recent publications of artisans, where each publication includes:

- The artisan's profile picture, name, and professional category
- The artisan's rating
- An image illustrating the completed work
- A description of the service provided
- Interactions in the form of likes, comments, and shares

A **"+ Share a Request"** button allows the client to publish a new service request. At the top of the page, a search bar enables the client to quickly find an artisan, accompanied by a QR code scan button that allows the client to directly scan an artisan's QR code and access their profile instantly.

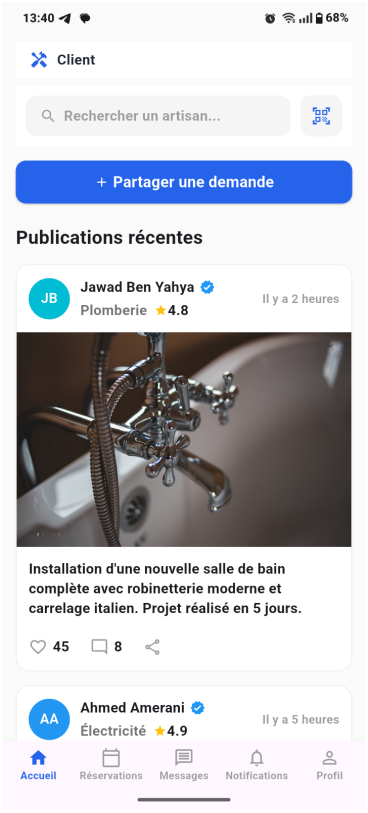


Figure 3.11: Client Home Page Interface

Client Share a Request Interface

The figure(Figure 3.14)represents the service request sharing interface of the **Allo Artisan** application. This interface appears as a popup when the client clicks on the **"+ Share a Request"** button. It offers two types of requests:

Normal Request: When the client selects the "Normal" mode, a blue message informs them that "Your request will be visible in the artisans' feed." The client must then fill in the service category, their area, and a detailed description of the request.

Urgent Request: When the client selects the "Urgent" mode, a red warning message indicates that "Active artisans will be notified immediately," ensuring a quick response. The client provides the same information as for a normal request.

In both cases, a **"Send"** button allows the client to submit the request, while a **"Cancel"** button closes the popup without saving.

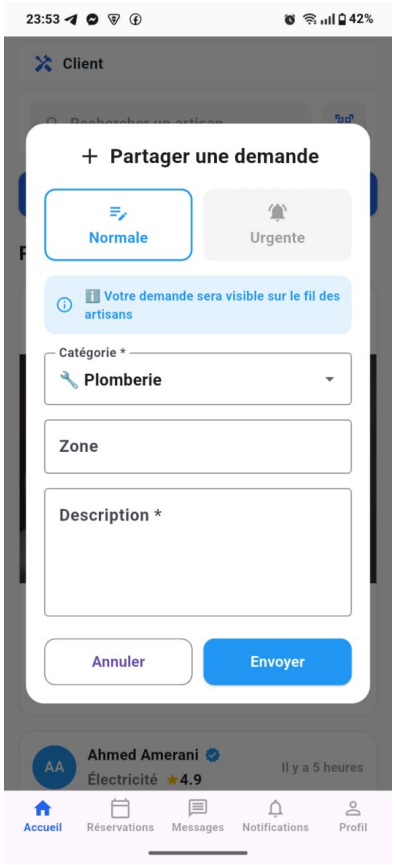


Figure 3.12: Normal Request

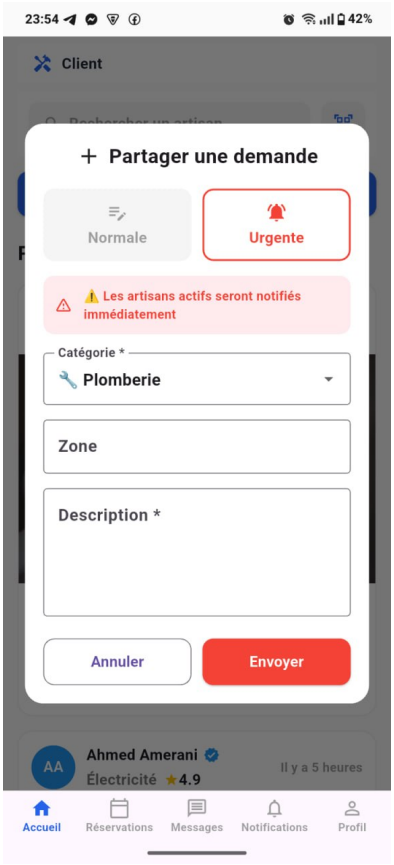


Figure 3.13: Urgent Request

Figure 3.14: Service Request Sharing Interface

Search for an Artisan Interface

The illustrated interface(Figure3.15) is a search screen of the Allo Artisan application. It offers three filters: the selection of the service category (here, plumbing), the maximum search distance using a slider, and the desired minimum rating. A “Reset” button allows the user to clear all filters with a single click

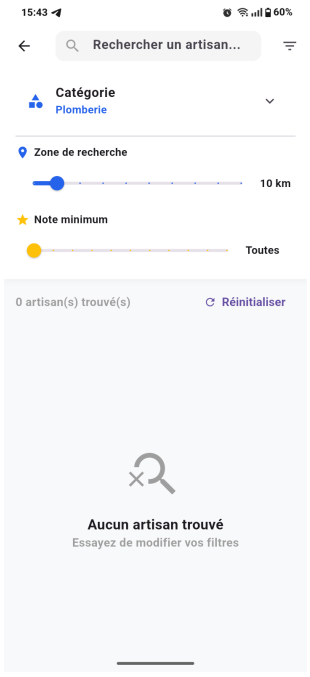


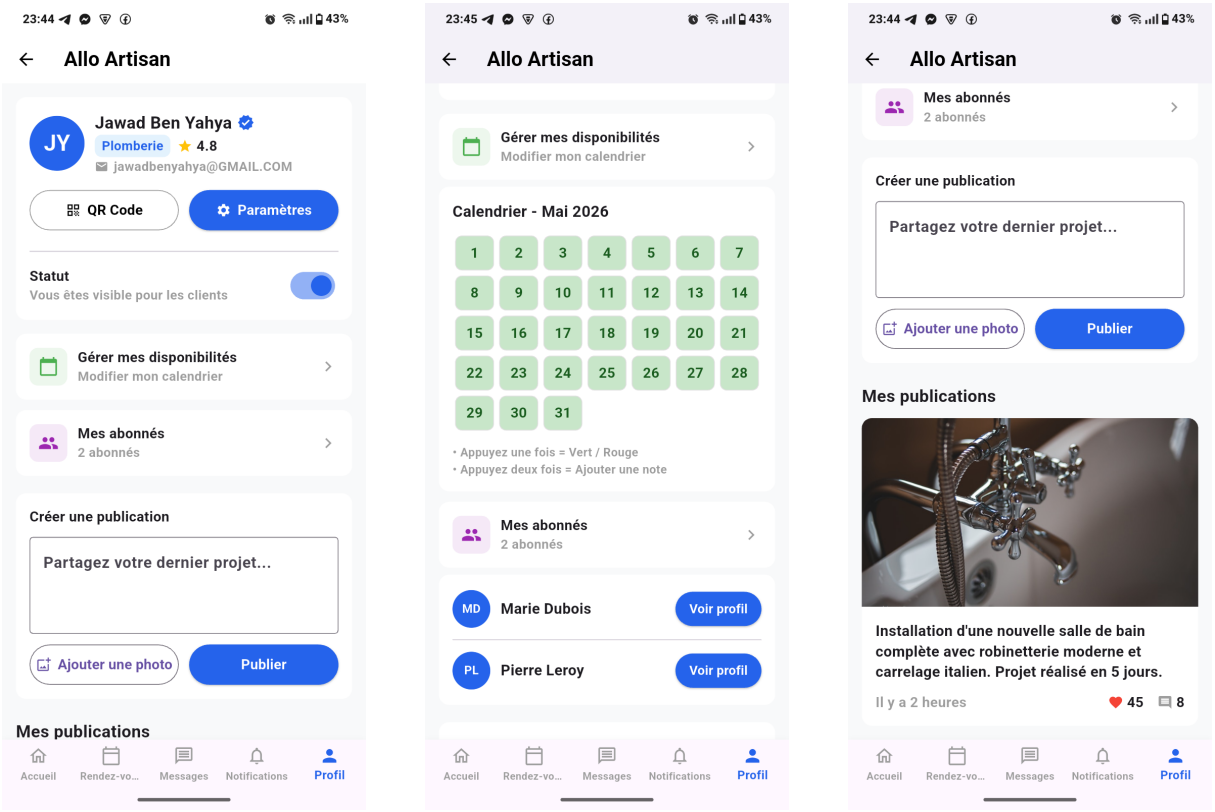
Figure 3.15: Search for an artisan Interface

Profile Artisan Interface

The figure above represents the profile interface of the Allo Artisan application on the artisan side, available in two views :

Private View (accessible by the artisan) : The artisan can consult and manage his complete profile. Displays his name, job category, rating, and email address. It also offers:

- A ”QR Code” button to generate his personal QR code
- A ”Settings” button to modify his information
- A visibility status that can be toggled to appear or not to clients
- Management of his availabilities via a calendar
- The list of his subscribers
- A ”Create a publication” section to share his work with a photo



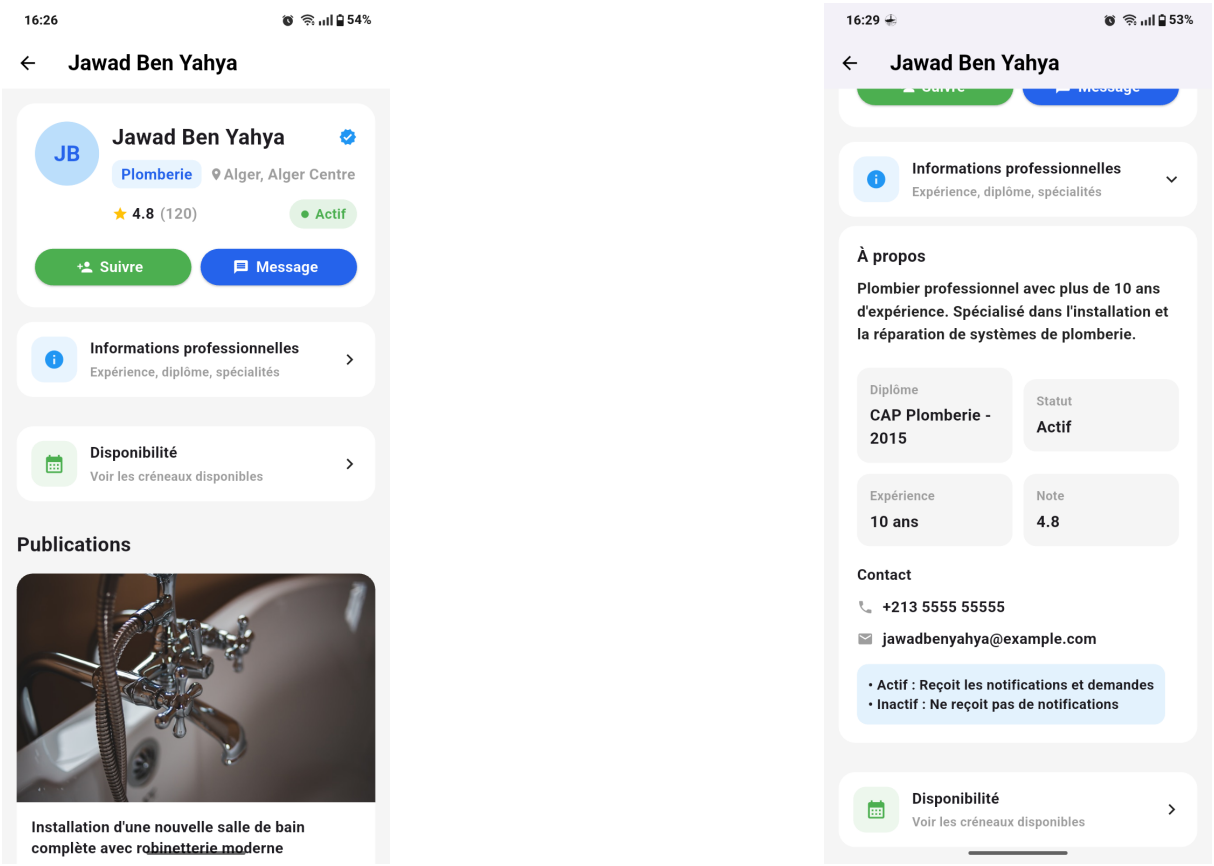
(a) Profile 1 (b) Profile 2 (c) Profile 3

Figure 3.16: Artisan Private Profile Interface

Public View (visible to clients): Clients can view the artisan’s public profile, which displays their name, category, rating, and email address. Two action buttons are available: ”Follow” and ”Message” allowing the client to follow the artisan or contact them directly. It also presents:

- Their personal QR code
- Their visibility status

- Their availability, including working days and hours
- The list of their followers
- Their published work posts that showcase completed services



(a) Public profile 1

(b) Public profile 2

Figure 3.17: Artisan Public Profile Interface

3.6 Conclusion

This chapter presents the implementation phase of our Allo Artisan application. We described the tools and technologies used during the development process,then illustrated the different interfaces of the application intended for both the client and the artisan. This chapter therefore concludes our project by translating all the previous stages into a concrete and functional mobile application

General Conclusion

This project was carried out with the objective of designing and developing **Allo Artisan**, a mobile application dedicated to facilitating the connection between clients and artisans in Algeria. Throughout the different phases of this work, we followed a structured and progressive methodology that ensured consistency between the identified requirements and the delivered solution. We began by analyzing existing platforms in the same domain, which allowed us to identify their key limitations and define a clear problem statement.
This study guided us in designing a solution structured around three axes: **trust, functionality, and usability** — materialized through features such as QR code-based identity verification, diploma validation, a multi-criteria search system, an availability calendar, and a post and notification mechanism for service requests.
The system was then modeled using a comprehensive set of UML diagrams before being implemented using **Flutter** for the frontend, **Go (Golang)** for the backend, with **PostgreSQL** as the database management system.
The resulting application offers a reliable and accessible experience for clients , artisans, and administrators alike.

As a perspective, future enhancements could include the integration of an **AI-based recommendation system**, an **in-app payment module**, and the development of a **complementary web platform** improvements that would further strengthen Allo Artisan as a scalable and sustainable solution for the digitalization of artisan services in Algeria.

Bibliography

[1] B. Charroux, A. Osmani, Y. Thierry-Mieg, *UML 2 – Pratique de la modélisation*, 2e édition, Pearson Education, collection Synthex.

[2] P. Roques, *UML 2 par la pratique – Études de cas et exercices corrigés*, 5e édition, Eyrolles.

Webography

[1] uml-diagrams.org, *UML Use Case Diagrams*, https://www.uml-diagrams.org/use-case-diagrams.html, Accessed: 2026-05-10.

[2] L. Audibert, *Cours UML - Diagramme de Classes*, https://laurent-audibert.developpez.com/Cours-UML/?page=diagramme-classes#L3, Accessed: 2026-05-18.

[3] L. Audibert, *Cours UML - Diagrammes de Composants et de Déploiement*, https://laurent-audibert.developpez.com/Cours-UML/?page=diagrammes-composants-deploiement#L8-3, Accessed: 2026-05-18.

[4] The Go Authors, *The Go Programming Language*, https://go.dev/, Accessed on May 22, 2026.

[5] Flutter, *Flutter*, https://flutter.dev/, Accessed on May 22, 2026.

[6] Visual Studio Code, *Visual Studio Code*, https://code.visualstudio.com/, Accessed on May 22, 2026.

[7] Android Studio, *Android Studio*, https://developer.android.com/studio, Accessed on May 22, 2026.

[8] Docker Inc, *Docker*, https://www.docker.com/, Accessed on May 22, 2026.

[9] PostgreSQL, *PostgreSQL*, https://www.postgresql.org/, Accessed on May 22, 2026.